Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 7180639, 17 pages
https://doi.org/10.1155/2020/7180639

Research Article

Improving Maneuver Strategy in Air Combat by Alternate Freeze Games with a Deep Reinforcement Learning Algorithm

Zhuang Wang[1], **Hui Li**[1,2], **Haolin Wu**[1], **and Zhaoxin Wu**[2]

[1]*College of Computer Science, Sichuan University, Chengdu, Sichuan 610065, China*
[2]*National Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University, Chengdu, Sichuan 610065, China*

Correspondence should be addressed to Hui Li; lihuib@scu.edu.cn

Received 11 April 2020; Accepted 9 June 2020; Published 30 June 2020

Academic Editor: Ramon Sancibrian

In a one-on-one air combat game, the opponent's maneuver strategy is usually not deterministic, which leads us to consider a variety of opponent's strategies when designing our maneuver strategy. In this paper, an alternate freeze game framework based on deep reinforcement learning is proposed to generate the maneuver strategy in an air combat pursuit. The maneuver strategy agents for aircraft guidance of both sides are designed in a flight level with fixed velocity and the one-on-one air combat scenario. Middleware which connects the agents and air combat simulation software is developed to provide a reinforcement learning environment for agent training. A reward shaping approach is used, by which the training speed is increased, and the performance of the generated trajectory is improved. Agents are trained by alternate freeze games with a deep reinforcement algorithm to deal with nonstationarity. A league system is adopted to avoid the *red queen* effect in the game where both sides implement adaptive strategies. Simulation results show that the proposed approach can be applied to maneuver guidance in air combat, and typical angle fight tactics can be learnt by the deep reinforcement learning agents. For the training of an opponent with the adaptive strategy, the winning rate can reach more than 50%, and the losing rate can be reduced to less than 15%. In a competition with all opponents, the winning rate of the strategic agent selected by the league system is more than 44%, and the probability of not losing is about 75%.

1. Introduction

Despite long-range radar and missile technology improvements, there is still a scenario that two fighter aircrafts may not detect each other until they are within the visual range. Therefore, modern fighters are designed for close combat, and military pilots are trained in air combat basic fighter maneuvering (BFM). Pursuit is a kind of BFM, which aims to control an aircraft to reach a position of advantage when it is fighting against another aircraft [1].

In order to reduce the workload of pilots and remove the need to provide them with complex spatial orientation information, many research studies focus on the autonomous air combat maneuver decision. Toubman et al. [2–5] used rule-based dynamic scripting in one-on-one, two-on-one, and two-on-two air combat, which requires hard coding the air-combat tactics into a maneuver selection algorithm. A virtual pursuit point-based combat maneuver guidance law for an unmanned combat aerial vehicle (UCAV) is presented and is used in X-Plane-based nonlinear six-degrees-of-freedom combat simulation [6, 7]. Eklund et al. [8, 9] presented a nonlinear, online model predictive controller for pursuit and evasion of two fixed-wing autonomous aircrafts, which rely on previous knowledge of the maneuvers.

Game-theoretic-based approaches are widely used in the automation of air combat pursuit maneuver. Austin et al. [10, 11] proposed a matrix game approach to generate intelligence maneuver decisions for one-on-one air combat. A limited search method is adopted over discrete maneuver choices to maximize a scoring function, and the feasibility of real-time autonomous combat is demonstrated in simulation. Ma et al. [12] formulated the cooperative occupancy decision-making problem in air combat as a zero-sum matrix game and designed the double-oracle combined

algorithm with neighborhood search to solve the model. In [13, 14], the air combat game is regarded as the Markov game, and then the pursuit maneuver strategy is solved by computing its Nash equilibrium. This approach solves the problem that the matrix game cannot deal with continuous multiple states. However, it is only suitable for rational opponents. An optimal pursuit-evasion fighter maneuver is formulated as a differential game and then solved by nonlinear programming, which is complex and requires enormous amount of calculation [15].

Other approaches for pursuit maneuver strategy generation include influence diagram, genetic algorithm, and approximate dynamic programming. Influence diagram is used to model the sequential maneuver decision in air combat, and high-performance simulation results are obtained [16–18]. However, this approach is difficult to be applied in practice since the influence diagram is converted into a nonlinear programming problem, which cannot meet the demand of fast computation during air combat. In [19], a genetics-based machine learning algorithm is implemented to generate high angle-of-attack air combat maneuver tactics for the X-31 fighter aircraft in a one-on-one air combat scenario. Approximate dynamic programming (ADP) can be employed to solve the air combat pursuit maneuver decision problem quickly and effectively [20, 21]. By controlling the roll rate, it can provide fast maneuver response in a changing situation.

Most of the previous studies have used various algorithms to solve the pursuit maneuver problem and have some satisfactory results, whereas two problems still exist. One is that previous studies have assumed the maneuver strategy of the opponent is deterministic or generated by a fixed algorithm. However, in realistic situations, these approaches are difficult to deal with the flexible strategies adopted by different opponents. The other is that traditional algorithms rely heavily on prior knowledge and have high computational complexity, which cannot adapt to the rapidly changing situation in air combat. Since UCAVs have received growing interest worldwide and the flight control system of the aircraft is developing rapidly towards intellectualization [22], it is necessary to do research on the intelligent maneuver strategy in UCAVs and manned aircraft combat.

In this paper, a deep reinforcement learning- (DRL-) [23] based alternate freeze game approach is proposed to train guidance agents which can provide maneuver instructions in an air combat. Using alternate freeze games, in each training period, one agent is learning while its opponent is frozen. DRL is a type of artificial intelligence, which combines reinforcement learning (RL) [24] and deep learning (DL) [25]. RL allows an agent to learn directly from the environment through trial and error without perfect knowledge of the environment in advance. A well-trained DRL agent can automatically determine an adequate behavior within a specific context trying to maximize its performance using few computational times. The theory of DRL is very suitable for solving sequential decision-making problems such as maneuver guidance in air combat.

DRL has been utilized in many decision-making fields, such as video games [26, 27], board games [28, 29], and robot control [30] and obtained great achievements of human level or superhuman performance. For aircraft guidance research, Waldock et al. [31] proposed a DQN method to generate a trajectory to perform perched landing on the ground. Alejandro et al. [32] proposed a DRL strategy for autonomous landing of the UAV on a moving platform. Lou and Guo [33] presented a uniform framework of adaptive control by using policy-searching algorithms for a quadrotor. An RL agent is developed for guiding a powered UAV from one thermal location to another by controlling bank angle [34]. For air combat research, You et al. [35] developed an innovative framework for cognitive electronic warfare tasks by using a DRL algorithm without prior information. Luo et al. [36] proposed a Q-learning-based air combat target assignment algorithm which avoids relying on prior knowledge and performs well.

Reward shaping is a method of incorporating domain knowledge into RL so that the algorithms are guided faster towards more promising solutions [37]. Reward shaping is widely adopted in the RL community, and it is also used in aircraft planning and control. In [4], a reward function is proposed to remedy the false rewards and punishments for firing air combat missiles, which allows computer-generated forces (CGFs) to generate more intelligent behavior. Tumer and Agogino [38] proposed difference reward functions in a multiagent air traffic system and showed that agents can manage effective route selection and significantly reduce congestion. In [39], two types of reward functions are developed to solve ground holding and air holding problems, which assist air traffic controllers in maintaining high standard of safety and fairness between airlines.

Previous research studies have shown benefits of using DRL to solve the air maneuver guidance problem. However, there are still two problems in applying DRL to air combat. One is that previous studies did not have a specific environment for air combat, either developing an environment based on the universal ones [32] or using a discrete grid world, which do not have the function of air combat simulation.

The other problem is that classic DRL algorithms are almost all one-sided optimization algorithms, which can only guide an aircraft to a fixed location or a regularly moving destination. However, the opponent in air combat has diversity and variability maneuver strategies, which are nonstationary, and the classic DRL algorithms cannot deal with them. Hernandez-Leal et al. reviewed how the nonstationarity is modelled and addressed by state-of-the-art multiagent learning algorithms [40]. Some researchers combine MDP with game theory to study reinforcement learning in stochastic games to solve the nonstationarity problem [41–43], while there are still two limitations. One is to assume the opponent's strategy is rational. In each training step, the opponent chooses the optimal action. A trained agent can only deal with a rational opponent, but cannot fight against a nonrational one. The other is to use linear programming or quadratic programming to calculate the Nash equilibrium, which leads to a huge amount of

calculations. The Minimax-DQN [44] algorithm which combines DQN and Minimax-Q learning [41] for the two-player zero-sum Markov game is proposed in a recent paper. Although it can be applied to complex games, it still can only deal with rational opponents and needs to use linear programming to calculate the Q value.

The main contributions of this paper can be summarized as follows:

(i) Middleware is designed and developed, which makes specific software for air combat simulation used as an RL environment. An agent can be trained in this environment and then used to guide an aircraft to reach the position of advantage in one-on-one air combat.

(ii) Guidance agents for air combat pursuit maneuver of both sides are designed. The reward shaping method is adopted to improve the convergence speed of training and the performance of the maneuver guidance strategies. An agent is trained in the environment where its opponent is also an adaptive agent so that the well-trained agent has the ability to fight against an opponent with the intelligent strategy.

(iii) An alternate freeze game framework is proposed to deal with nonstationarity, which can be used to solve the problem of variable opponent strategies in RL. The league system is adopted to select an agent with the highest performance to avoid the *red queen* effect [45].

This paper is organized as follows. Section 2 introduces the maneuver guidance strategy problem of one-on-one air combat maneuver under consideration and the DRL-based model as well as training environment to solve it. The training optimization includes reward shaping and alternate freeze games which are presented in Section 3. The simulation and results of the proposed approach are given in Section 4. The final section concludes the paper.

2. Problem Formulation

In this section, the problem of maneuver guidance in air combat is introduced. The training environment is designed, and the DRL-based model is set up to solve it.

2.1. Problem Statement. The problem solved in this paper is a one-on-one air combat pursuit-evasion maneuver problem. The red aircraft is regarded as our side and the blue one as an enemy. The objective of the guidance agent is to learn a maneuver strategy (policy) to guide the aircraft from its current position to a position of advantage and maintain the advantage. At the same time, it should guide the aircraft not to enter the advantage position of its opponent.

