IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 8, 2021, accepted February 14, 2021, date of publication February 19, 2021, date of current version March 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3060426

Application of Deep Reinforcement Learning in Maneuver Planning of Beyond-Visual-Range Air Combat

DONGYUAN HU, RENNONG YANG, JIALIANG ZUO, ZE ZHANG, JUN WU, AND YING WANG
Air Force Engineering University, Xi'an 710053, China
Corresponding author: Dongyuan Hu (hudyuan@163.com)

ABSTRACT Beyond-visual-range (BVR) engagement becomes more and more popular in the modern air battlefield. The key and difficulty for pilots in the fight is maneuver planning, which reflects the tactical decision-making capacity of the both sides and determinates success or failure. In this paper, we propose an intelligent maneuver planning method for BVR combat with using an improved deep Q network (DQN). First, a basic combat environment builds, which mainly includes flight motion model, relative motion model and missile attack model. Then, we create a maneuver decision framework for agent interaction with the environment. Basic perceptive variables are constructed for agents to form continuous state space. Also, considering the threat of each side missile and the constraint of airfield, the reward function is designed for agents to training. Later, we introduce a training algorithm and propose perceptional situation layers and value fitting layers to replace policy network in DQN. Based on long short-term memory (LSTM) cell, the perceptional situation layer can convert basic state to high-dimensional perception situation. The fitting layer does well in mapping action. Finally, three combat scenarios are designed for agent training and testing. Simulation result shows the agent can avoid the threat of enemy and gather own advantages to threat the target. It also proves the models and methods of agents are valid and intelligent air combat can be realized.

INDEX TERMS Beyond visual range, intelligent air combat, maneuver planning and decision, missile kill envelope, deep reinforcement learning, LSTM-DQN.

I. INTRODUCTION

With the development from informatization to intelligence, technological progress will hasten the revolution of war. The control of intelligence which dominated by intelligent operations, will be the focus of the next war. Based on intelligent algorithms, the maneuver planning could help pilots quickly tactically decides and gain profitable positions in the air battlefield. Intelligent maneuver planning attracts the attention of many scholars academically, which also has great application value in practice. This is the focus of our research as well.

In the current war, the hardware performance of radar and missiles improves rapidly. Beyond-visual-range (BVR) air combat has become the preferred combat method to gain air supremacy. The fighters use airborne detection equipment to lock down their opponents and attack with medium-range or long-range air-to-air missiles outside the visual range (more than 15km). The combat mode originated from the Vietnam War and gradually matured during the Gulf War. In the Gulf War, the detection range of most aircraft's fire control radar could reach more than 100km, the tracking distance also reached tens of kilometers, and it could track multiple targets at the same time. With the gradual maturity of the third-generation aircraft and the rapid development of the fourth-generation aircraft technology, BVR combat have gradually become the mainstream style for pilots to combat. While the research on BVR mostly focuses on effectiveness evaluation [1]–[3] and radar target detection [4], [5], there is little research on maneuver planning at present.

Under using artificial intelligence technology, intelligent maneuver planning can sense the two sides battlefield data, calculate the best tactical tactics, provide pilots with a maneuver planning solution. The target of the methods is to get the aircraft out of the enemy's attack quickly and accumulate own advantages gradually. Maneuver planning is also called tactical decision making or maneuvering decision.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenbao Liu.

The current research on maneuvering decision mainly focuses on short-range combat rather than BVR. Two reasons lead to this. One is the medium and long-range air-to-air missiles are mainly used for BVR. The tactical mechanism and rules on missile are different from the short-range air combat. Largely, no mature theory has yet been formed to guide pilots. Second, the two sides cannot use real missile to training. It is difficult to calculate the damage assessment of the missile when one side makes a maneuvering action.

For the first one, we refer to the method of close air combat and consult professional pilots for BVR experience. For the second, we introduce missile kill envelope concept [4], [5] to evaluate opponents. Different from the traditional missile attack zone, missile kill envelope can provide a basis for the situation assessment to the warring parties.

Many scholars have done in-depth research on close air combat and proposed most mature algorithms. We summarize these methods and divide into two categories: reactive decision-making and deductive decision-making.

