A Prioritized Experience Replay Based DDPG Approach to the Strategy Generation for UCAVs

Junsen Lu, Yun-Bo Zhao*, Yu Kang, Yuhui Wang and Yimin Deng

***Abstract*— Unmanned combat aerial vehicles are becoming essential participants in future air-combat scenarios, while the optimal control strategy remains a great challenge due to the high dynamics of the aerial vehicles themselves as well as the environmental uncertainties in air-combat. Based on a deep deterministic policy gradient algorithm framework, an air combat decision-making strategy is designed and implemented, and further a prioritized experience replay method is proposed for the proposed algorithm to further improve the efficiency in the training process. Simulation experiments show that, at much reduced training cost, the proposed approach achieves superior air combat performance with fast convergence.**

I. INTRODUCTION

Unmanned Combat Air Vehicles (UCAVs) have been developing fast in recent years and have shown their important role in several real-world air combats [1], serving as autonomous or semi-autonomous aerial offensive or defensive weapons, due to their advanced capabilities of detecting, tracking, etc. [2] Considering the extremely high dynamics of UCAVs themselves and the strong uncertainties in air combats, one key requirement for the development of UCAVs is the real-time effective strategies generation [3]. We have seen many pioneering approaches to this challenge, using the rule-based approach [4], the probabilistic model/fuzzy logic model [5], [6], the computational intelligence hybrid approach [7], [8], etc. But without any doubt further improvements are urgently needed.

We notice that in recent years the Reinforcement Learning (RL) based approach has been attracting more and more attentions in UCAVs strategy generation, since such an approach can greatly improve the robustness of UCAVs strategy generation [9], [10]. Many results focus on air combat maneuver decisions, but most of them are based on establishing tactical action sets and basic action sets, and discretizing the action space to achieve maneuver decision [11]. Although this kind of approach can quickly output the decision sequence, it is inconsistent with the actual air combat situation. The high discretization of the action space will lead to "dimension disaster". On the contrary, the low discretization will reduce the control accuracy.

This work was supported by the National Key Research and Development Program of China (No. 2018AAA0100801).

Junsen Lu, Yun-Bo Zhao and Yu Kang are with Department of Automation, University of Science and Technology of China, Hefei 230026, China.

Yuhui Wang is with Department of Automation, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China.

Yimin Deng is with School of Automation Science and Electrical Engineering,Beihang University, Beijing, 100191, China.

* Corresponding author. ybzhao@ustc.edu.cn

To solve the mentioned problem, Lowe et al proposed the Deep Deterministic Policy Gradient (DDPG) algorithm [12]–[14]. In order to deal with the problem of continuous space, the DDPG algorithm learns a centralized Q function for each agent with the idea of centralized training, decentralized execution [15]. In this way, the DDPG algorithm can alleviate non-stationary problems and stabilize training based on global information. Several research studies have applied DDPG to the 1V1 UCAV air combat and have achieved good results.

To further reduce the training time and improve the stability of the training process, we here propose an improved DDPG algorithm using prioritized experience replay (PER) method [16], [17]. Based on the assumption that the RL agent can learn more knowledge from some transitions than other experiences, PER takes TD-error as the criterion to measure the value of each experience and "replay" higher value experiences more frequently. In this work, we propose an improved DDPG algorithm with prioritized experience replay to the air combat decision-making process, whose effectiveness is then verified by numerical examples.

The rest of the paper is organized as follows. In Section II, the air-combat game model is introduced with necessary definitions of the key concepts. In Section III, the DDPG algorithm with prioritized experience replay is discussed to generate air combat strategy. In section IV, several simulation experiments are considered to illustrate the effectiveness of the proposed method, and Section V concludes the paper.

II. PROBLEM FORMULATION

In this section, the 1v1 air combat game environment is introduced. The two players, referred to as red and blue, are assumed to maneuver in the same horizontal plane. The target of each player is to reach the tail of the adversary and track stably to satisfy the launching condition of the missiles. We first describe the system state, actions and the dynamics of UACV in the game. The goal reward function is then given according to Algorithm 1.