The dynamics of the aircraft is described by a point-mass model [17] and is given by the following differential equations:

$$\begin{cases} \dot{x} = v\cos\theta\cos\psi, \\ \dot{y} = v\cos\theta\sin\psi, \\ \dot{z} = v\sin\theta, \\ \dot{\theta} = \dfrac{1}{mv}\left((L + T\sin\beta) \times \cos\phi - mg\cos\theta\right), \\ \dot{\psi} = \dfrac{1}{mv\cos\theta}(L + T\sin\beta) \times \sin\phi, \\ \dot{v} = \dfrac{1}{m}(T\cos\beta - D) - g\sin\theta, \end{cases} \tag{1}$$

where (x, y, z) are 3-dimensional coordinates of the aircraft. The terms v, θ, and ψ are speed, flight path angle, and the heading angle of the aircraft, respectively. The mass of the aircraft and the acceleration due to the gravity are denoted by m and g, respectively. The three forces are lift force L, drag force D, and thrust force T. The remaining two variables are the angle of attack β and the bank angle ϕ, as shown in Figure 1.

In this paper, the aircraft is assumed to fly at a fixed velocity in the horizontal plane, and the assumption can be written as follows:

$$\begin{cases} \beta = 0, \\ \theta = 0, \\ T = D, \\ L = \dfrac{mg}{\cos\phi}. \end{cases} \tag{2}$$

The equations of motion for the aircraft are simplified as follows:

$$\begin{cases} x_{t+\delta t} = x_t + v\delta t\cos\left(\psi_{t+\delta t}\right), \\ y_{t+\delta t} = y_t + v\delta t\sin\left(\psi_{t+\delta t}\right), \\ \psi_{t+\delta t} = \psi_t + \dot{\psi}_{t+\delta t}\delta t, \\ \dot{\psi}_{t+\delta t} = \dfrac{g}{v}\tan\left(\phi_{t+\delta t}\right), \\ \phi_{t+\delta t} = \phi_t + A_t\dot{\phi}_t\delta t, \end{cases} \tag{3}$$

where v, $\dot{\phi}_t$, ϕ_t, $\dot{\psi}_t$, and ψ_t are the speed, roll rate, bank angle, turn rate, and heading angle of the aircraft at time t, respectively. The term A_t is the action generated by the guidance agent, and the control actions available to the aircraft are $a \in \{\text{roll-left, maintain-bank-angle, roll-right}\}$. The simulation time step is denoted by δt.

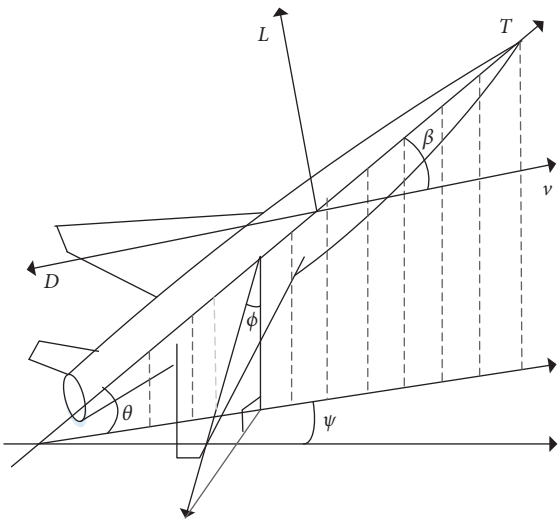

Figure 1: Aircraft dynamics parameters.

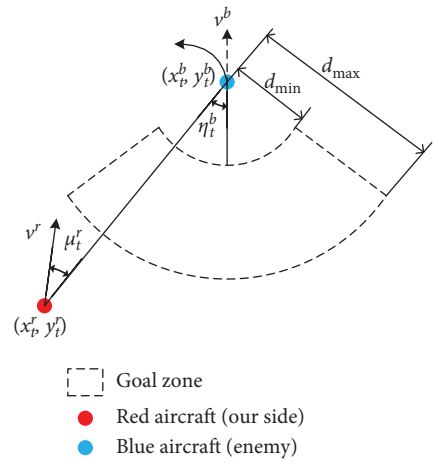

Figure 2: Aircraft maneuver guidance problem in air combat.

In aircraft maneuver strategy design, maneuvers in a fixed plane are usually used to measure its performance. Figure 2 shows a one-on-one air combat scenario [20], in which each aircraft flies at a fixed velocity in the X-Y plane under a maneuver strategy at time t. The position of advantage is that one aircraft gains opportunities to fire at its opponent. It is defined as

$$\begin{cases} d_{\min} < d_t < d_{\max}, \\ \left|\mu_t^r\right| < \mu_{\max}, \\ \left|\eta_t^b\right| < \eta_{\max}, \end{cases} \tag{4}$$

where superscripts r and b refer to red and blue, respectively. The term d_t is the distance between the aircraft and the target at time t, μ_t^r is the deviation angle which is defined as the angle difference between the line-of-sight vector and the heading of our aircraft, and η_t^b is the aspect angle which is between the longitudinal symmetry axis (to the tail direction) of the target plane and the connecting line from the target plane's tail to attacking the plane's nose. The terms $d_{\min}$, $d_{\max}$, $\mu_{\max}$, and $\mu_{\min}$ are thresholds, where the subscripts *max* and *min* represent the upper and lower bounds of the corresponding variables, respectively, which are determined by the combat mission and the performance of the aircraft.

The expression $d_{\min} < d_t < d_{\max}$ makes sure that the opponent is within the attacking range of the air-to-air weapon of the aircraft. The expression $|\mu_t^r| < \mu_{\max}$ refers to an area in which the blue aircraft is difficult to escape with sensor locking. The expression $|\eta_t^b| < \eta_{\max}$ defines an area where the killing probability is high when attacking from the rear of the blue aircraft.

2.2. DRL-Based Model. An RL agent interacts with an environment over time and aims to maximize the long-term reward [24]. At each training time t, the agent receives a state S_t in a state space S and generates an action A_t from an action space A following a policy π: $S \times A \longrightarrow \mathbb{R}$. Then, the agent receives a scalar reward R_t and transitions to the next state S_{t+1} according to the environment dynamics, as shown in Figure 3.

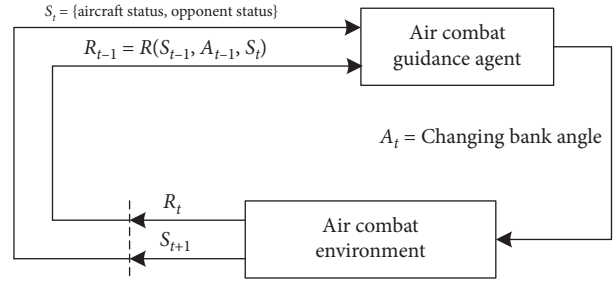

Figure 3: Reinforcement learning in air combat.

A value function $V(s)$ is defined to evaluate the air combat advantages of each state, which is the expectation of discounted cumulated rewards on all states following time t:

$$V(s) = E_\pi\left\{R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots \mid S_t = s\right\}, \tag{5}$$

where $\gamma \in [0, 1]$ is the discount factor and policy $\pi(s, a)$ is a mapping from the state space to the action space.

The agent guides an aircraft to maneuver by changing its bank angle ϕ using roll rate $\dot{\phi}$. The action space of DRL is $\{-1, 0, 1\}$, and A_t can take a value from three options at time t, which means roll-left, maintain-bank-angle, and roll-right, respectively. The position of the aircraft is updated according to (3). The action-value function $Q(s, a)$ can be used, which is defined as the value of taking action a in state s under a policy π:

$$Q(s, a) = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k R_{t+k} \mid S_t = s, A_t = a\right\}. \tag{6}$$

The Bellman optimality equation is

$$Q^*(s, a) = E\left\{R_t + \gamma \max_{a'} Q^*(S_{t+1}, a') S_t = s, A_t = a\right\}. \tag{7}$$

Any policy that is greedy with respect to the optimal evaluation function $Q^*(s, a)$ is an optimal policy. Actually, $Q^*(s, a)$ can be obtained through iterations using temporal-difference learning, and its updated formula is defined as

$$Q(S_t, A_t) \longleftarrow Q(S_t, A_t) + \alpha\Big(R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big), \tag{8}$$

where $Q(S_t, A_t)$ is the estimated action-value function and α is the learning rate.

In air combat maneuvering applications, the reward is sparse, and only at the end of a game (the terminal state), according to the results of winning, losing, or drawing, a reward value is given. This makes learning algorithms to use delayed feedback to determine the long-term consequences of their actions in all nonterminal states, which is time consuming. Reward shaping is proposed by Ng et al. [46] to introduce additional rewards into the learning process, which will guide the algorithm towards learning a good policy faster. The shaping reward function $F(S_t, A_t, S_{t+1})$ has the form

$$F(S_t, A_t, S_{t+1}) = \gamma\Phi(S_{t+1}) - \Phi(S_t), \tag{9}$$

where $\Phi(S_t)$ is a real-valued function over states. It can be proven that the final policy after using reward shaping is equivalent to the final policy without it [46]. R_t in previous equations can be replaced by $R_t + F(S_t, A_t, S_{t+1})$.

Taking the red agent as an example, the state space of one guidance agent can be described with a vector:

$$S_t = \left\{x_t^r, y_t^r, \psi_t^r, \phi_t^r, x_t^b, y_t^b, \psi_t^b\right\}. \tag{10}$$

The state that an agent received in one training step includes the position, heading angle, and bank angle of itself and the position, heading angle, and bank angle of its opponent, which is a 7-dimensional infinite vector. Therefore, the function approximate method should be used to combine features of state and learned weights. DRL is a solution to address this problem. Great achievements have been made using the advanced DRL algorithms, such as DQN [26], DDPG [47], A3C [48], and PPO [49]. Since the action of aircraft guidance is discrete, DQN algorithm can be used. The action-value function can be estimated with function approximation such as $\widehat{Q}(s, a, \mathbf{w}) \approx Q_\pi(s, a)$, where $\mathbf{w}$ are the weights of a neural network.

DQN algorithm is executed according to the training time step. However, there is not only training time step but also simulation time step. Because of the inconsistency, DQN algorithm needs to be improved to train a guidance agent. The pseudo-code of DQN for aircraft maneuver guidance training in air combat is shown in Algorithm 1. At each training step, the agent receives information about the aircraft and its opponent and then generates an action and sends it to the environment. After receiving the action, the aircraft in the environment is maneuvered according to this action in each simulation step.