Reactive decision-making belongs to an optimization method, whose basis is to set up rules of engagement and match them with the current state. The best maneuver planning solution can be search within the rules through the use of optimization algorithm. For example, expert system method [8]–[10], bayesian network method [11]–[13] and fuzzy reasoning method [14], [15]. The result in [8] shows that expert system method can meet the requirement of time limitation, while it's hard to deal with the state out of traditional sample. Ernest [15] introduces a genetic fuzzy method to deal with situations outside of the rules and made great process in optimization. While the fuzzy trees proposed in [15] are too complicated to use in BVR.

Deductive decision-making method, such as game theory [17], [18], dynamic programming (DP) [19], [20], and Monte Carlo search [21], can give a solution considering the next n-step decisions from the current state. Those methods can find solutions without rule experience, but real-time effect is poor. The result in [21] shows the search time exceeds the decision time, so the state and action have to be simplified. To decrease time complexity, literature [20] proposes an approximate dynamic programming to improve traditional DP [19], and approximately replace the global optimal solution with a local optimal value.

In recent years, with the successful application of deep reinforcement learning (DRL) in complex sequential decision problem such as AlfaGo [23], AlfaGo Zero [24], and AlfaStar, it has become possible that using reinforcement learning to solve air combat maneuver decision-making problem. The method of offline learning and online decision-making can satisfy both the time limit and the optimal solution requirement.

The application of reinforcement learning in air combat is mainly based on the value function search [25]–[27] and policy search [28], [29]. Literature [25] applies the Q-learning method to air combat, but does not perform separate learning for the red and blue agents. Q-learning is a model-based learning technique getting good effects in aerial vehicle guidance [26]. The simulation experiments in [26] achieve faster convergence compared with traditional guidance schemes. Literature [27] improves the model and trains network for each single agent with Q-learning method, while the use of discrete state space and discrete action space make the results of air combat lack continuity. Literature [29] uses the actor-critic (AC) framework to solve the maneuver decision problem and designs the reward function, the algorithm is only effective in two-dimensional space. DDPG is applied to air combat decision-making and the effectiveness is tested in [30]. However, the design of the reward and punishment function is relatively simple, and it is difficult for the agent to learn complex tactical strategies.

In general, the above studies have greatly improved the efficiency and performance of decision algorithm, while most of them can be applied to close-range air combat and cannot be directly used in BVR combat. At the same time, many methods ignore three common matters. One is the missile attack area, which is the key in air combat. This type of method is only suitable for gun combat but not for medium-range or long-range air combat with missiles. The second is the design of reward function. The reward and punishment value directly affects whether the agent can learn the regularities of combat. The third is the network's ability to sense confrontation. It is difficult for agents to get perceptual information directly from state features with a simple network.

In view of the three problems, this paper mainly has the following contributions:

1. Missile attack zone and missile kill envelope are comprehensively considered for BVR combat in the paper. We use the missile attack zone to judge whether the confrontation stop, and propose four missile kill envelopes as the basis for assessing both sides.
2. We design the reward function with two parts. One is the basic punishment value, which mainly focuses on battlefield boundary and reducing meaningless engagement. The other is the situational punishment value, which mainly reflects potential threat of the missile to opponents with using kill zone. It is useful for agents to learn engagement rules more intuitively.
3. Aiming at the problem the directly mapping from state to action is difficult to achieve, we propose an improvement algorithm for the DQN. The LSTM cell is used to construct perceptional situation network which can process basic motion state and output high-dimensional perception variables. Besides, we introduce deep full connection network to fit the value function. The whole LSTM-DQN algorithm does well in agent's state-situation-action mapping.

II. AIR COMBAT ENVIRONMENT DESIGN

In this section, we set up the basic model for one-to-one air combat confrontation environment, including flight motion model, relative motion model and missile attack model.