A. States

A 3-D environment is set up and the 1v1 air combat game is studied in the 3-D environment. As shown in Fig.1, the states of both UCAVs are defined by the following variables:

$$s = [q_r, q_b, d, \beta, \Delta h] \tag{1}$$

where the subscript r and b denote the parameters of our-side and the opponent, respectively, the line connecting two UAVs is the line of sight (LOS), d is the distance of LOS, q_b

is the antenna train angle (ATA) of blue, which is the angle between velocity vector of blue UCAV v_b and LOS, qr is the aspect angle (AA) of blue, which is the angle between LOS and velocity vector of red UCAV v_r, β is the angle between the heading directions of two UCAVs, and Δh is the height difference between two UCAVs.

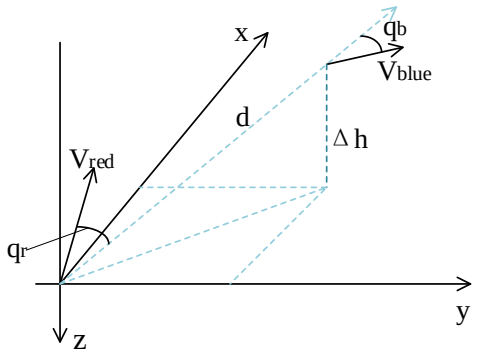

Fig. 1. The definition of states variables

The situation of both UCAVs are described by their position and velocity, and the variables can be calculated as follows,

$$\begin{cases} q_r = & \arccos[(x_b-x_r)\cos\psi_r\cos\tau_r+(y_b-y_r)\sin\psi_r \\ & \cos\tau_r+(z_b-z_r)\sin\tau_r]/d \\ q_b = & \arccos[(x_r-x_b)\cos\psi_r\cos\tau_r+(y_r-y_b)\sin\psi_r \\ & \cos\tau_r+(z_r-z_b)\sin\tau_r]/d \\ d = & \sqrt{(x_r-x_b)^2+(y_r-y_b)^2+(z_r-z_b)^2} \\ \beta = & \arccos(\cos\psi_r\cos\tau_r\cos\psi_b\cos\tau_b+ \\ & \cos\tau_r\sin\tau_r\cos\tau_b\sin\tau_b+\sin\tau_r\sin\tau_b) \end{cases} \tag{2}$$

where (x_r,y_r,z_r),(x_b,y_b,z_b) are the coordinates of two UCAVs, respectively. ψ_r,ψ_b represent the heading angle of two UCAVs respectively. τ_r, τ_b represents the heading angle of the two UCAVs, respectively. The position variables of the aircraft have no limits.

B. Dynamics

The dynamics of air craft can be simplified to a three-degree-of-freedom dynamic motion model. The UCAV dynamic model can be expressed as follows,

$$\begin{cases} \frac{dv}{dt} = g(n_x - sin\tau) \\ \frac{d\tau}{dt} = \frac{g}{v}(n_z cos\gamma_x - cos\tau) \\ \frac{d\psi}{dt} = \frac{g}{vcos\tau}(n_z sin\gamma_x) \end{cases} \tag{3}$$

where n_x is the tangential overload, which represents the tangential acceleration, n_z is the vertical overload, which represents the normal acceleration, γ_x is the roll angle, which represents rotation angle rotate along x axis. The motion model of UCAV can be expressed as follows,

$$\begin{cases} \frac{dx}{dt} = vcos\tau cos\psi \\ \frac{dy}{dt} = vcos\tau sin\psi \\ \frac{dz}{dt} = vsin\tau \end{cases} \tag{4}$$

C. Action

This paper uses continuous action space to control the aircrafts. Each UACV updates its state according to the dynamics and motion equations respectively after taking the actions. The acceleration of UCAV in each direction is n_x, n_z and γ_x, and these variables will lead to the changes of state information. Therefore, the control vector u of UCAV is defined as

$$u = [n_x, n_z, \gamma_x] \tag{5}$$

The different value of these three variables will lead the aircrafts to different actions,and each of them have upper bounds and lower limits respectively. The ranges of values are defined as: $n_x \in [-5,5]$, $n_z \in [-2,2]$,$\gamma_x \in [-\frac{\pi}{2},\frac{\pi}{2}]$.