2.3. Middleware Design. An RL agent improves its ability through continuous interaction with the environment. However, there is no maneuver guidance environment for air combat, and some combat simulation software can only execute agents, not train them. In this paper, middleware is designed and developed based on commercial air combat simulation software [50] coded in C++. The functions of middleware include interface between software and RL agents, coordination of the simulation time step and the RL training time step, and reward calculation. By using it, an RL environment for aircraft maneuver guidance training in air combat is performed. The guidance agents of both aircrafts have the same structure. Figure 4 shows the structure and the training process of the red side.

In actual or simulated air combat, an aircraft perceives the situation through multisensors. By analyzing the situation after information fusion, maneuver decisions are made by the pilot or the auxiliary decision-making system. In this system, situation perception and decision-making are the same as in actual air combat. An assumption is made that a sensor with full situation perception ability is used, through which an aircraft can obtain the position and heading angle of its opponent.

Maneuver guidance in air combat using RL is an episodic task, and there is the notion of episodes of some length, where the goal is to take the agent from a starting state to a goal state [51]. First, a flight level with fixed velocity and the one-on-one air combat scenario is setup in simulation software. Then, the situation information of both aircrafts is randomly initialized, including their positions, heading angles, bank angles, and roll rates, which constitute the starting state of each agent. The goal states are achieved when one aircraft reaches the position of advantage and maintains it, one aircraft flies out of the sector, or the simulation time is out.

Middleware can be used in the training and application phases of agents. The training phase is the process of making the agent from zero to have the maneuver guidance ability, which includes training episodes and testing episodes. In the training episode, the agent is trained by the DRL algorithm, and its guidance strategy is continuously improved. After a number of training episodes, some testing episodes are performed to verify the ability of the agent in the current stage, in which the strategy does not change. The application phase is the process of using the well-trained agent to maneuver guidance in air combat simulation.

A training episode is composed of training steps. In each training step, first, the information recognized by the airborne sensor is sent to the guidance agent through middleware as a tuple of state coded in Python, see Steps 1, 2, and 3 in Figure 4. Then, the agent generates an action using its neural networks according to the exploration strategy, as shown in Steps 4 and 5. Simulation software receives a guidance instruction transformed by middleware and sends the next situation information to middleware, see Steps 6 and 7. Next, middleware transforms the situation information into state information and calculates reward and sends them to the agent, as shown in Steps 2, 3, and 8. Because agent training is an iterative process, all the steps except Step 1 are executed repeatedly in an episode. Last, the tuple of the current state, action, reward, and the next state in each step is stored in the replay memory for the training of the guidance agent, see Steps 9 and 10. In each simulation step, the aircraft maneuvers according to the instruction and recognizes the situation according to its sensor configuration.

(1) Set parameters of the controlled aircraft and opponent
(2) Set training time step size Δt and simulation time step size δt
(3) Initialize maximum number of training episode M and maximum simulation time T in each episode
(4) Initialize replay memory D to capacity N
(5) Initialize action-value function Q and target action-value function Q^- with random weights
(6) **for** episode = 0, M **do**
(7) Initialize state S_0
(8) **for** $t = 0, T/\Delta t$ **do**
(9) With probability ϵ select a random action A_t
(10) Otherwise select $A_t = \arg\max_a Q(S_t, a, \mathbf{w})$
(11) **for** $i = 0, \Delta t/\delta t$ **do**
(12) Execute action A_t in air combat simulation software
(13) Obtain the positions of aircraft and target
(14) **if** episode terminates **then**
(15) break
(16) **end if**
(17) **end for**
(18) Observe reward R_t and state S_{t+1}
(19) Store transition $[S_t, A_t, R_t, S_{t+1}]$ in D
(20) **if** episode terminates **then**
(21) break
(22) **end if**
(23) Sample random minibatch of transitions $[S_j, A_j, R_j, S_{j+1}]$ from D
(24) **if** episode terminates at step $j + 1$ **then**
(25) set $Y_j = R_j$
(26) **else**
(27) set $Y_j = R_j + \max_a Q^-(S_{j+1}, a, \mathbf{w}^-)$
(28) **end if**
(29) Perform a gradient descent step on $(Y_j - Q(S_j, A_j, \mathbf{w_j}))^2$ with respect to the network parameters $\mathbf{w}$
(30) Every C steps reset $Q^- = Q$
(31) **end for**
(32) **end for**

Algorithm 1: DQN [26] for maneuver guidance agent training.

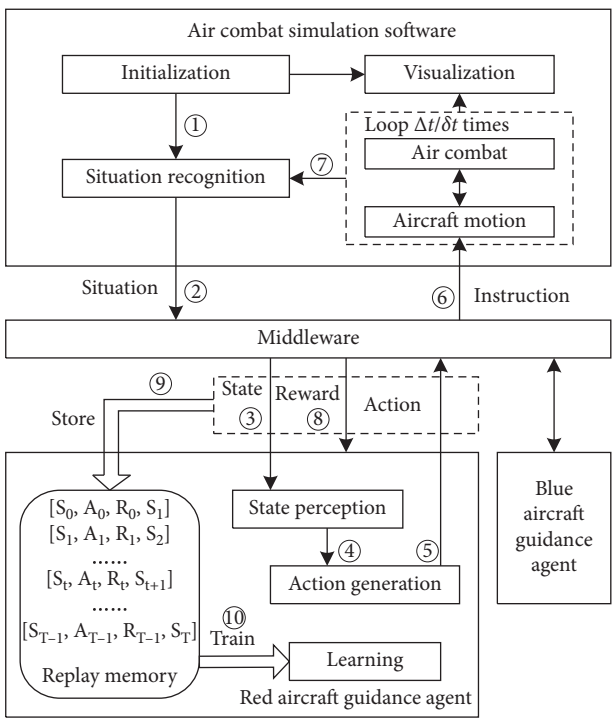

Figure 4: Structure and training process of the red side.

3. Training Optimization

3.1. Reward Shaping. Two problems need to be solved when using the DRL method to train a maneuver guidance agent in air combat. One is that the only criterion to evaluate the guidance is the successful arrival to the position of advantage, which is a sparse reward problem, leading to slow convergence of training. Secondly, in realistic situations, the time to arrive at the position of advantage and the quality of the trajectory need to be considered. In this paper, a reward shaping method is proposed to improve the training speed and the performance of the maneuver guidance strategy.

Reward shaping [46] is usually used to modify the reward function to facilitate learning while maintaining optimal policy, which is a manual endeavour. In this study, there are four rules to follow in reward shaping:

(i) Limited by the combat area and the maximum number of control times, the aircraft is guided to the position of advantage

(ii) The aircraft should be guided closer and closer to its goal zone

(iii) The aircraft should be guided away from the goal zone of its opponent

(iv) The aircraft should be guided to its goal zone in short time as much as possible

According to the above rules, the reward function is defined as

$$R(S_t, A_t, S_{t+1}) = R_t + F(S_t, A_t, S_{t+1}), \tag{11}$$

where R_t is the original scalar reward and $F(S_t, A_t, S_{t+1})$ is the shaping reward function. R_t is given by

$$R_t = w_1 T(S_{t+1}) + w_2, \tag{12}$$

where $T(S_{t+1})$ is the termination reward function. The term w_1 is a coefficient, and w_2 is a penalty for time consumption, which is a constant.

There are four kinds of termination states: the aircraft arrives at the position of advantage; the opponent arrives at its position of advantage; the aircraft moves out of the combat area; and the maximum number of control times has been reached, and each aircraft is still in the combat area and has not reached its advantage situation. Termination reward is the reward obtained when the next state is termination, which is often used in the standard RL training. It is defined as

$$T(S_{t+1}) = \begin{cases} c_1, \text{if win}, \\ c_2, \text{else if loss}, \\ c_3, \text{else if out of the sector}, \\ c_4, \text{else if no control times left}, \\ 0, \text{else}. \end{cases} \tag{13}$$

Usually, c_1 is a positive value; c_2 is a negative value; and c_3 and c_4 are nonpositive.

The shaping reward function $F(S_t, A_t, S_{t+1})$ has the form as described in (9). The real-value function $\Phi(S_t)$ is an advantage function of the state. The larger the value of this function, the more advantageous the current situation is. This function can provide additional information to help the agent to select action, which is better than only using the termination reward. It is defined as

$$\Phi(S_t) = \frac{D(S_t)O(S_t)}{100}, \tag{14}$$

where $D(S_t)$ is the distance reward function and $O(S_t)$ is the orientation reward function, which are defined as

$$D(S_t) = exp\left(\frac{-\left|d_t - (d_{\max} + d_{\min})/2\right|}{180^\circ k}\right), \tag{15}$$

$$O(S_t) = 1 - \frac{\left|\mu_t^r\right| + \left|\eta_t^b\right|}{180^\circ}, \tag{16}$$

where k has units of meters/degrees and is used to adjust the relative effect of range and angle, and a value of 10 is effective.

3.2. Alternate Freeze Game DQN for Maneuver Guidance Agent Training. Unlike board and RTS games, the data of human players in air combat games are very rare, so pretraining methods using supervised learning algorithms cannot be used. We can only assume a strategy used by the opponent and then propose a strategy to defeat it. We then think about what strategies the opponent will use to confront our strategy and then optimize our strategy.

In this paper, alternate freeze learning is used for games to solve the nonstationarity problem. Both aircrafts in a one-on-one air combat scenario use DRL-based agents to adapt their maneuver strategies. In each training period, one agent is learning from scratch, while the strategy of its opponent, which was obtained from the previous training period, is frozen. Through games, the maneuver guidance performance of each side rises alternately, and different agents with high level of maneuver decision-making ability are generated. The pseudo-code of alternate freeze game DQN is shown in Algorithm 2.

An important effect that must be considered in this approach is the *red queen* effect. When one aircraft uses a DRL agent and its opponent uses a static strategy, the performance of the aircraft is absolute with respect to its opponent. However, when both aircrafts use DRL agents, the performance of each agent is only related to its current opponent. As a result, the trained agent is only suitable for its latest opponent, but cannot deal with the previous ones.

Through K training periods of alternate freeze learning, K red agents and $K + 1$ blue agents are saved in the training process. The league system is adopted, in which those agents are regarded as players. By using a combination of strategies from multiple opponents, the opponent's mixed strategy becomes smoother. In different initial scenarios, each agent guides the aircraft to confront the aircraft guided by each opponent. The goal is to select one of our strategies through the league so that the probability of winning is high, and the probability of losing is low. According to the result of the competition, the agent obtains different points. The top agent in the league table is selected as the optimum one.