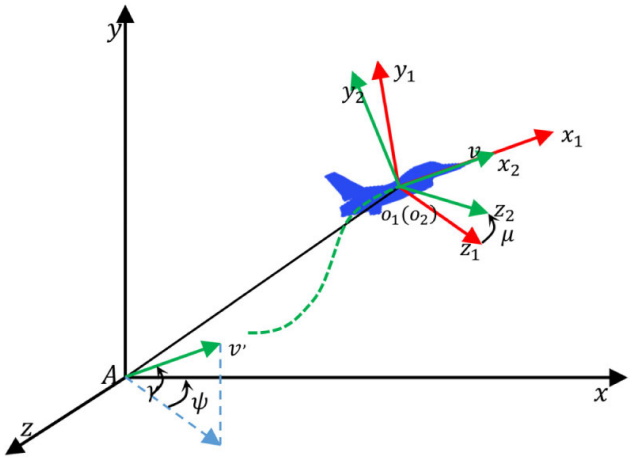


FIGURE 1. Three-degree-of-freedom motion model of the aircraft in ground coordinate system(*AXYZ*). $O_1x_1y_1z_1$ is the airframe coordinate system, $O_2x_2y_2z_2$ is the track coordinate system, μ is the velocity roll angle. v' is the translation of vector v to the origin A.

A. FLIGHT MOTION MODEL

The ground coordinate system is stationary relative to the earth. The equation of motion of the aircraft will be more concise in this reference system, as in equation (1).

$$\begin{cases} \dot{x} = v\cos\gamma\sin\psi \\ \dot{y} = v\cos\gamma\cos\psi \\ \dot{z} = v\sin\gamma \end{cases} \tag{1}$$

where $\dot{x}, \dot{y}, \dot{z}$ represents the change rate of position on each axis, v represents the aircraft's speed in the ground coordinate system, γ and ψ respectively represent track pitch angle and yaw angle. The flight diagram is shown in Fig. 1.

In the three-degree-of-freedom model, the airframe coordinate system $O_1x_1y_1z_1$ and the velocity coordinate system $O_3x_3y_3z_3$ can approximately coincide. The axis of the airframe coordinate system o_1x_1 is in the same direction as the velocity vector, and the axis of the track coordinate system o_2x_2 is also coinciding with the velocity vector direction, the rotation angle of the two coordinate systems is the velocity roll angle μ. As shown in Fig. 2.

Three control variables can complete various types of aircraft maneuvers. During the flight, the overload formed by the thrust and lift acting on the aircraft can be decomposed into tangential overload and normal overload. The dynamic equation of the aircraft's mass center is constructed as follows:

$$\begin{cases} \dot{v} = g\,(n_x - \sin\gamma) \\ \dot{\gamma} = \dfrac{g}{v}\,(n_z\cos\mu - \cos\gamma) \\ \dot{\psi} = \dfrac{g n_z \sin\mu}{v\cos\gamma} \end{cases} \tag{2}$$

where n_x, n_z represent the tangential overload and normal overload of the aircraft. $[n_x, n_z, \mu]$ is basic control command in flight motion model and is used to execute all BVR tactical maneuver.

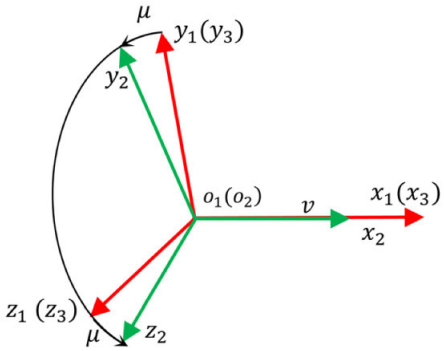


FIGURE 2. Track coordinate system $O_2x_2y_2z_2$ and velocity coordinate system $O_3x_3y_3z_3$.

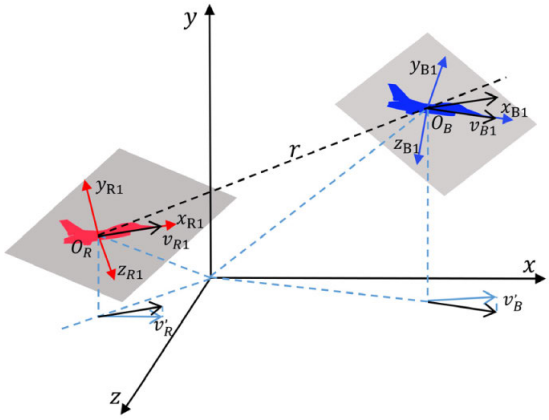


FIGURE 3. The spatial position and situation between red aircraft and blue aircraft in ground coordinate system. $O_Rx_{R1}y_{R1}z_{R1}$ and $O_Bx_{B1}y_{B1}z_{B1}$ represent red side airframe coordinate system and blue side airframe coordinate system respectively. v_{R1} and v_{B1} is speed of both sides in their self-system. v'_{R1} and v'_{B1} are projection in *oxz* plane.