D. Rewards

In a real air combat, the goal of the player is to reach and maintain an advantageous position behind the opponent, which is seen as a sufficient condition to launch a missile, and to obtain the highest reward. Therefore, we define a reward function for every state of the system to quantify the goal. The rules are made as follows: if the player's plane flies into the goal zone of the enemy, it will receive a positive reward. On the contrary, it will receive a negative reward if the opponent makes it to the goal zone. In summary, there could be three possible results in a time-limited episode: win, lose and out of the area. This paper considers 1vs1 air combat and the rewards of both aircrafts should be opposite. The rewards of each situation are expressed as in Algorithm 1,

Algorithm 1 Reward Function $R(s)$

Require: state s
Ensure: reward R
Caculate q_r,q_b,d,β,Δh
if $d < d_{min}$ **then**
  R=-10
else if $d_{min} < d < d_{max}$ **then**
  if $q_r < 30$ and $q_b > 30$ **then**
    R=10
  else if $q_r > 30$ and $q_b < 30$ **then**
    R=-10
  else
    R=0
  end if
else
  R=0
end if

III. DDPG with Prioritized Experience Replay

In this section, the method of DDPG and the idea of prioritized experience replay are introduced and the integrated algorithm of DDPG with prioritized experience replay is then discussed.

A. Deep Reinforcement Learning

In a standard reinforcement learning algorithm, the goal of the agent is to interact with the environment and maximize the long-term rewards. The process of this interaction is usually formulated as a Markov Decision Process (MDP). MDP can be described by a four-element tuple (S,A,R,P), where S is the state space, A is the action space, $R: S\times A\to R$, is the reward function and $P: S\times A\times A\to[0,1]$ is the transition probability. The long-term reward is defined as $R_0=\sum_{i=0}^{T}\gamma_i r(s_i,a_i)$ where $r(s_i,a_i)$ is the reward of executing action a_i at state a_i, T represents the termination time step and γ is the discount factor. The target of the agent is to learn a policy $\pi: S\to A$ to maximizes R_0. The action-value function is defined to represent the expected long-term reward of executing action a_t at state s_t,

$$Q(s_t,a_t)=E[R_t^{\gamma}|s=s_t,a=a_t]=E[\sum_{i=t}^{T}\gamma^{j-t}r(s_i,a_i)] \tag{6}$$

The optimal action-value function is often obtained by Bellman Equation,

$$Q^*(s_t,a_t)=E[r(s_t,a_t)+\gamma\max_{a_{t+1}}Q^*(s_{t+1},a_{t+1})] \tag{7}$$

However, the approach mentioned above can only deal with the situation where both the state space and the action space are discrete. In order to adopt the above algorithm in continuous space, deep neural networks are designed and implemented. DQN is then proposed combining the conventional reinforcement learning algorithms (e.g. Q-learning) with deep neural networks. DQN uses two networks, the action-value network $Q(s_t,a_t,\omega)$ is used to approximate the action-value function $Q(s_t,a_t)$ and the actor network $\mu(s_t,v)$ is used to approximate the actor function $\mu(s_t)$,where ω and v are the parameters of the network.

B. Deep Deterministic Policy Gradient

Based on the framework of the AC algorithm, the DDPG algorithm consists of an actor part and a critic part, and the two parts are treated by DQN accordingly. Besides, DDPG adopts a pair of neural networks with the same structure named evaluation neural network and target neural network for each part. The output of an actor part is a deterministic action whose network is defined as $a=\mu(s,v)$. Previous policy gradients used a random strategy, each acquisition action required sampling the distribution of the current optimal strategy, while DDPG used a deterministic strategy, determined directly by a function μ. In addition, Actor has a target network with the same structure but different parameters to update the value network, i.e. critic part. The outputs of both networks are action. The purpose of the critic part is to fit a value function $Q(s,a,\omega)$. There is also an estimated network and a target network. Both networks output the Q-value of the current state but have different inputs. The input of the critic target network has two parts, the observation value of the current state and the action output from the actor target network. The input of the critic estimated network is action output from the actor target network. For the training process of the neural network of the actor, the loss function is expressed by the average value of the state and action and the goal is to maximize the loss function:

$$Loss=-mean(v(s,a;\theta_{critic}) \tag{8}$$

For the critic, TD-error is used to update the parameters of the evaluation neural network and the goal of training process is to minimized the loss function,

$$\begin{aligned} &TD-error=r(s_t,a_t)+\gamma Q^{'}(s_{t+1},a_{t+1},\omega)-Q(s_t,a_t,\omega)\\ &Loss=(TD-error)^2 \end{aligned} \tag{9}$$

The structure of the DDPG algorithm is shown in Fig. 2.

