A Deep Reinforcement Learning Approach for Cooperative Target Defense

Yanxue Xiong[1], Zhigang Wang[2], and Liangjun Ke[1(✉)]

[1] State Key Laboratory for Manufacturing Systems Engineering, School of Automation Science and Engineering, Xi'an Jiaotong University, Xi'an, China
keljxjtu@xjtu.edu.cn

[2] CETC Key Laboratory of Data Link Technology Xi'an, Xi'an, China

Abstract. This paper considers a new variant of the pursuit-evasion problem, called the cooperative target defense problem with three agents (attacker, targeter, and defender) in a 3D space. The targeter tries to fly as quickly as possible from a starting point to the terminal, while the defender seeks to protect it from the attacker. The problem is difficult to solve under traditional game theory methods, while deep reinforcement learning (DRL) has shown strong adaptability in these complex and higher-dimensional tasks. Inspired by the successful applications of Proximal Policy Optimization (PPO), this paper proposes a PPO-based algorithm for the problem, intending to derive the optimal behavioral policies for both sides. We design the corresponding state space, action space, and rewards of the agents. Three kinds of reward functions are proposed for the attacker and compared by experimental results. Our study provides a good foundation for the cooperative target defense problem.

Keywords: Pursuit-evasion game · Differential game · Deep reinforcement learning · Cooperative target defense

1 Introduction

The pursuit-evasion problem is an important problem in the fields of game theory and artificial intelligence, widely used in aerospace technology, robotics, police security, cyber security, and other fields [6,10]. It has many variants, one of them is Targeter-Attacker-Defender Game (TAD Game) [9]. In this game, there are three agents including an attacker, a targeter, and a defender. The attacker aims to pursue the targeter protected by a defender, while the targeter tries to move from a starting point to a terminal point as soon as possible.

For this game, Von Moll [17], Liang [8,9], and Zhou [19] have solved it based on differential game theory. And later, Fu et al. [3], and Lin et al. have gone further with other methods [10]. But in more complex situations, it becomes difficult or even unworkable for these conventional methods to solve the problem [13,15].

Y. Tan and Y. Shi (Eds.): DMBD 2022, CCIS 1744, pp. 17–26, 2022.
https://doi.org/10.1007/978-981-19-9297-1_2

Reinforcement learning is one of the main paradigms of machine learning, which uses rewards to guide the learning process [14]. As an important tool for multistage decisions, it has been used for the pursuit-evasion problem and its variants [4]. Motivated by the prominent representation ability of deep learning [7], deep reinforcement learning (DRL), a unified framework of reinforcement learning and deep learning, has been widely studied since 2015 [11]. In recent years, deep reinforcement learning has been successfully applied in video games [16], robots [1], and other various fields [2]. Since deep Q-learning was proposed [11], many popular deep reinforcement learning algorithms have been proposed. Among them, the Proximal Policy Optimization (PPO) algorithm [12] is very promising.

In this paper, we consider a special Targeter-Attacker-Defender Game (TAD Game) in which all agents stay in a three-dimensional environment, which is studied for the first time. There are still bottlenecks in the exploration of strategies and the chain reaction caused by the rising dimensionality. To deal with such a complex problem, we design a PPO-based algorithm by taking into account the fundamental characteristics of the agents. Particularly, we propose three ways of reward setting for the attacker. The effectiveness of reward settings and the algorithm is studied according to experimental results.

The remaining of this paper is structured as follows. In Sect. 2, we present the basic process of PPO and MAPPO [18] which is an effective multi-agent version of PPO. In Sect. 3, the formal description of the 3D TAD problem is presented. In Sect. 4, we present the detail of our algorithm. In Sect. 5, the experimental study is presented. The main results are summarized in Sect. 6.

2 Background

The basic process of our algorithm is based on PPO. As reinforcement learning is usually described by a Markov decision process, an agent selects an action a_t in its policy π to interact with the environment under the state s_t. Then it gets a feedback reward r_t and steps into the next state. π is assigned a parameter θ. The training purpose is achieved by using a policy $\pi_{\theta'}$, which has a small difference from π_θ, to collect data as a demonstration to update θ.

The concept of deep learning was introduced in 2006 by G.E. Hinton et al. [5]. It is suggested that Deep Neural Networks (DNN) structure can be used in training sample data, which makes it easier to realize our learning tasks. The classical Actor-Critic network is adopted, in which the actor network outputs actions according to the state; the critic network outputs the value, which is used to evaluate the actions outputted by the actor network.