B. RELATIVE MOTION MODEL

In BVR confrontation, the relative position and relative situation of the two parties are the basis for maneuver decision-making. The more precise and detailed the expression of the spatial relationship, the more conducive to situational perception and decision-making. Assuming that the coordinate position of the red side in the ground coordinate system is $P_R = (x_R, y_R, z_R)$, and the position of the blue side is $P_B = (x_B, y_B, z_B)$, the relative situation of the two sides is shown in Fig. 3.

The distance between red aircraft and blue aircraft is:

$$\begin{cases} \Delta x = x_B - x_R \\ \Delta y = y_B - y_R \\ \Delta z = z_B - z_R \\ r = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \end{cases} \tag{3}$$

where Δz is the relative height of the two sides, $\Delta z = 0$ represents the red and the blue are flying in the same altitude level. Otherwise it reflects the difference in height energy.

Using red aircraft as the benchmark, the speed vector of the red is converted to the ground coordinate system. r_{RB} defines the relative distance vector, whose direction is from the red

center to the blue one, and the value is r, r_{RB} can be calculate as follow:

$$\begin{cases} v_R = (v_{R1}\cos(\gamma_R)\cos(\psi_R), v_{R1}\sin(\gamma_R), \\ \qquad v_{R1}\cos(\gamma_R)\sin(\psi_R)) \\ v_{R1}^{'} = (v_{R1}\cos(\gamma_R)\cos(\psi_R), 0, \\ \qquad v_{R1}\cos(\gamma_R)\sin(\psi_R)) \\ r_{RB} = re_{RB} = (x_B - x_R, y_B - y_R, z_B - z_R) \end{cases} \tag{4}$$

The azimuth of the red side is:

$$\varphi_R = arc\cos(\frac{r_{RB}\cdot v_R}{|r_{RB}|\,|v_R|}) \tag{5}$$

Entry angle of the target (blue side) is:

$$q_B = arc\cos(\frac{r_{RB}\cdot v_B}{|r_{RB}|\,|v_B|}) \tag{6}$$

Using the same method, we can calculate blue aircraft parameters as follow:

$$\begin{cases} v_B = (v_{B1}\cos(\gamma_B)\cos(\psi_B), v_{B1}\sin(\gamma_B), \\ \qquad v_{B1}\cos(\gamma_B)\sin(\psi_B)) \\ v_{B1}^{'} = (v_{B1}\cos(\gamma_B)\cos(\psi_B), 0, \\ \qquad v_{B1}\cos(\gamma_B)\sin(\psi_B)) \\ r_{BR} = (x_R - x_B, y_R - y_B, z_R - z_B) \end{cases} \tag{7}$$

The azimuth of the blue side is:

$$q_R = arc\cos(\frac{r_{BR}\cdot v_R}{|r_{BR}|\,|v_R|}) \tag{8}$$

Entry angle of the target (red side):

$$q_R = arc\cos(\frac{r_{BR}\cdot v_R}{|r_{BR}|\,|v_R|}) \tag{9}$$

The angle between the two speed vectors is:

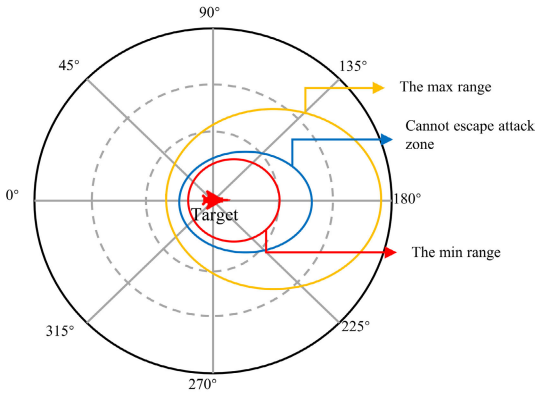
$$\eta_{RB} = arc\cos(\frac{v_R\cdot v_B}{|v_R|\,|v_B|}) \tag{10}$$

The angle formed by the projection vectors of the two velocity in the plane oxz is:

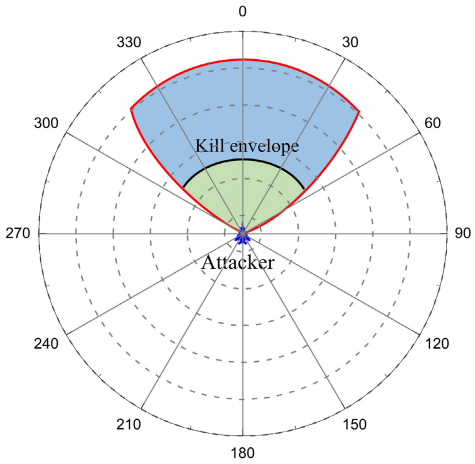
$$\eta_{RB}^{'} = arc\cos(\frac{v_{R1}^{'}\cdot v_{B1}^{'}}{|v_{R1}^{'}|\,|v_{B1}^{'}|}) \tag{11}$$

C. MISSILE ATTACK MODEL

Air-to-air missile attack area is an important indicator for measuring the threat range and operational ability. There are two common ways to describe the missile attack zone, the attack envelope and the kill envelope. The attack envelope is a closed-loop area centered on the target aircraft. The attacker can destroy the target with a certain probability when launching missiles in this envelope. As shown in Fig. 4 (a), there is a max boundary, a min boundary, and an inescapable zone. The kill envelope is centered on the attack aircraft, describing the attackable range of the air-to-air missiles carried on the attack aircraft. It takes a certain flight time for the missile to hit the target after being launched. The kill envelope can be divided according to the type of maneuver that the target performs during the missile flight time. As shown in Fig. 4(b).



(a) Attack envelope of missile



(b) Kill envelope of missile

FIGURE 4. The two type of missile attack zone.

1) ATTACK ENVELOPE

The following restrictions are mainly considered when calculating the attack envelope:

1. The maximum and minimum flying height of the missile;
2. The minimum speed of the missile before encountering the target;
3. Safety distance limit;
4. Maximum flight time of the missile.

Based on the above restrictions, the following types of divisions can be made on the missile attack envelope:

a: MAXIMUM ATTACK ENVELOPE

That is the maximum area or boundary that meets the above restrictions. The maximum attack envelope reflects extreme performance of the missile.

b: NON-ESCAPING ENVELOPE

After attacker launches the missile in this zone, the target cannot get rid of the missile attack no matter what kind of maneuver it makes. The kill probability can reach the max.

c: SAFETY ENVELOPE

When the missile encounters a target and explodes, it will create debris and shock waves. The safety envelope protects the attack aircraft from damage. The safety envelope is also minimum attack range.

The results of literature [31] show that the attack envelope is a function of the flight status, launch angle of the missile, target entry angle, the target aircraft's flight status, and so on. These parameters can be used to fit the attack area, and the specific fitting process can be referenced in [3], [32].

2) KILLING ENVELOPE

When an attack aircraft launches a missile within the attackable range, the target aircraft generally adopts a series of overload maneuver to avoid the missile tracking, so to reduce the kill probability We propose four kill envelope based on the different overload maneuver.

a: MAXIMUM KILL ENVELOPE

Within this distance, the target aircraft and the attack aircraft are in the current situation, assuming that the target aircraft does not make any maneuvers throughout the entire process, and the attack aircraft launches missiles to intercept the target. Beyond this distance, the interception fails.

b: 90° TURN KILL ENVELOPE

Within this distance, the attack aircraft launches the missile in the previous state, assuming that the target makes a 90° side-turn maneuver at the end of the missile's flight, and the seeker can intercept the target exactly. Beyond this distance, the target aircraft can use side-turn tactics to get rid of the missile.

c: 180° ROTATION KILL ENVELOPE

Within this distance, the attack aircraft launches the missile according to the current state. During the flight of the missile, firstly, the target aircraft makes a 90° side turn maneuver to guide missile change its tracking trajectory, and another 90° side turn maneuver at the end flight to accelerate the evacuation, consuming missile kinetic energy. During the whole process, the target aircraft completed a 180° turning maneuver. Beyond the envelope, the missile could not threat the target.

d: MINIMUM SAFE EMISSION ENVELOPE

When the missile hits the target aircraft or the fuze bursts, fragments will be throw out. Outside this distance, the fragments will not affect the attack aircraft.

The attack envelope and the kill envelope describe the law of combat from different views. The attack envelope is very intuitive in expressing the conditions for the termination of operations. When the attacking aircraft enters into the non-escape envelope for a period of time, it can be determined and the attacker wins. The kill envelope is more advantageous in describing the situation and threat effect at any time.