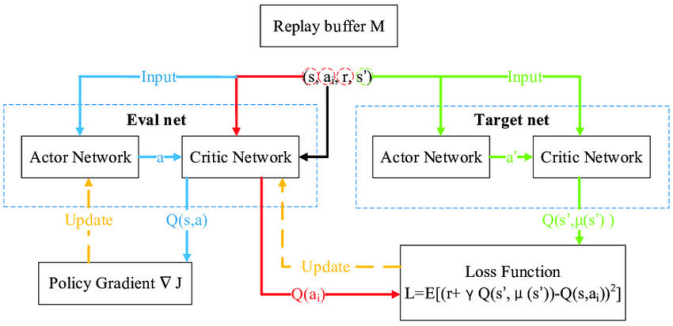

Fig. 2. The structure of DDPG algorithm

C. Prioritized Experience Replay

The core of priority experience playback is to design a criterion to measure the importance of each experience and to more frequently replay experiences with either very successful attempts or very bad attempts. A reasonable approximation of the criterion is the TD-error of transition since it describes how this experience is unexpected for critical networks. TD-error is the difference between the actual Q value and the estimated Q value, and is often used to update the estimation of the action-value function $Q(s,a)$. It reflects what extent an agent can learn from the experience. The bigger the absolute TD-error is, the more positive the correction for the expected action value is. The agent can avoid making wrong actions and improve the performance by replaying these experiences more frequently.

The probability of the sampled experience j entering the memory replay buffer is defined as

$$P(j)=\frac{D_j^{\alpha}}{\sum_k D_k^{\alpha}} \tag{10}$$

where $D_j=\frac{1}{\text{rank}(j)}>0$, $\text{rank}(j)$ is the rank of the experience j with absolute TD-error being the criterion. The parameter α is the adjustment factor. The experiences with a larger rank have a higher probability to enter the replay buffer. The use of prioritized experience replay prevents the over-fitting of the neural networks and leads to a better performance of the algorithm.

However, since we tend to replay the experience with high TD errors more frequently, it will undoubtedly change the

sample distribution and state access frequency. This change may cause the training process of neural network to oscillate or even diverge. To deal with this problem, the importance sampling weight is used when calculating the weight change:

$$W_j = \frac{1}{S^{\beta} \cdot P(j)^{\beta}} \tag{11}$$

where S is the size of the replay buffer, $P(j)$ is the probability of the sampled experience j, the parameter β controls to what extent you want to offset the effect of prioritized experience replay on the convergence results. If $\beta = 1$, it means the influence is completely cancelled out.

The integrated DDPG algorithm with prioritized experience replay is shown in Algorithm 2.

Algorithm 2 DDPG training process with Prioritized Experience Replay

Initialize action-value network $Q(s_t, a_t, \omega)$, actor network $\mu(s_t, v)$ with ω and v separately, replay buffer R with size S
Initialize target network $Q'(s_t, a_t, \omega)$, $\mu(s_t, v)$ with ω and v separately
Initialize maximum priority, parameters α, β updating rate of the target network λ, minibatch K
for episode = 1 to M **do**
  Initialize a random noise process N
  Observe initial state information s_0
  for $t = 1$ to max-episode-length **do**
    Add noise N in the exploration policy and select action a_i according to the new policy
    Execute actions $a_i = (a_1, a_2)$ and observe reward R and new state information; s_{t+1}
    Store experience (s_t, a_t, r_t, s_{t+1}) in replay buffer R
    set $s \leftarrow s_{t+1}$
    for agent $i = 1, 2$ **do**
      Sample experience j with probability $P(j)$
      Compute corresponding importance-sampling weight $W(j)$ and TD-error $\delta(j)$
      Update the priority of transition j according to absolute TD-error $|\delta_j|$
    end for
    Update critic network by minimizing the loss function $L = \frac{1}{K}\sum_i \omega_i \delta_i^2$
    Update actor network using the policy gradient $\nabla_v \mu|s_i = \frac{1}{K}\sum_i \nabla_a Q(s_t, a_t, \omega)|_{s=s_i, a=\mu(s_i,v)} \nabla_v \mu(s_t, v)|_{s_i}$
    Adjust the parameters of the target network $Q'(s_t, a_t, \omega)$ and $\mu'(s_t, v)$with an updating rate λ
  end for
end for

IV. Simulation and Analysis

In this section, experimental settings are first introduced and then the proposed algorithm is tested in simulations. Also, a comparison with the conventional DDPG algorithm is given.