Thus, for a trajectory $\tau = \{s_1, a_1, s_2, a_2, \ldots, s_t, a_t\}$, the probability of its occurrence is described as $p_\theta(\tau)$. The return can be calculated by Eq. 1 where γ is the discount factor:

$$R(\tau) = \sum_{t=t_0}^{T} \gamma^{t-t_0} r_t. \tag{1}$$

The advantage function is described as

$$A^{\theta}\left(s_t, a_t\right) = \sum_{t=t_0}^{T_n} \left(R\left(\tau^n\right) - b\right) = \sum_{t=t_0}^{T} \gamma^{t-t_0} r_t - b, \tag{2}$$

where the baseline b is usually regarded as the value function.

$$\begin{aligned} &J_{PPO}^{\theta^k}(\theta) \\ &\approx \min\left(\frac{p_\theta\left(a_t \mid s_t\right)}{p_{\theta^k}\left(a_t \mid s_t\right)} A^{\theta^k}\left(s_t, a_t\right), clip\left(\frac{p_\theta\left(a_t \mid s_t\right)}{p_{\theta^k}\left(a_t \mid s_t\right)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}\left(s_t, a_t\right)\right). \end{aligned} \tag{3}$$

In order to optimize the objective function of the PPO-clip in Eq. 3, the next step is to update the AC network by backward the loss functions which are

$$\begin{aligned} loss_{Actor} &= -min(\frac{p_\theta\left(a_t \mid s_t\right)}{p_{\theta^k}\left(a_t \mid s_t\right)} A^{\theta^k}\left(s_t, a_t\right), clip(\frac{p_\theta\left(a_t \mid s_t\right)}{p_{\theta^k}\left(a_t \mid s_t\right)}, 1-\varepsilon, 1+\varepsilon) A^{\theta^k}\left(s_t, a_t\right)), \\ loss_{Critic} &= \left(G_t - Value\right)^2, \end{aligned} \tag{4}$$

where G_t is the actual discounted return.

Similar to the PPO algorithm, a centralized value function network is designed in the MAPPO algorithm. With centralized training and decentralized execution, agents are able to interact with each other through a global value function. And some additional techniques are used to improve the efficiency of the algorithm [18].

3 The Problem Description

In this section, we formally describe the cooperative target defense problem. In a three-dimensional environment, there are three agents, namely Attacker(**A**), Targeter(**T**), and Defender(**D**). The task of **A** is to attack **T** which attempts to reach a terminal point safely under the protection of **D**. A demo scenario is shown in Fig. 1.

In this game, these agents play under the following rules:

- **A, T, D** move with constant velocity magnitudes v_A, v_T, v_D ($v_A = v_D > v_T$) respectively. But each agent can control its yaw and pitch angle to change the moving direction. The change of the yaw and pitch angle must be less than $\pi/2$ and $\pi/4$. The turning radius is determined by its axis length and angle change.
- **T**'s initial position is $T_0\left(x_{T_0}, y_{T_0}, z_{T_0}\right)$, while $ter($ x_{ter}, y_{ter}, $z_{ter})$ is its terminal point. The distance between them is denoted as $|T_0 - ter|$, and the gaming time is limited within $\Gamma = \frac{|T_0 - ter|}{V_T} \times 2$.

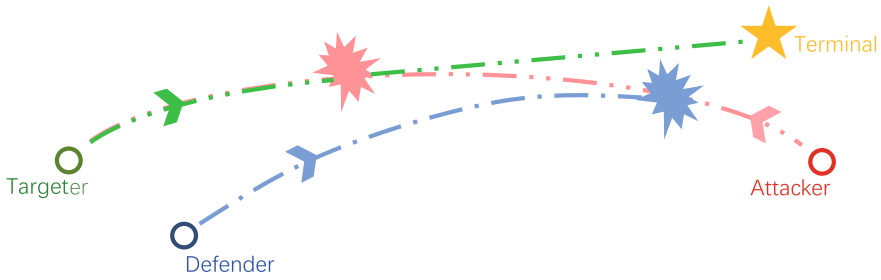

Fig. 1. Schematic diagram of the three dimensional Targeter-Attacker-Defender problem

- **A** can capture **T** within the distance of $|V_A|$ around it. T will be captured if the current $A_t\,(x_{A_t}, y_{A_t}, z_{A_t})$ and $T_t\,(x_{T_t}, y_{T_t}, z_{T_t})$ satisfies $|A_t - T_t| \leq V_A$. Similarly, **D**'s capture range and terminal spherical are defined.
- If **A** captures **T** or dispels **T** away from the terminal within the time limit of the game, the game ends and **A** wins; if **T** enters the terminal in time or **D** captures **A** before **T** is captured, the game ends and the TD team wins.