4. Simulation and Results

In this section, first, random air combat scenarios are initialized in simulation software for maneuver guidance agent training and testing. Second, agents of both sides with reward shaping are created and trained using the alternate freeze game DQN algorithm. Third, taking the well-trained agents as players, the league system is adopted, and the top agent in the league table is selected. Last, the selected agent is evaluated, and the maneuver behavior of the aircraft is analyzed.

4.1. Simulation Setup. The air combat simulation parameters are shown in Table 1. The Adam optimizer [52] is used for learning the neural network parameters with a learning rate of 5×10^{-4} and a discount $\gamma = 0.99$. The replay memory size is 1×10^5. The initial value of ϵ in ϵ-greedy exploration is 1, and the final value is 0.1. For each simulation, the reward shaping parameters c_1, c_2, c_3, and c_4 are set to 2, −2, −1, and −1, respectively. The terms w_1 and w_2 are set to 0.98, and 0.01, respectively. The training period K is 10.

(1) Set parameters of both aircrafts
(2) Set simulation parameters
(3) Set the number of training periods K and the condition for ending each training period $W_{\text{threshold}}$
(4) Set DRL parameters
(5) Set the opponent initialization policy π_0^{blue}
(6) **for** period = 1, K **do**
(7) **for** aircraft = [red, blue] **do**
(8) **if** aircraft = red **then**
(9) Set the opponent policy $\pi = \pi_{\text{period}-1}^{\text{blue}}$
(10) Initialize neural networks of red agent
(11) **while** Winning rate< $W_{\text{threshold}}$ **do**
(12) Train agent using Algorithm 1
(13) **end while**
(14) Save the well-trained agent, whose maneuver guidance policy is $\pi_{\text{period}}^{\text{red}}$
(15) **else**
(16) **if** period = K + 1 **then**
(17) break
(18) **else**
(19) Set the opponent policy $\pi = \pi_{\text{period}}^{\text{red}}$
(20) Initialize neural networks of blue agent
(21) **while** Winning rate< $W_{\text{threshold}}$ **do**
(22) Train agent using Algorithm 1
(23) **end while**
(24) Save the well-trained agent, whose maneuver guidance policy is $\pi_{\text{period}}^{\text{blue}}$
(25) **end if**
(26) **end if**
(27) **end for**
(28) **end for**

Algorithm 2: Alternate freeze game DQN for maneuver guidance agent training in air combats.

Table 1: Air combat simulation parameters.

Red aircraft parameters	
Velocity	200 m/s
Maximum roll angle	75°
Roll rate	40° s
Initial position	Random position and heading angle
Blue aircraft parameters	
Velocity	200 m/s
Maximum roll angle	75°
Roll rate	40° s
Initial position	Map center, northward
Initial policy (π_0^b)	Maximum overload turn
Simulation parameters	
Position of advantage	$d_{\max}$: 500 m; $d_{\min}$: 100 m; $\mu_{\max}$: 60°; $\eta_{\max}$: 30°
Combat area	5 km × 5 km
Simulation time T	500 s
Simulation time step δt	0.1 s
Training time step Δt	0.5 s

We evaluated three neural networks and three minibatches. Each neural network has 3 hidden layers, and the number of units is 64, 64, 128 for the small neural network (NN), 128, 128, 256 for the medium one, and 256, 256, 512 for the large one. The sizes of minibatch (MB) are 256, 512, and 1024, respectively. These neural networks and minibatches are used in pairs to train red agents and play against the initial blue agent. The simulation result is shown in Figure 5. First, a large minibatch cannot make the training successful for small and medium neural networks, and its training speed is slow for a large network. Second, using a small neural network, the average reward obtained will be lower than that of the other two networks. Third, if the same neural network is adopted, the training speed using the medium minibatch is faster than using a small one. Last, considering the computational efficiency, the selected network has 3 hidden layers with 128, 128, and 256 units, respectively. The selected minibatch size is 512.

In order to improve the generality of a single agent, initial positions are randomized in air combat simulation. In the first iteration of the training process, the initial strategy is formed by a randomly initialized neural network for the red agent, and the conservative strategy of the maximum-overload turn is adopted by the blue one. In the subsequent training, the training agent is trained from scratch, and its opponent adopts the strategy obtained from the latest training.

Air combat geometry can be divided into four categories (from red aircraft's perspective): offensive, defensive, neutral, and head-on [7], as shown in Figure 6(a). Defining the deviation angle and aspect angle from −180° to 180°, Figure 6(b) shows an advantage diagram where the distance is 300 m. Smaller $|\mu^r|$ means that the heading or gun of the red aircraft has better aim at its opponent, and smaller $|\eta^b|$ implies a higher possibility of the blue aircraft being shot or facing a fatal situation. For example, in the offensive

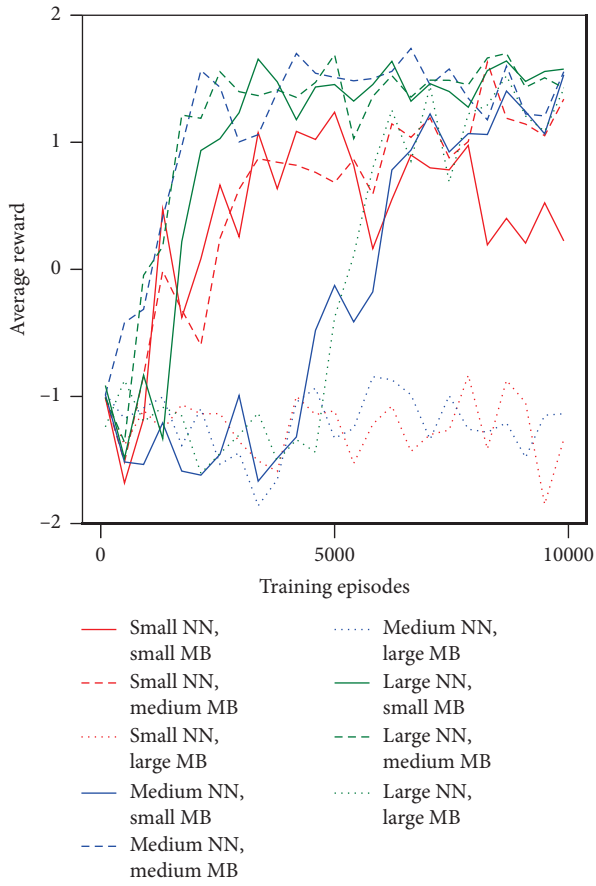

Figure 5: Evaluation of the neural network and minibatch.

scenario, the initial state value of the red agent is large because of small $|\mu^r|$ and $|\eta^b|$, according to equations (14) and (16). Therefore, in the offensive initial scenario, the win probability of our side will be larger than that of the opponent, while the defensive initial scenario is reversed. In neutral and head-one initial scenarios, the initial state values of both sides are almost equal. The performance of the well-trained agents is verified according to the classification of four initial situations in the league system.

4.2. Simulation Results. The policy naming convention is π_i^b or π_i^r, which means a *blue* or *red* policy produced after i periods, respectively. After every 100 training episodes, the test episodes run 100 times, and the learning curves are shown in Figure 7. For the first period (π_1^r vs. π_0^b), due to the simple maneuver policy of the blue aircraft, the red aircraft can achieve almost 100% success rate through about 4000 episodes of training, as shown in Figure 7(a). At the beginning, most of the games are draw because of the random initialization of the red agent and the evasion strategy of its opponent. Another reason is that the aircraft often flies out of the airspace in the early stages of training. Although the agent will get a penalty for this, the air combat episode will still get a draw. During the training process, the red aircraft is looking for winning strategies in the airspace, and its winning rate is constantly increasing. However, because of its pursuit of offensive and neglect of defensive, the winning rate of its opponent is also rising. In the later stage of training, the red aircraft gradually understands the opponent's strategy, and it can achieve a very high winning rate.

With the iterations in games, both agents have learnt the intelligent maneuver strategy, which can guide the aircraft in the pursuit-evasion game. Figure 7(b) shows a typical example that π_5^b is the trainer and π_5^r is its opponent. After about 5,000 episodes of training, the blue aircraft can reduce the losing rate to a lower level. Since the red one adopts an intelligent maneuver guidance strategy, the blue agent cannot learn a winning strategy in a short time. After about 20,000 episodes, the agent gradually understands the opponent's maneuver strategy, and its winning rate keeps increasing. The final winning rate is stable between 50% and 60%, and the losing rate is below 10%.

The training process iteration of π_{10}^r vs. π_9^b is shown in Figure 7(c). Because the maneuver strategy of the blue aircraft is an intelligent strategy which has been trained iteratively, training for the red agent is much more difficult than that in the first iteration. The well-trained agent can win more than 60% and lose less than 10% in the game against π_9^b.

It is hard to get a higher winning rate in every training except for the scenario that the opponent is π_0^b. This is because both aircrafts are homogeneous, the initial situations are randomized, and the opponent in the training environment is intelligent. In some situations, as long as the opponent's strategy is intelligent enough, the trained agent is almost impossible to win. Interestingly, in some scenarios where the opponent is intelligent enough and the aircraft cannot defeat it no matter how the agent operates, the agent will guide the aircraft out of the airspace to obtain a draw. This gives us an inspiration, that is, in a scenario where you cannot win, it may be the best way to get out of the combat area.

For four initial air combat situations, two aircrafts are guided by the agents trained in each iteration stage. The typical flight trajectories are shown in Figure 8. Since π_0^b is a static strategy, π_1^r can easily win in offensive, head-on, and neutral situations, as shown in Figures 8(a), 8(c), and 8(d). In the defensive situation, the red aircraft first turns away from its opponent who has the advantage and then looks for opportunities to establish an advantage situation. There are no fierce rivalries in the process of combat, as shown in Figure 8(b).

Although π_5^r is an intelligent strategy, well-trained π_5^b can still gain an advantage in combat, as shown in Figures 8(e)–8(h). In the offensive situation (from the red aircraft's perspective), the blue aircraft cannot easily get rid of red one's following, and after fierce confrontation, it can establish an advantage situation, as shown in Figure 8(e). Figure 8(f) shows the defensive situation that the blue aircraft can keep its advantage until it wins. In the other two scenarios, the blue aircraft performs well, especially in the initial situation of neutral, which adopts the delayed-turning tactics and successfully wins in air combat.