TABLE I
The Initial States Of the Parameters Of the UCAVs

Parameter	Blue UCAV	Red UCAV
x	0	[-2000,2000] randomly
y	0	[-2000,2000] randomly
z	0	[-2000,2000] randomly
v	200	[100,300] randomly
ψ	0	0
τ	0	0

A. parameters setting

The proposed DDPG with PER algorithm is built by Tensorflow module. The details of the simulation are given below. Both actor and critic networks use a fully connected network with 64 units in hidden layer and the activation function is *tanh* function. The learning rate is set as 0.01, batch-size is set as 1024 and the discount factor γ is set as 0.95. The transition memory buffer size is 10000 and the exponent β in priority experience replay is set to 0.6. The initial positions of the UCAVs are shown in Table I.

B. training

We conduct two experiments, in which the strategy of adversary UCAV is generated by conventional DDPG algorithm and the strategy of our UCAV is generated by DDPG and the proposed DDPG with PER algorithm respectively. In each experiment, we simulate 10^6 episodes. We use the mean episode reward to evaluate the performance of the two algorithms and the results are shown in Fig. 3.

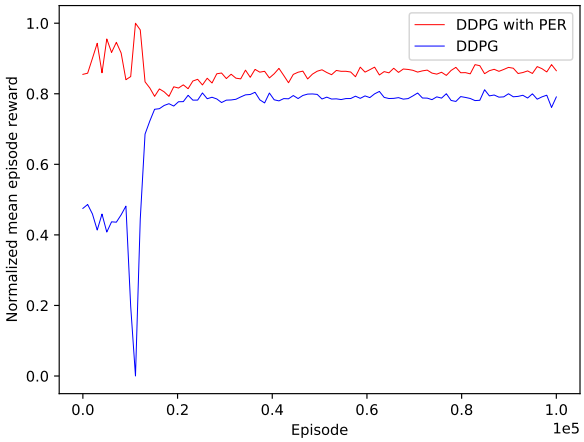

Fig. 3. The convergence process of the mean episode reward of our UCAV using DDPG and DDPG with PER

From Fig. 3, it can be seen that the mean episode rewards of both DDPG and DDPG with PER converge to a certain result and the DDPG with PER converges to a higher result than the conventional DDPG. It means that the introduce of prioritized experience replay can make the network performs better and lead us to advantage in a 1v1 air combat. Thus, the proposed algorithm is better than the conventional DDPG algorithm.

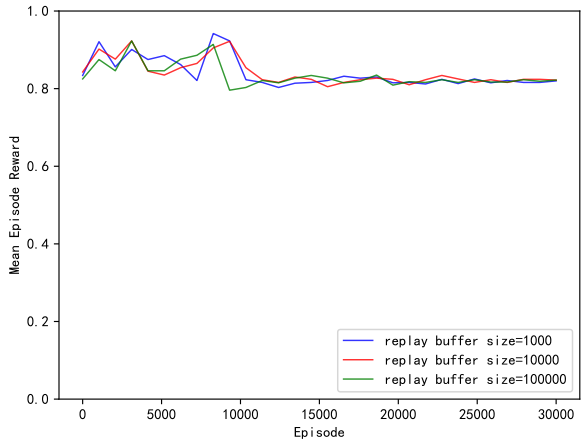

Fig. 4. Performance of PER based DDPG under different sizes of replay buffer

In addition, we change the value of the size of the replay buffer S and try to test whether DDPG with prioritized sampling is robust enough to be insensitive to changes. The size of the buffer is set as 10^4, 10^5 and 10^6, respectively. The results are shown in Fig.4.

From Fig.4, it can be seen that the convergence time and convergence value of the PER based DDPG almost remain the same as the size of the replay buffer changes. It shows that the proposed algorithm has a strong robustness to the change of the parameters.