At time t, let $d_{AT}(t)$, $d_{DA}(t)$, and $d_{Tter}(t)$ denote the distance between **A** and **T**, the distance between **D** and **A**, the distance between **T** and terminal respectively. In order to win the game, **A** tries to minimize J_A which is defined as follows:

$$J_A = w_{\mathrm{A}_1} d_{AT}(t) - w_{\mathrm{A}_2} d_{DA}(t), \tag{5}$$

where w_{A_1} and w_{A_2} are two positive weights.

In order to win the game, the TD team tries to minimize J_T and J_D which are defined as follows:

$$\begin{aligned} J_T &= w_{\mathrm{T}_1} d_{Tter}(t) - w_{\mathrm{T}_2} d_{AT}(t), \\ J_D &= d_{DA}(t), \end{aligned} \tag{6}$$

where w_{T_1} and w_{T_2} are two positive weights.

4 The Proposed Algorithm

For the attacker, a PPO algorithm is proposed for policy optimization. Since the targeter and defender are cooperative, a multi-agent PPO [12] is proposed for policy optimization.

4.1 State and Action Space

In the coordinate system $o - xyz$, the motion of each agent can be decomposed into the xoy and yoz planes (see a demo of T in Fig. 2). Then the action space (φ_i, θ_i) is defined by $(\varphi_i \in [0, 2\pi), \theta_i \in [-\pi/2, \pi/2])$ where $i \in \{\mathbf{T},\mathbf{A},\mathbf{D}\}$. φ, θ are the yaw and pitch angle respectively. The state of each agent is $(A_t, D_t, T_t, \varphi_{At}, \theta_{At}, \varphi_{Dt}, \theta_{Dt}, \varphi_{Tt}, \theta_{Tt})$.

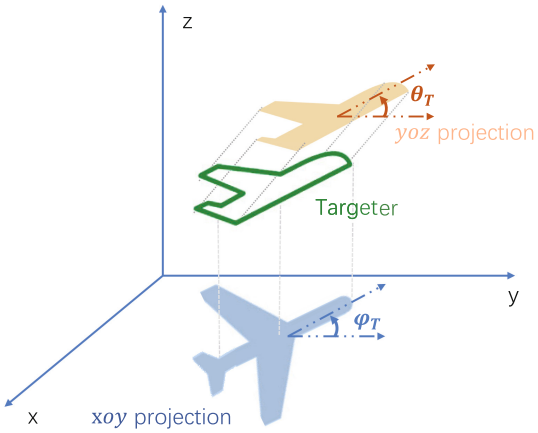

Fig. 2. Illustration of Targeter's action space.

4.2 Reward Functions

For the attacker, the basic idea to define the reward is to encourage it to stay away from the defender and get close to the targeter. Three methods are suggested to define the reward for the attacker according to different considerations.

1) method 1: the reward is defined according to whether **A**'s behavior is favorable for capturing **T** and avoiding **D**. Define $\Delta AT = |A_{t-1} - T_{t-1}| - |A_t - T_t|$, $\Delta AD = |A_{t-1} - D_{t-1}| - |A_t - D_t|$.

$$r_A = \begin{cases} 45/\Gamma & \text{if} \quad \Delta AT > 0.1 \wedge \Delta AD \leq 0.1 \\ 30/\Gamma & \text{if} \quad \Delta AT > 0.1 \wedge \Delta AD > 0.1 \\ 3/\Gamma & \text{if} \quad \Delta AT \leq 0.1 \wedge \Delta AD \leq 0.1 \\ 0 & \text{if} \quad \Delta AT \leq 0.1 \wedge \Delta AD > 0.1 \end{cases}. \tag{7}$$

2) method 2: At the current time, let $\overrightarrow{\sigma_A}$ and $\overrightarrow{\sigma_A}^*$ be **A**'s actual and reference direction respectively where the reference direction is calculated by the velocity of targeter at the last timestep. The angle between them is written as $\Delta\sigma_A$. σ_a stands for the acceptable angular offset. The reward is defined according to the magnitude of $\Delta\sigma_A$.