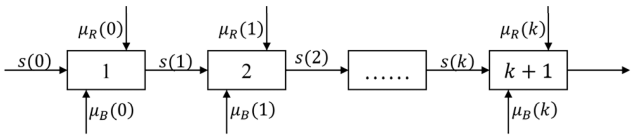


FIGURE 5. Red and blue decision process. $\mu(k)$ represents the maneuver strategy in k step, $s(k)$ is the state which can be calculate by environment model.

III. AIR COMBAT MODEL WITH REINFORCEMENT LEARNING

A. DESCRIPTION OF MANEUVER PLANNING

The core of air combat confrontation is maneuver planning, also called maneuver decision-making, whose task is to generate a series of control commands based on the current state of the two sides and enhance self-advantage. It is a non-linear mapping problem from multiple inputs to multiple outputs. The inputs in each decision step are the state parameters of both sides, such as $s = (x_R, y_R, z_R, v_R, \varphi_R, q_R, x_B, y_B, z_B, v_B, \varphi_B, q_B, r,)$. It can also be described by a combination of basic parameters and nonlinear transformations. The more accurate the situation information, and the easier the decision-making solve. Take one-to-one BVR air combat as an example. The joint decision-making of the two sides makes the operational state change continuously, the process as shown in Fig. 5.

In the step of k, the game strategy of the red and blue is $\mu_k = (\mu_R(k), \mu_B(k))$, which can be descripted by a defined maneuver or a control command. The single-step loss function after executing the strategy μ_k in current state is a non-linear function, $d(s_k, \mu_k, k)$, which indicates the effect of decision-making in the current state. The maneuvering decision problem can be expressed as follows.

$$\begin{cases} J_k(s_k) = \min \{d(s_k, \mu_k, k) + J_{k+1}(s_{k+1})\} \\ s_{k+1} = T(s_k, \mu_k, k) \end{cases} \tag{12}$$

where $J_k(s_k)$ is the overall loss function value from the current state to the end of the game, $T(\cdot)$ is the state transition function. The optimization cost of the decision is: $J^*(s_k) = J(s_k, \mu^*)$, μ^* is the optimal strategy.

Combining with Fig. 5, decision-making problem can be regard as a Markov decision process. Current state of both sides is only related to the last state and actions, and has nothing to do with the states and actions at other moments. Equation (1) is a dynamic programming problem, which can be solved by reinforcement learning.

B. REINFORCEMENT LEARNING

Reinforcement learning (RL) is the method of using trial and error to explore max benefit from environment for agent. The agent will not be told which action to take, but will be offered a profit signal to discover which strategy can generate substantial returns. RL usually expresses in the form of four tuples: $< S, A, P, R >$. The state space S constituted by all the states of the environment. A is the action space which includes

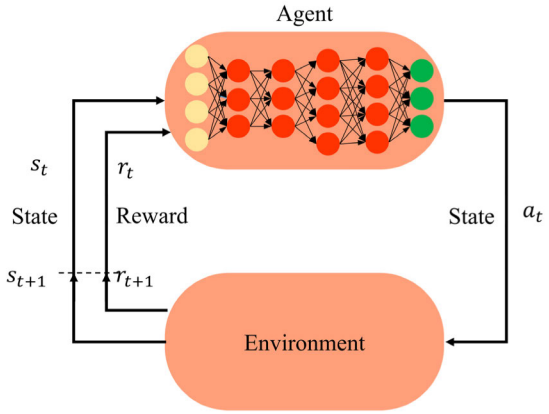


FIGURE 6. The interaction between the agent and the environment. a_t, s_t, r_t respectively represent the action,state and reward at present, s_{t+1} and r_{t+1} are given by environment for the next interaction.