$\pi_9^b$is an intelligent strategy, and well-trained π_{10}^r can achieve more than half of the victory and less than 10% of the failure. However, unlike π_1^r vs. π_0^b, π_{10}^r cannot easily win, especially in head-on situations, as shown in Figures 8(i)–8(l).

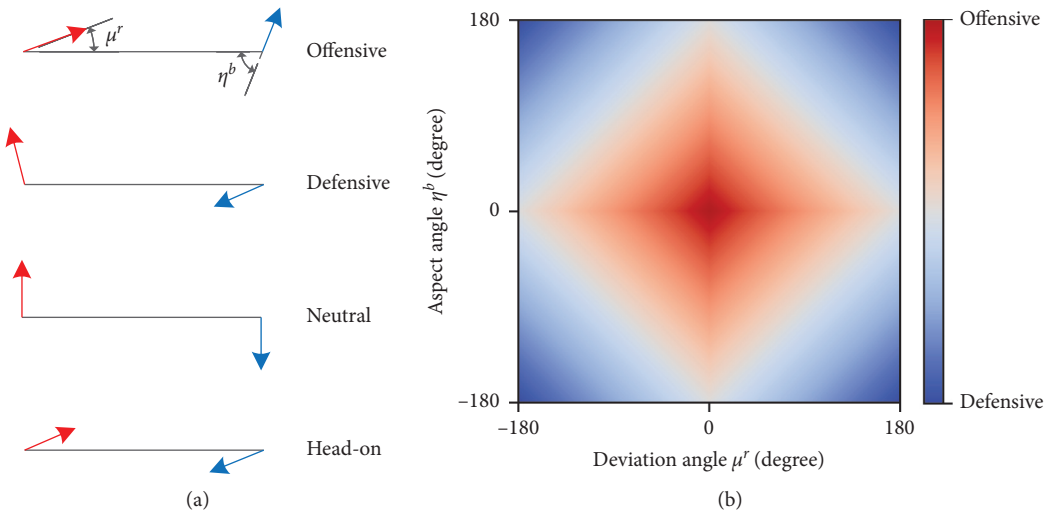

Figure 6: Situation recognition based on deviation angle and aspect angle. (a) Four initial air combat situation categories. (b) Situation recognition diagram.

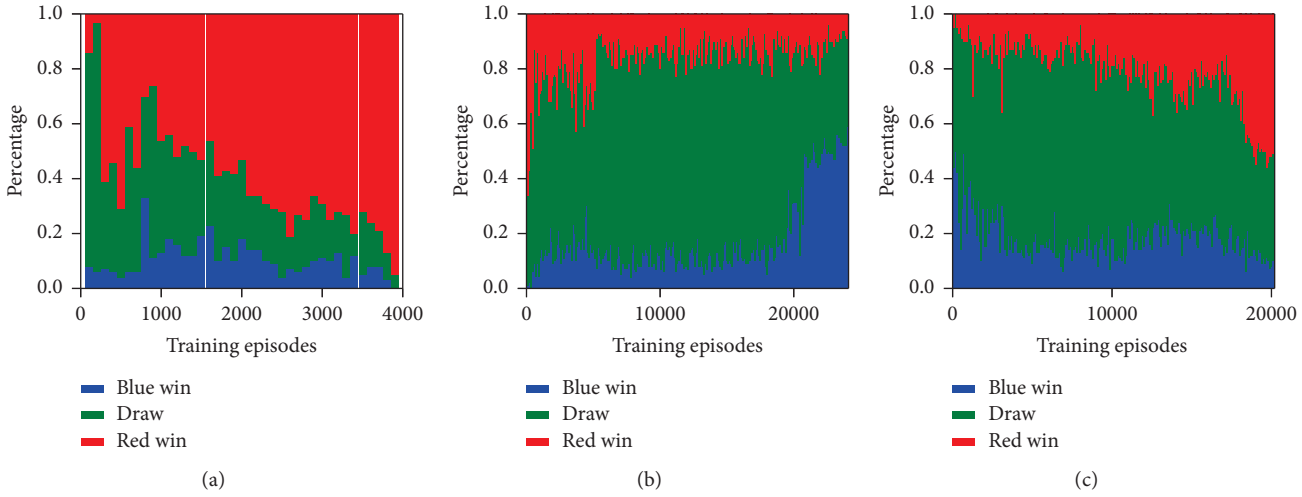

Figure 7: Learning curves in the training process. (a) π_1^r vs. π_0^b. (b) π_5^b vs. π_5^r. (c) π_{10}^r vs. π_g^b.

In the offensive situation, the blue aircraft tries to get rid of the red one by constantly adjusting its bank angle, but it is finally captured by the red one. In the defensive situation, the red aircraft adopted the circling back tactics cleverly and quickly captured its opponent. In the head-on situation, the red and blue aircrafts alternately occupy an advantage situation, and after fierce maneuver confrontation, the red aircraft finally wins. In the neutral situation, the red one constantly adjusts its bank angle according to the opponent's position and wins.

For the above three pairs of strategies, the head-on initial situations are taken as examples to analyze the advantages of both sides in the game, as shown in Figure 9. For the π_1^r vs. π_0^b scenario, because π_0^b has no intelligent maneuver guidance ability, although both sides are equally competitive at first half time, the red aircraft continues to expand its advantages until it wins, as shown in Figure 9(a). When the blue agent learns the intelligence strategy, it can defeat the red one, as shown in Figure 9(b). For the π_{10}^r vs. π_9^b scenario, the two sides have alternately established an advantage position, and the red aircraft has not won easily, as shown in Figure 9(c). Combining with the trajectories shown in Figures 8(c), 8(g), and 8(k), the maneuver strategies of both sides have been improved using the alternate freeze game DQN algorithm. The approach has the benefit of discovering new maneuver tactics.

4.3. League Results. The objective of the league system is to select a red agent who generates maneuver guidance instructions according to its strategy, so as to achieve the best results without knowing the strategy of its opponent. There are ten agents on each side, which are saved in the iteration of games. Each red agent fights each blue agent 400 times, including 100 times for each of four initial scenarios. The detailed league results are shown in Table 2. Results with a win/loss ratio greater than 1 are highlighted in bold.

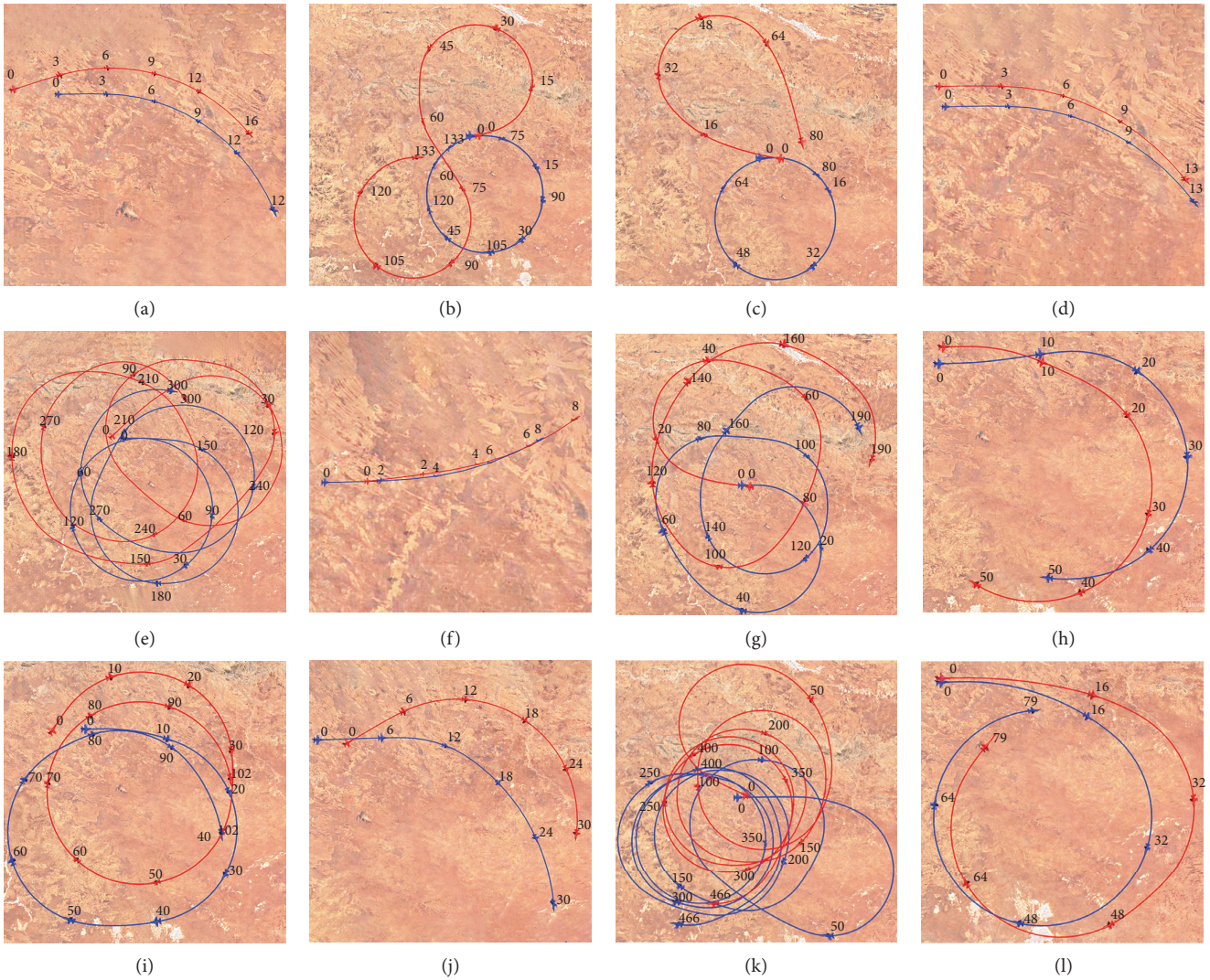

Figure 8: Typical trajectory results in the training iteration of air combat games. (a) π_1^r vs. π_0^b, offensive. (b) π_1^r vs. π_0^b, defensive. (c) π_1^r vs. π_0^b, head-on. (d) π_1^r vs. π_0^b, neutral. (e) π_5^b vs. π_5^r, offensive. (f) π_5^b vs. π_5^r, defensive. (g) π_5^b vs. π_5^r, head-on. (h) π_5^b vs. π_5^r, neutral. (i) π_{10}^r vs. π_g^b, offensive. (j) π_{10}^r vs. π_g^b, defensive. (k) π_{10}^r vs. π_g^b, head-on. (l) π_{10}^r vs. π_g^b, neutral.