$$r_A = \begin{cases} max(2 - \Delta\sigma_A/\sigma_a, 1)/\Gamma \times 30 & \text{if} \quad \Delta\sigma_A \leq \sigma_a \\ max(2 - \Delta\sigma_A/\sigma_a, 0)/\Gamma \times 30 & \text{if} \quad \sigma_a < \Delta\sigma_A \leq 2\sigma_a \\ max(1 - \Delta\sigma_A/\sigma_a, -2)/\Gamma \times 30 & \text{if} \quad \Delta\sigma_A > 2\sigma_a \end{cases}. \tag{8}$$

3) method 3: Define λ_t as the distance between A_t and the predicted collision of **A** and **T** at time t. To penalize the occurrence of over-chasing, set a counter C to record the times of the situation. The reward is defined according to

whether λ_t becomes smaller in contrast to λ_{t-1}.

$$r_A = \begin{cases} min(2|\lambda_t - \lambda_{t-1}| + 1, 2)/\Gamma \times 10 & \text{if} \quad \lambda_{t-1} - \lambda_t \geq 0.05 \\ -max(-2|\lambda_t - \lambda_{t-1}|, -1)/\Gamma \times 10 - \gamma^C & \text{if} \quad \lambda_{t-1} - \lambda_t < 0.05 \end{cases}. \tag{9}$$

For the targeter, the reward is given as follows:

$$r_T = w_{\mathrm{T}_1} d_{Tter}(t) - w_{\mathrm{T}_2} d_{AT}(t). \tag{10}$$

For the defender, the reward r_D is given as follows:

$$r_D = d_{DA}(t). \tag{11}$$

5 Experimental Results and Analysis

Experiments are conducted in a simulation environment based on Python version 3.7. The initial angles are $\varphi_i = \theta_i = 0 (i \in \{\mathbf{T},\mathbf{A},\mathbf{D}\})$, speeds are $|V_A| = |V_D| = 0.18, |V_T| = 0.13$, axis lengths are $L = 0.1$ and terminal point is $(0, 0, 0)$. The training process is stopped when the maximum number of episodes reaches 10,000. Take w_{T_1} to be 0.6, w_{T_2} to be 0.4, and γ to be 1.08. When training the attacker, the defender moves towards the attacker and the targeter moves close to the terminal while avoiding the attacker. After that, the TD team is trained.

5.1 The Experimental Study on the Reward Methods for the Attacker

At first, we compare the three methods of defining reward for the attacker. Each model (i.e., each method) is tested for 100 rounds. The winning percentages are shown in Table 1. As seen from Table 1, the attacker trained by method 2 can complete the task the best, while the agent trained by method 1 performs the poorest.

Some policies that attacker had taken are given in Fig. 3. As seen from Fig. 3(a), Fig. 3(b), and Fig. 3(c), once in a safe area, a smart **A** would purely pursue **T** (i.e., without considering **D**). Especially, due to the condition of $|v_A > v_T|$ and the limit in turning angle, as shown in Fig. 3(a) and Fig. 3(b), **A** cannot capture **T** directly through the shortest distance, instead it outflanks **T**. Also, a smart **A** can try its best to pursue **T**, avoiding flying near **D** when there is a threat from **D**, like the policies shown in Fig. 3(d), Fig. 3(e), and Fig. 3(f).

Table 1. A's winning percentage obtained by the three reward methods.