C. testing and evaluation

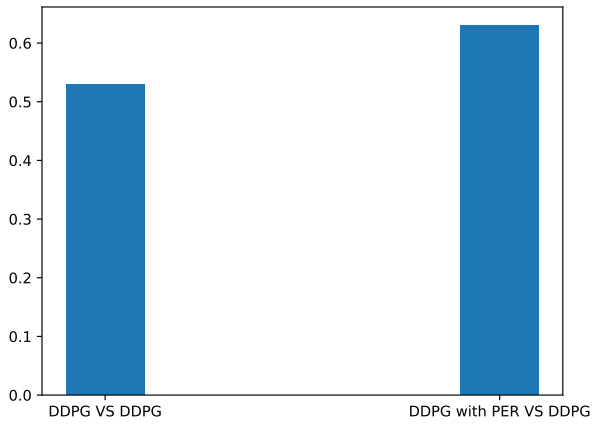

Fig. 5. Comparison between DDPG and DDPG with PER on winning rate of air combat

To further evaluate the quality of learned air combat maneuver strategy trained by the DDPG with PER, we use win rate to measure the performance of DDPG with PER. The adversary UCAV adopts DDPG and our UCAV adopts DDPG with PER and DDPG respectively. We conduct the experiments 100 times and the win rate under each condition is shown in Fig. 5.

From Fig.5, we can see the win rate of DDPG is among 0.5, while the win rate of DDPG with PER can reach 0.63 and is higher than that of DDPG. Thus, the algorithm proposed in this paper can perform better in an air combat than conventional DDPG algorithm.

V. Conclusions

A reinforcement learning based approach for UCAV maneuver strategy generation is proposed. The DDPG algorithm is improved by adopting the prioritized experience replay method. The proposed algorithm is simulated in a 1V1 UCAV air combat environment and the results show that, with the introduce of prioritized experience replay, the strategy generated improves greatly the autonomous capability of UCAV.

References

[1] F. U. Li, L. I. Wei, S. O. Automation, and S. A. University, "Analysis on the air combat strategies for fighterst," *Journal of Shenyang Aerospace University*, 2013.

[2] L. G. Dan and P. G. Gonsalves, "Agent-based simulation environment for ucav mission planning and execution," 2000.

[3] Y. Zhong, J. Liu, and G. Shen, "Recognition method for tactical maneuver of target in autonomous close-in air combat," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 33, no. 9, pp. 1056–1059, 2007.

[4] H. Shin, J. Lee, H. Kim, and D. H. Shim, "An autonomous aerial combat framework for two-on-two engagements based on basic fighter maneuvers," *Aerospace Science and Technology*, 2018.

[5] F. Austin, G. Carbone, H. Hinz, M. Lewis, and M. Falco, "Game theory for automated maneuvering during air-to-air combat," *Journal of Guidance Control and Dynamics*, vol. 13, no. 6, pp. 1143–1149, 1990.

[6] D. L. Luo, C. L. Shen, B. Wang, and W. H. Wu, "Air combat decision-making for cooperative multiple target attack using heuristic adaptive genetic algorithm," 2005.

[7] H. Duan, X. Wei, and Z. Dong, "Multiple ucavs cooperative air combat simulation platform based on pso, aco, and game theory," *IEEE Aerospace and Electronic Systems Magazine*, vol. 28, no. 11, pp. 12–19, 2013.

[8] N. Ernest, D. Carroll, C. Schumacher, M. Clark, and G. Lee, "Genetic fuzzy based artificial intelligence for unmanned combat aerialvehicle control in simulated air combat missions," *Journal of Defense Management*, vol. 06, no. 1, 2016.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *Computer ence*, 2015.

[10] Y. Ma, G. Gong, and X. Peng, "Cognition behavior model for air combat based on reinforcement learning," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 36, no. 4, pp. 379–383, 2010.

[11] P. Liu and Y. Ma, "A deep reinforcement learning based intelligent decision method for ucav air combat," in *Asian Simulation Conference*, 2017.

[12] A. Author and A. Address, "Deterministic policy gradient algorithms," 2002.

[13] W. Kong, D. Zhou, K. Zhang, and Z. Yang, "Air combat autonomous maneuver decision for one-on-one within visual range engagement base on robust multi-agent reinforcement learning," in *2020 IEEE 16th International Conference on Control and Automation (ICCA)*, 2020.

[14] Y. Ma, S. Bai, Y. Zhao, C. Song, and J. Yang, "Strategy generation based on reinforcement learning with deep deterministic policy gradient for ucav," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2020.

[15] D. Silver, G. Lever, N. Heess, T. Degris, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning*, 2014.

[16] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Transactions on Systems Man and Cybernetics Part C*, vol. 42, no. 2, pp. 201–212, 2012.

[17] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel ddpg method with prioritized experience replay," in *IEEE International Conference on Systems*, 2017.