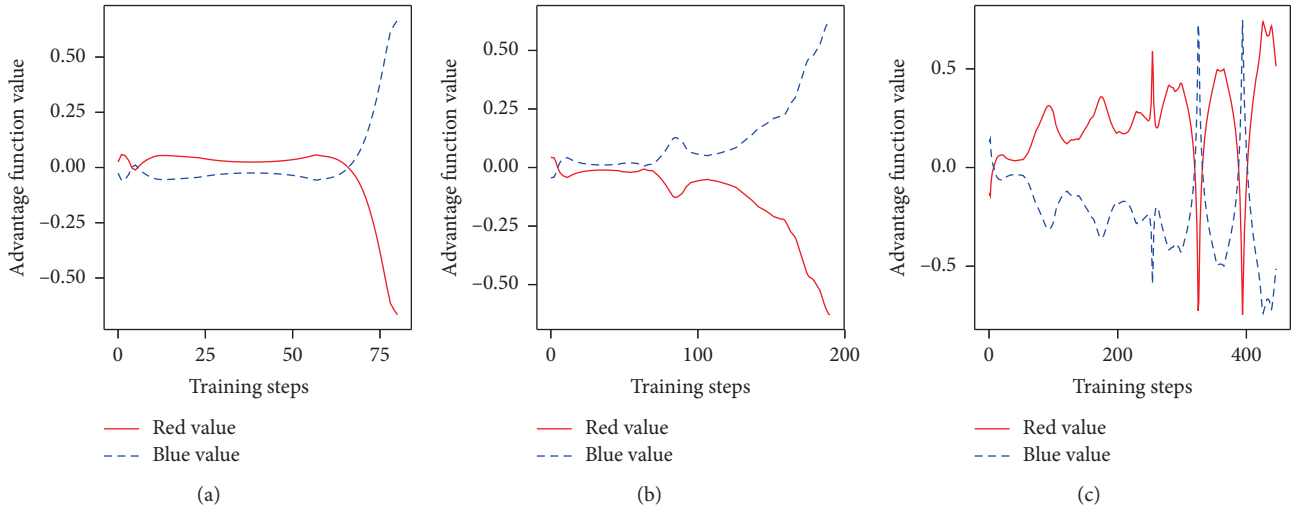

Figure 9: Advantage curves in head-on air combat games. (a) π_1^r vs. π_0^b. (b) π_5^b vs. π_5^r. (c) π_{10}^r vs. π_g^b.

Table 2: League results.

		π_0^b	π_1^b	π_2^b	π_3^b	π_4^b	π_5^b	π_6^b	π_7^b	π_8^b	π_9^b	π_{10}^b
π_1^r	Win	**393**	17	83	60	137	75	96	126	119	133	123
	Draw	**7**	103	213	198	95	128	131	117	113	113	114
	Loss	**0**	280	104	142	168	197	173	157	168	154	163
π_2^r	Win	**384**	**267**	4	82	78	94	88	135	142	**161**	152
	Draw	**16**	**120**	128	178	143	137	157	76	100	**87**	94
	Loss	**0**	**13**	268	140	179	169	155	189	158	**152**	154
π_3^r	Win	**387**	**185**	**253**	5	82	97	103	123	134	**154**	131
	Draw	**11**	**104**	**131**	115	194	148	145	132	84	**102**	97
	Loss	**2**	**111**	**16**	280	124	155	152	145	182	**144**	172
π_4^r	Win	**381**	**172**	**182**	**237**	31	92	137	**154**	**149**	**167**	**158**
	Draw	**19**	**123**	**102**	**117**	110	151	101	**97**	**144**	**92**	**104**
	Loss	**0**	**105**	**116**	**46**	259	157	162	**149**	**107**	**141**	**138**
π_5^r	Win	**382**	**164**	**172**	**187**	**221**	29	95	137	**168**	**154**	**162**
	Draw	**18**	**122**	**113**	**142**	**139**	144	118	103	**121**	**143**	**113**
	Loss	**0**	**114**	**115**	**71**	**40**	227	187	160	**111**	**103**	**125**
π_6^r	Win	**380**	**162**	**177**	**193**	**197**	**217**	52	**106**	**134**	**152**	**143**
	Draw	**19**	**105**	**98**	**132**	**101**	**134**	139	**194**	**176**	**131**	**144**
	Loss	**1**	**133**	**125**	**75**	**102**	**49**	209	**100**	**90**	**117**	**113**
π_7^r	Win	**383**	**169**	**169**	**180**	**165**	**187**	**210**	56	82	122	**133**
	Draw	**14**	**98**	**111**	**105**	**123**	**134**	**133**	137	180	132	**156**
	Loss	**3**	**133**	**120**	**115**	**112**	**79**	**57**	207	138	144	**111**
π_8^r	Win	**383**	**154**	**157**	**172**	**168**	**177**	**182**	**213**	36	107	114
	Draw	**16**	**123**	**109**	**114**	**132**	**102**	**97**	**134**	141	124	138
	Loss	**1**	**123**	**134**	**114**	**100**	**121**	**121**	**53**	223	169	148
π_9^r	Win	**387**	**140**	**162**	**157**	**154**	**162**	**179**	**182**	**215**	37	102
	Draw	**11**	**133**	**97**	**105**	**127**	**131**	**97**	**104**	**149**	125	147
	Loss	**2**	**127**	**141**	**138**	**119**	**107**	**124**	**114**	**36**	238	151
π_{10}^r	Win	**379**	**155**	146	147	143	**159**	**167**	**169**	**170**	**219**	42
	Draw	**10**	**102**	84	92	105	**97**	**131**	**104**	**85**	**140**	131
	Loss	**11**	**143**	170	161	152	**144**	**102**	**127**	**145**	**41**	227

π_i^r, where $i \in [1, 10]$, is trained in the environment with π_{i-1}^b as its opponent, and π_j^b is trained against π_j^r, $j \in [1, 10]$. The result of π_1^r fighting with π_0^b is ideal. However, since π_0^b is a static strategy rather than an intelligent strategy, π_1^r does not perform well in the game with other intelligent blue agents. For π_2^r to π_{10}^r, although there is no special training for against π_0^b, the performance is still very good because π_0^b is a static strategy. In the iterative game process, the trained agents can get good results in the confrontation with the frozen agents in the environment. For example, π_2^r has an overwhelming advantage over π_1^b, so does π_2^b and π_2^r.

Because of the *red queen* effect, although π_{10}^r has an advantage against π_9^b, it is in a weak position against the early blue agents, such as π_2^b, π_3^b, and π_4^b. The league table is shown in Table 3, in which 3 points are for win, 1 point for draw, and 0 points for loss. In the early stage of training, the performance of the trained agents will be gradually improved. The later trained agents are better than the former ones, such as from π_1^r to π_6^r. As the alternate freeze training goes on, the performance does not get better, such as π_7^r to π_{10}^r. The top of the score table is π_6^r, which not only has the highest score but also has advantages when playing with all blue agents except π_6^b, as shown in Table 2. In the competition with all the opponents, it can win 44% and remain unbeaten at 75%.

In order to verify the performance of this agent, it is confronted with the opponent agents trained by ADP [20] and Minimax-DQN [44]. Each of the four typical initial situations is performed 100 times for both opponents, and the results are shown in Table 4. The time cost to generate an action for each algorithm is shown in Table 5. It can be found that the performance of the agent presented in this paper is comparable to that of ADP, and the computational time is slightly reduced. However, ADP is a model-based method, which is assumed to know all the information such as the roll rate of the opponent. Compared with Minimax-DQN, the agent has the overall advantage. This is because Minimax-DQN assumes that the opponent is rational, while the opponent in this paper is not, which is closer to the real world. In addition, compared with Minimax-DQN, the computational efficiency of the algorithm proposed has been significantly improved because linear programming is used to select the optimal action at each step in Minimax-DQN, which is time consuming.

4.4. Agent Evaluation and Behavior Analysis. Evaluating agent performance in air combat is not simply a matter of either winning or losing. In addition to winning and losing rate, two criteria are added to represent success level: average

Table 3: League table.

Rank	Agent (strategy)	Win	Draw	Loss	Score
1	π_6^r	**1913**	**1373**	**1114**	**7112**
2	π_7^r	1858	1323	1219	6897
3	π_5^r	1871	1276	1253	6889
4	π_9^r	1877	1226	1297	6857
5	π_8^r	1863	1230	1307	6819
6	π_{10}^r	1896	1081	1423	6769
7	π_4^r	1860	1160	1380	6740
8	π_3^r	1654	1263	1483	6225
9	π_2^r	1587	1236	1577	5997
10	π_1^r	1362	1332	1706	5418

Table 4: Algorithm comparison.

		vs. ADP [20]	vs. Minimax-DQN [44]
Offensive	Win	57	66
	Draw	28	21
	Loss	15	13
Defensive	Win	19	20
	Draw	31	42
	Loss	50	38
Head-on	Win	43	46
	Draw	16	26
	Loss	41	28
Neutral	Win	32	43
	Draw	32	22
	Loss	36	35

Table 5: Time cost to generate an action.

Algorithm	Time cost (ms)
The proposed algorithm	13.8
ADP	14.2
Minimax-DQN	786

time to win (ATW) and average disadvantage time (ADT). ATW is measured as the average elapsed time required to maneuver to the advantage position in the scenario where our aircraft wins. Smaller ATW is better than a larger one. ADT is the average accumulated time that the advantage function value of our aircraft is less than that of the opponent, which is used as a criterion to evaluate the risk exposure from the adversary weapons.