	Method 1	Method 2	Method 3
Winning percentage	0.4800	**0.6253**	0.5763

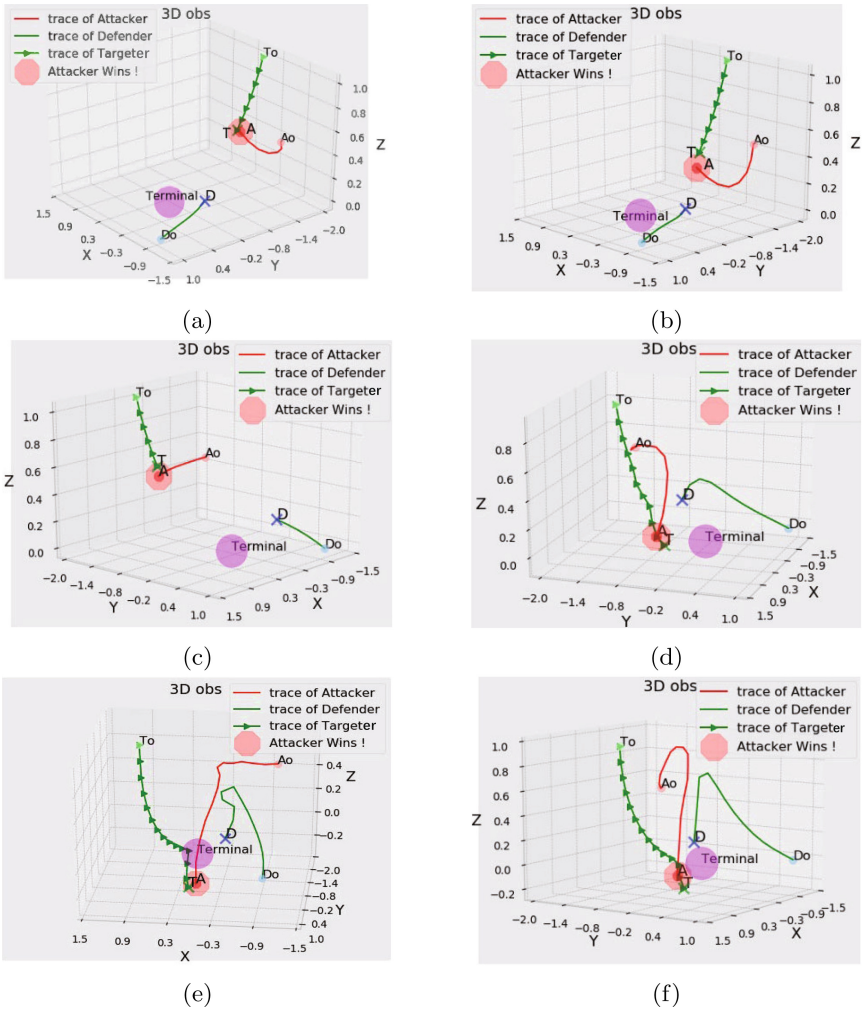

Fig. 3. Several snapshots where attacker successfully completes the task

5.2 The Experimental Study on the Behavior of Defender and Targeter

Figure 4 shows some of the policies the TD team had adopted. According to Fig. 4, when **T** moves straight toward the terminal (see Fig. 4(a) and Fig. 4(b)) or **D** intercepts **A** (see Fig. 4(c)), the whole team can win the game. It is obvious that the agents learn the trick that if both of them try their best to complete their own tasks, the chances of winning can be improved. So they also perform as shown in Fig. 4(d), Fig. 4(e), and Fig. 4(f). Moreover, in Fig. 4(c), the TD team is smart enough to cooperatively win the game. In detail, **T** lures **A** to fall into **D**'s capture region.

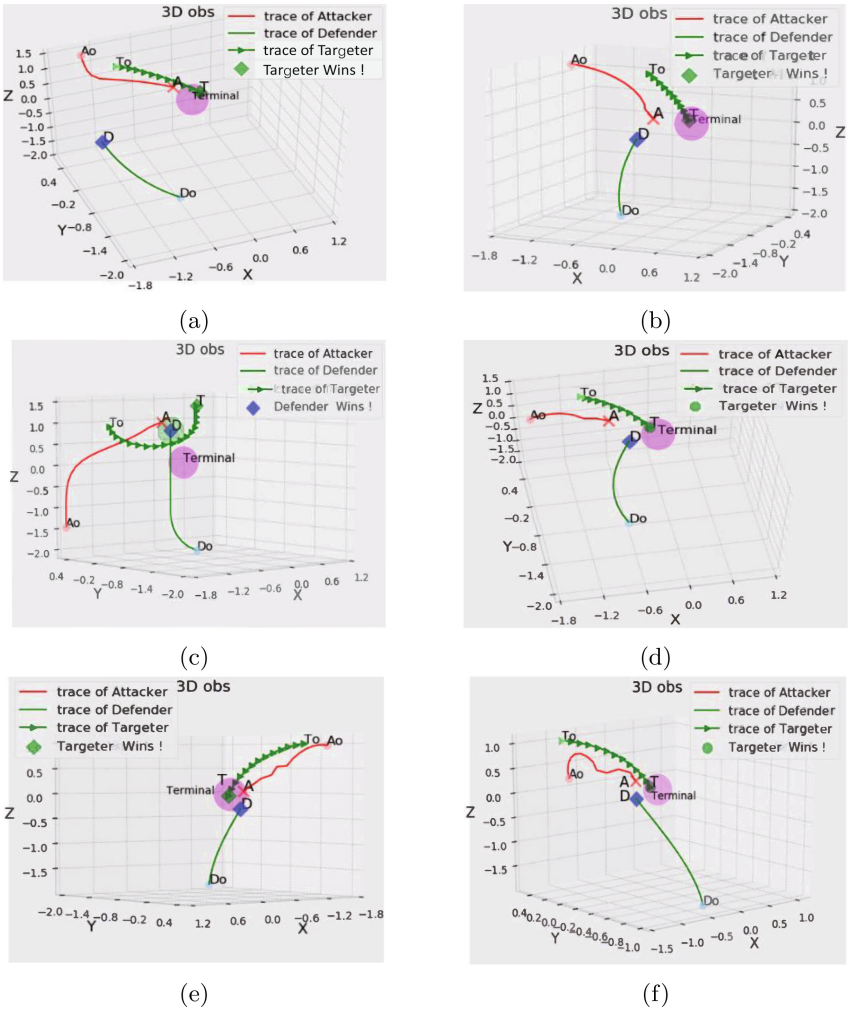

Fig. 4. Several snapshots where the TD team successfully completes the task

6 Conclusion

This paper studies the cooperative target defense problem for the first time. Since this problem is very difficult, we design a Proximal Policy Optimization based algorithm with the aim of obtaining the optimal behavioral policies. For every agent in the game, we design their corresponding state space, action space, and rewards. Moreover, for the attacker, we propose three kinds of reward functions. The experimental results show that method 2 is the best. Moreover, the effectiveness of the proposed algorithm is analyzed.

In the future, it may be interesting to use various reinforcement learning algorithms to deal with the cooperative target defense problem. In addition, it may be better to use the self-play technique to improve the behaviors of these agents.

Acknowledgment. This work was supported by the National Natural Science Foundation of China under Grant 61973244 and Grant 61573277. It is also supported by the open fund of CETC Key Laboratory of Data Link Technology (CLDL-20202101-1).

References

1. Andrychowicz, O.M., et al.: Learning dexterous in-hand manipulation. Int. J. Robot. Res. **39**(1), 3–20 (2020)
2. Degrave, J., et al.: Magnetic control of tokamak plasmas through deep reinforcement learning. Nature **602**(7897), 414–419 (2022)
3. Fu, H., Liu, H.H.T.: Optimal solution of a target defense game with two defenders and a faster intrude. Unmanned Syst. **9**(03), 247–262 (2021)
4. Givigi, S.N., Schwartz, H.M., Lu, X.: A reinforcement learning adaptive fuzzy controller for differential games. J. Intell. Rob. Syst. **59**(1), 3–30 (2010)
5. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
6. Kong, W., Zhou, D., Yang, Z., Zhao, Y., Zhang, K.: UAV autonomous aerial combat maneuver strategy generation with observation error based on state-adversarial deep deterministic policy gradient and inverse reinforcement learning. Electronics **9**(7), 1121 (2020)
7. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
8. Liang, L., Deng, F., Lu, M., Chen, J.: Analysis of role switch for cooperative target defense differential game. IEEE Trans. Autom. Control **66**(2), 902–909 (2020)
9. Liang, L., Deng, F., Peng, Z., Li, X., Zha, W.: A differential game for cooperative target defense. Automatica **102**, 58–71 (2019)
10. Lin, B., Qiao, L., Jia, Z., Sun, Z., Sun, M., Zhang, W.: Control strategies for target-attacker-defender games of USVs. In: 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE), pp. 191–198 (2021). https://doi.org/10.1109/CACRE52464.2021.9501329
11. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
12. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
13. Sun, W., Tsiotras, P., Lolla, T., Subramani, D.N., Lermusiaux, P.F.: Multiple-pursuer/one-evader pursuit-evasion game in dynamic flowfields. J. Guid. Control. Dyn. **40**(7), 1627–1637 (2017)
14. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
15. Tang, X., Ye, D., Huang, L., Sun, Z., Sun, J.: Pursuit-evasion game switching strategies for spacecraft with incomplete-information. Aerosp. Sci. Technol. **119**, 107112 (2021)
16. Vinyals, O., et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)

17. Von Moll, A., Casbeer, D.W., Garcia, E., Milutinović, D.: Pursuit-evasion of an evader by multiple pursuers. In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 133–142. IEEE (2018)
18. Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of PPO in cooperative, multi-agent games. arXiv preprint arXiv:2103.01955 (2021)
19. Zhou, Z., Zhang, W., Ding, J., Huang, H., Stipanović, D.M., Tomlin, C.J.: Cooperative pursuit with voronoi partitions. Automatica **72**, 64–72 (2016)