A thousand air combat scenarios are generated in simulation software, and the opponent in each scenario is randomly chosen from ten blue agents. Each well-trained red agent is used to perform these confrontations, and the results are shown in Figure 10. Agent number i means the policy trained after i periods. From π_1^r to π_5^r, the performance of agents is gradually improving. After π_5^r, their performance is stabilized, and each agent has different characteristics. Comparing π_5^r to π_{10}^r, the winning and losing rates are almost the same as those in Table 3. The winning rate of the six agents is similar, with the highest π_5^r winning 6% higher than the lowest π_{10}^r, as shown in Figure 10(a). However, π_6^r is the agent with the lowest losing rate, and its losing rate is more than 10% lower than that of other agents, as shown in Figure 10(b). For ATW, π_7^r is less than 100 s, π_6^r is 108.7 s, and that of other agents is more than 110 s, as shown in Figure 10(c). For ADT, π_5^r, π_6^r, and π_8^r are less than 120 s, π_7^r is above 130 s, and π_9^r and π_{10}^r are between 120 and 130 s, as shown in Figure 10(d).

In summary, with the increase in the number of iterations, the *red queen* effect appears obviously. The performance of π_8^r, π_9^r, and π_{10}^r is no better than that of π_6^r and π_7^r. Although the winning rate of π_7^r is not high and has more disadvantage time, it can always win quickly. It is an aggressive agent and can be used in scenarios where our aircraft needs to beat the opponent quickly. The winning rate of π_6^r is similar to other agents, while its losing rate is the lowest. Its ATW is only higher than π_7^r, and ADT is the lowest. In most cases, it can be selected to achieve more victories while keeping itself safe. π_6^r is an agent with good comprehensive performance, which means that the selection method of the league system is effective.

For the four initial scenarios, air combat simulation using π_6^r is selected for behavior analysis, as shown in Figure 11. In the offensive scenario, the opponent tried to escape by turning in the opposite direction. Our aircraft used a lag pursuit tactic, which is simple but effective. At the 1st second, our aircraft did not keep up with the opponent but chose to fly straight. At the 9th second, it gradually turned to the opponent, established, and maintained the advantage and finally won, as shown in Figure 11(a).

In the defensive scenario, our aircraft was at a disadvantage position, and the opponent was trying to lock us in

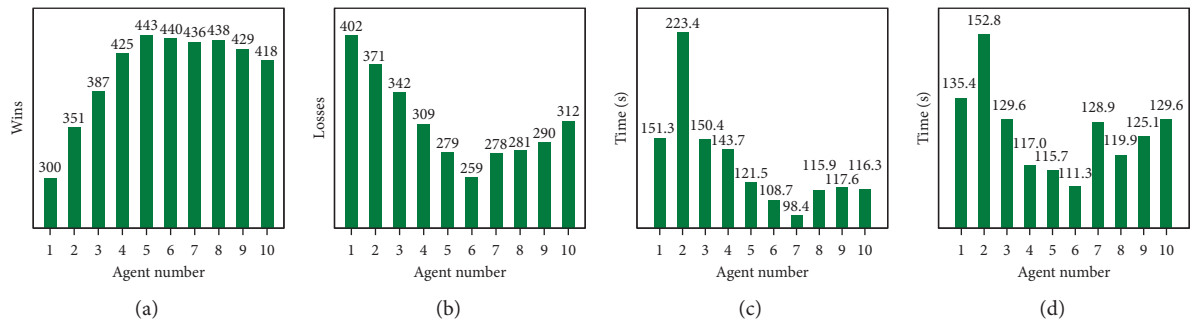

Figure 10: Agent evaluation. (a) Number of winning. (b) Number of losing. (c) Average time to win. (d) Average disadvantage time.

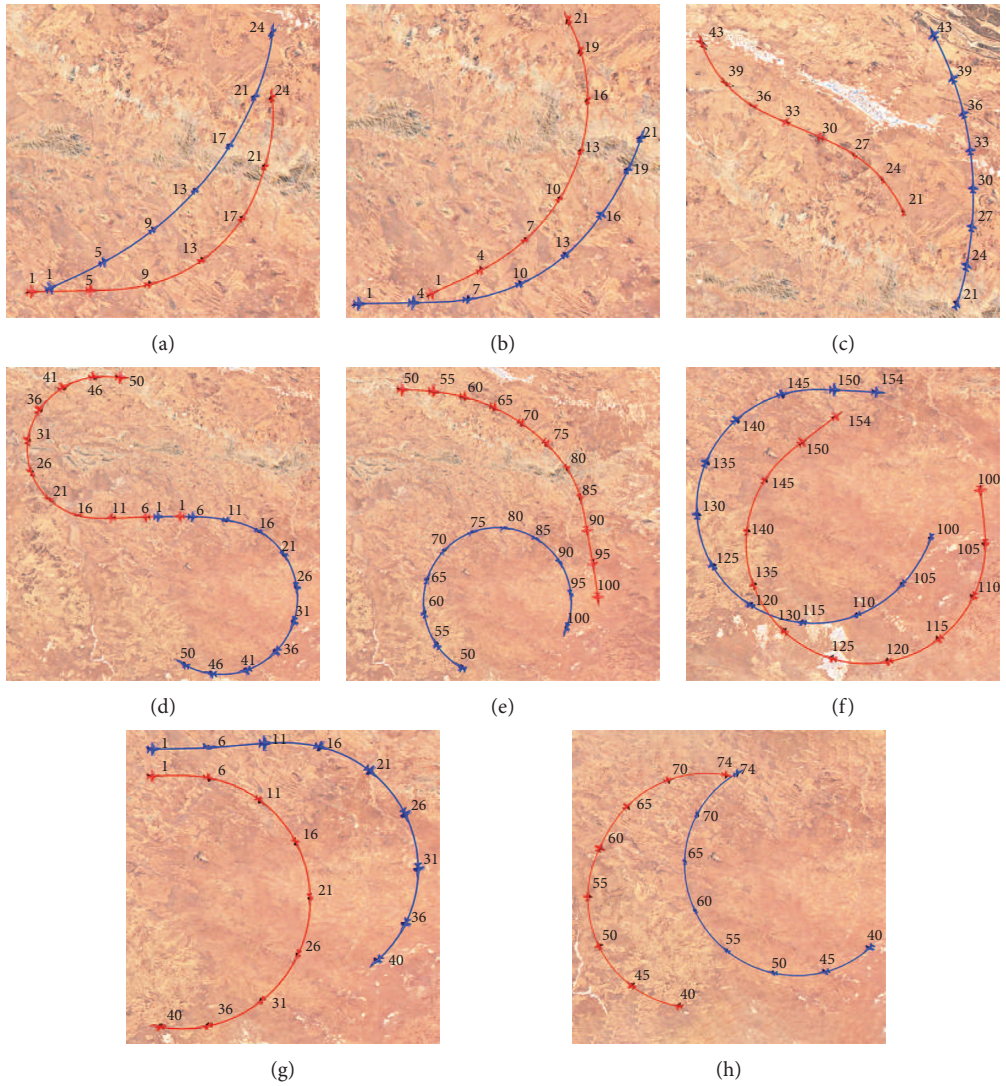

Figure 11: Trajectories under four typical initial situations. (a) Offensive. (b) Defensive (0–21 s). (c) Defensive (21–43 s). (d) Head-on (0–50 s). (e) Head-on (50–100 s). (f) Head-on (100–154 s). (g) Neutral (0–40 s). (h) Neutral (40–74 s).

with a lag pursuit tactic, as shown in Figure 11(b). At the 30th second, our aircraft adjusted its roll angle to the right to avoid being locked by the opponent, as shown in Figure 11(c). After that, it continued to adjust the roll angle, and when the opponent noticed that our strategy had changed, our aircraft had already flown out of the sector. In situations where it is difficult to win, the agent will use a safe maneuver strategy to get a draw.

In the head-on scenario, both sides adopted the same maneuver strategy in the first 50 s, that is, the maximum overload turn and waiting for opportunities, as shown in Figure 11(d). The crucial decision was made in the 50th second; the opponent was still hovering and waiting for an opportunity, while our aircraft stopped hovering and reduced its turning rate to fly towards the opponent. At the 90th second, the aircraft reached an advantage situation over its opponent, as shown in Figure 11(e). In the final stage, our aircraft adopted the lead pursuit tactic to establish and maintain the advantage and won, as shown in Figure 11(f).

In the neutral scenario, our initial strategy was to turn away from the opponent to find opportunities. The opponent's strategy was to lag pursuit and successfully reached the rear of our aircraft at the 31st second, as shown in Figure 11(g). However, after the 40th second, our aircraft made a wise decision to reduce the roll angle, thereby increasing the turning radius. At the 60th second, the disadvantaged situation was terminated, as shown in Figure 11(h). After that, our aircraft increased its roll angle, and the opponent was in a state of maximum overload right-turn, which made it unable to get rid of our aircraft, and finally, it lost. It can be seen that trained by alternate freeze games using the DQN algorithm and selected through the league system, the agent can learn tactics such as lead pursuit, lag pursuit, and hovering and can use them in air combat.

5. Conclusion

In this paper, middleware connecting air combat simulation software and the reinforcement learning agent is developed. It provides an idea for researchers to design different middleware to transform existing software into reinforcement learning environment, which can expand the application field of reinforcement learning. Maneuver guidance agents with reward reshaping are designed. Through training in the environment, an agent can guide an aircraft to fight against its opponent in air combat and reach an advantage situation in most scenarios. An alternate freeze game algorithm is proposed and combined with RL. It can be used in nonstationarity situations where other players' strategies are variable. Through the league system, the agent with improved performance after iterative training is selected. The league results show that the strategy quality of the selected agent is better than that of the other agents, and the *red queen* effect is avoided. Agents can learn some typical angle tactics and behaviors in the horizontal plane and perform these tactics by guiding an aircraft to maneuver in one-on-one air combat.

In future works, the problem would be extended to 3D maneuvering with less restrictive vehicle dynamics. The 3D formulation will lead to a larger state space and more complex actions, and the learning mechanism would be improved to deal with them. Beyond an extension to 3D maneuvering, a 2-vs-2 air combat scenario would be established, in which there are collaborators as well as adversaries. One option is to use a centralized controller, which turns to a single-agent DRL problem to obtain the joint action of both aircrafts to execute in each time step. The other option is to adopt a decentralized system, in which both agents take a decision for themselves and might cooperate to achieve a common goal.

In theory, the convergence of the alternate freeze game algorithm needs to be analyzed. Furthermore, in order to improve the diversity of potential opponents in a more complex air combat scenario, the league system would be refined.

Nomenclature

(x, y, z):	3-dimensional coordinates of an aircraft
v:	Speed
θ:	Flight path angle
ψ:	Heading angle
β:	Angle of attack
ϕ:	Bank angle
m:	Mass of an aircraft
g:	Acceleration due to gravity
T:	Thrust force
L:	Lift force
D:	Drag force
$\dot{\psi}$:	Turn rate
$\dot{\phi}$:	Roll rate
d_t:	Distance between the aircraft and the target at time t
$d_{\min}$:	Minimum distance of the advantage position
$d_{\max}$:	Maximum distance of the advantage position
μ_t:	Deviation angle at time t
$\mu_{\max}$:	Maximum deviation angle of the advantage position
η_t:	Aspect angle at time t
$\eta_{\max}$:	Maximum aspect angle of the advantage position
S:	State space
S_t:	State vector at time t
s:	All states in the state space
A:	Action space
A_t:	Action at time t
a:	All actions in the action space
π: $S \longrightarrow A$:	Policy
$R(S_t, A_t, S_{t+1})$:	Reward function
R_t:	Scalar reward received at time t
$V(s)$:	Value function
$Q(s, a)$:	Action-value function
$Q^*(s, a)$:	Optimal action-value function
$Q(S_t, A_t)$:	Estimated action-value function at time t
γ:	Discount factor
α:	Learning rate
T:	Maximum simulation time in each episode
Δt:	Training time step size
δt:	Simulation time step size
$F(S_t, A_t, S_{t+1})$:	Shaping reward function
$\Phi(S_t)$:	A real-valued function on states
$T(S_t)$:	Termination reward function
$D(S_t)$:	Distance reward function
$O(S_t)$:	Orientation reward function
K:	Number of training periods.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 91338107 and U1836103 and the Development Program of Sichuan, China, under Grants 2017GZDZX0002, 18ZDYF3867, and 19ZDZX0024.

Supplementary Materials

Aircraft dynamics: the derivation of the kinematic and dynamic equations of the aircraft. Code: a simplified version of the environment used in this paper. It does not have commercial software and middleware, but its aircraft model and other functions are the same as those proposed in this paper. This environment and the RL agent are packaged as a supplementary material. Through this material, the alternate freeze game DQN algorithm proposed in this paper can be reproduced. (*Supplementary Materials*)

References

[1] L. R. Shaw, "Basic fighter maneuvers," in *Fighter Combat Tactics and Maneuvering*, M. D. Annapolis, Ed., Naval Institute Press, New York, NY, USA, 1st edition, 1985.

[2] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. Herik, *Dynamic Scripting with Team Coordination in Air Combat Simulation*, Lecture Notes in Computer Science, Kaohsiung, Taiwan, 2014.

[3] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. Herik, "Centralized versus decentralized team coordination using dynamic scripting," in *Proceedings of the European Modeling and Simulation Symposium*, pp. 1–7, Porto, Portugal, 2014.

[4] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. Herik, "Rewarding air combat behavior in training simulations," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 1397–1402, Kowloon, China, 2015.

[5] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. Herik, "Transfer learning of air combat behavior," in *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 226–231, Miami, FL, USA, 2015.

[6] D. I. You and D. H. Shim, "Design of an aerial combat guidance law using virtual pursuit point concept," *Proceedings of the Institution of Mechanical Engineers*, vol. 229, no. 5, pp. 1–22, 2014.

[7] H. Shin, J. Lee, and D. H. Shim, "Design of a virtual fighter pilot and simulation environment for unmanned combat aerial vehicles," in *Prodeedings of the Advances in Flight Control Systems*, pp. 1–21, Grapevine, TX, USA, 2017.

[8] J. M. Eklund, J. Sprinkle, and S. S. Sastry, "Implementing and testing a nonlinear model predictive tracking controller for aerial pursuit/evasion games on a fixed wing aircraft," in *Poceedings of the Control System Applications*, pp. 1509–1514, Portland, OR, USA, 2005.

[9] J. M. Eklund, J. Sprinkle, and S. S. Sastry, "Switched and symmetric pursuit/evasion games using online model predictive control with application to autonomous aircraft," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 3, pp. 604–620, 2012.

[10] F. Austin, G. Carbone, M. Falco, H. Hinz, and M. Lewis, *Automated Maneuvering Decisions for Air-to-Air Combat*, Grumman Corporate Research Center, Bethpage, NY, USA, 1987.

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

[12] Y. Ma, G. Wang, X. Hu, H. Luo, and X. Lei, "Cooperative occupancy decision making of multi-uav in beyond-visual-range air combat a game theory approach," *IEEE Access*, vol. 323, 2019.

[13] J. P. Hespanha, M. Prandini, and S. Sastry, "Probabilistic pursuit-evasion games: a one-step Nash approach," in *Proceedings of the 36th IEEE Conference on Decision and Control*, Sydney, NSW, Australia, 2000.

[14] A. Xu, Y. Kou, L. Yu, B. Xu, and Y. Lv, "Engagement maneuvering strategy of air combat based on fuzzy markov game theory," in *Proceedings of IEEE International Conference on Computer*, pp. 126–129, Wuhan, China, 2011.

[15] K. Horic and B. A. Conway, "Optimal fighter pursuit-evasion maneuvers found via two-sided optimization," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 1, pp. 105–112, 2006.

[16] K. Virtanen, T. Raivio, and R. P. Hamalainen, "Modeling pilot's sequential maneuvering decisions by a multistage influence diagram," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 4, pp. 665–677, 2004.

[17] K. Virtanen, J. Karelahti, and T. Raivio, "Modeling air combat by a moving horizon influence diagram game," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 5, pp. 1080–1091, 2006.

[18] Q. Pan, D. Zhou, J. Huang et al., "Maneuver decision for cooperative close-range air combat based on state predicted influence diagram," in *Proceedings of the Information and Automation for Sustainability*, pp. 726–730, Macau, China, 2017.

[19] R. E. Smith, B. A. Dike, R. K. Mehra, B. Ravichandran, and A. El-Fallah, "Classifier systems in combat: two-sided learning of maneuvers for advanced fighter aircraft," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 421–437, 2000.

[20] J. S. McGrew, J. P. How, B. Williams, and N. Roy, "Air-combat strategy using approximate dynamic programming," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 5, pp. 1641–1654, 2010.

[21] Y. Ma, X. Ma, and X. Song, "A case study on air combat decision using approximated dynamic programming," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[22] I. Bisio, C. Garibotto, F. Lavagetto, A. Sciarrone, and S. Zappatore, "Blind detection: advanced techniques for WiFi-based drone surveillance," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 938–946, 2019.

[23] Y. Li, "Deep reinforcement learning," 2018.

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, London, UK, 2nd edition, 2018.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[26] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[27] Z. Kavukcuoglu, R. Liu, Z. Meng, Y. Zhang, Y. Yu, and T. Lu, "On reinforcement learning for full-length game of starcraft," 2018.
[28] D. Silver and C. J. Maddison, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[29] D. Hubert and I. Antonoglou, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
[30] J. Schrittwieser, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: a survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
[31] A. Waldock, C. Greatwood, F. Salama, and T. Richardson, "Learning to perform a perched landing on the ground using deep reinforcement learning," *Journal of Intelligent and Robotic Systems*, vol. 92, no. 3-4, pp. 685–704, 2018.
[32] R. R. Alejandro, S. Carlos, B. Hriday, P. Paloma, and P. Campoy, "A deep reinforcement learning strategy for UAV autonomous landing on a moving platform," *Journal of Intelligent and Robotic Systems*, vol. 93, no. 1-2, pp. 351–366, 2019.
[33] W. Lou and X. Guo, "Adaptive trajectory tracking control using reinforcement learning for quadrotor," *Journal of Intelligent and Robotic Systems*, vol. 13, no. 1, pp. 1–10, 2016.
[34] C. Dunn, J. Valasek, and K. Kirkpatrick, "Unmanned air system search and localization guidance using reinforcement learning," in *Proceedings of the of the AIAA Infotech Aerospace 2014 Conference*, pp. 1–8, Garden Grove, CA, USA, 2014.
[35] S. You, M. Diao, and L. Gao, "Deep reinforcement learning for target searching in cognitive electronic warfare," *IEEE Access*, vol. 7, pp. 37432–37447, 2019.
[36] P. Luo, J. Xie, and W. Che, C. Man, "Q-learning based air combat target assignment algorithm," in *Proceedings of IEEE International Conference on Systems*, pp. 779–783, Budapest, Hungary, 2016.
[37] M. Grzes, S. Paulo, "Reward shaping in episodic reinforcement learning," in *Proceedings of the International Foundation for Autonomous Agents and Multiagent Systems*, pp. 565–573, Brazil, 2017.
[38] K. Tummer and A. Agogino, H. Hakodate, "Agent reward shaping for alleviating traffic congestion," in *Proceedings of the International Foundation for Autonomous Agents and Multiagent Systems*, pp. 1–7, Japan, 2006.
[39] L. L. B. V. Cruciol, A. C. de Arruda, L. Weigang, L. Li, and A. M. F. Crespo, "Reward functions for learning to control in air traffic flow management," *Transportation Research Part C: Emerging Technologies*, vol. 35, pp. 141–155, 2013.
[40] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: dealing with non-stationarity," 2019.
[41] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the International Conference on International Conference on Machine Learning*, pp. 157–163, New Brunswick, NJ, USA, 1994.
[42] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proceedigs of the International Conference on International Conference on Machine Learning*, pp. 322–328, Williamstown, MA, USA, 2001.
[43] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, vol. 4, pp. 1039–1069, 2003.
[44] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," 2020.
[45] R. E. Smith, B. A. Dike, B. Ravichandran, A. Fallah, and R. Mehra, "Two-sided, genetics-based learning to discover novel fighter combat maneuvers," in *Lecture Notes in Computer Science*, pp. 233–242, Springer, Berlin, Heidelberg, 2001.
[46] A. Ng, D. Harada, and S. Russell, S. Bled, "Policy invariance under reward transformations: theory and application to reward shaping," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pp. 278–287, Berlin, Germany, 1999.
[47] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," in *Proceedings of the International Conference on Learning Representations*, Puerto Rico, MA, USA, 2016.
[48] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pp. 1–19, New York, NY, USA, 2016.
[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.
[50] Commercial Air Combat Simulation Software FG_SimStudio, http://www.eyextent.com/producttail/11/7.
[51] M. Wiering and M. van Otterlo, *Reinforcement Learning: State-of-The-Art*, Springer Press, Heidelberg, Germany, 2014.
[52] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pp. 1–15, San Diego, CA, USA, 2015.