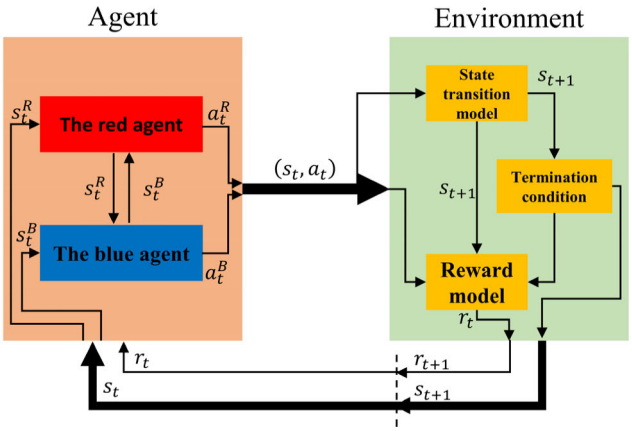


FIGURE 7. Reinforcement learning air combat decision-making framework.

all agent strategy. P represents transition probability. When the agent takes a certain action in the current state with P, the environment transfers current state to another state, and feeds back a reward to the agent. The universal framework for the interaction between agent and environment is shown in Fig. 6.

At the current moment, the agent locates in the state $s_t \in S$, selects an action $a_t \in A$. The environment model moves agent to next state s_{t+1} based on the current state and action, and generates feedback r_{t+1}. The following trajectories can be generated during multiple interactions:

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots$$

So that the cumulative income of the agent from the current moment to the end is:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{13}$$

The goal of reinforcement learning is to maximize the expectation of cumulative income:

$$\pi^* = \arg\max_{\pi} G_0 = \arg\max_{\pi} E_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{14}$$

where π^* refers to the optimal strategy, γ refers to the discount rate, r_t refers to the instant reward at the moment. The state value function is:

$$v_{\pi}(s) = E_{\pi}[G_t | s_t = s] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s] \tag{15}$$

In the optimal strategy π^*, the optimal state value function can be obtained as follows:

$$v^*(s) = \max_{\pi} v_{\pi}(s) \tag{16}$$

The action value function is:

$$\begin{aligned} q_{\pi}(s, a) &= E_{\pi}[G_t | s_t = s, a_t = a] \\ &= E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a] \end{aligned} \tag{17}$$

Under the optimal strategy π^*, the optimal state value function can be obtained:

$$q^*(s) = \max_{\pi} q_{\pi}(s) \tag{18}$$

C. REINFORCEMENT LEARNING AIR COMBAT DECISION-MAKING FRAMEWORK

The one-to-one air combat decision framework mainly carries out the interaction between agents and combat environment, as shown in Fig. 7. Agent represents the red and blue aircraft, whose core role is to make out the current situation of engagement and independently complete the mapping from state to action. The environment consists of many basic models, such as the state transition model, the air combat reward and punishment model, and the termination judgment model. The main work of the environment is to realize the transition from current state to next state according to agent's maneuver strategy. At the same time, the judgment model is used to determine whether the combat is terminated, and the reward model gives the reward value for the current decision.

At time t, the environment gives the basic motion parameters of red and blue agent s_t. The agents respectively construct own primary perceptual state variables s_t^R, s_t^B, and complete the calculation of decision variables a_t^R and a_t^B, then pass back to the environment. The state transition model in the environment calculates the next agent state s_{t+1} based on the current state of the agent and its decision-making actions. The termination judgment model is used to evaluate whether the air combat termination condition is reached. At the same time, the reward model determines punishment value r_{t+1} for the agent based on the agent's current state, action, next moment state, and termination condition judgment result. Finally, the environment will pass s_{t+1} and r_{t+1} to the agent. So far, a cycle of interaction between the agent and the environment is completed.

There are three basic elements in the air combat framework: agent state variables, agent actions, and environmental rewards. The agent state is different from the basic state variables which is basic motion variable and given by the

TABLE 1. Air combat agent state variables.

Variables	SYMBOL	Element
Independent State	IS_1	$\mathrm{P}_B=(x_B,y_B,z_B)$
	IS_2	$\mathrm{P}_R=(x_R,y_R,z_R)$
	IS_3	$\mathbf{v}_R=(v_{R1}\cos(\gamma_R)\cos(\psi_R), v_{R1}\sin(\gamma_R), v_{R1}\cos(\gamma_R)\sin(\psi_R))$
	IS_4	$\mathbf{v}_B=(v_{B1}\cos(\gamma_B)\cos(\psi_B), v_{B1}\sin(\gamma_B), v_{B1}\cos(\gamma_B)\sin(\psi_B))$
Relative State	RS_1	$r=(\Delta x,\Delta y,\Delta z)$
	RS_2	$\Delta v=(v_{Rx}-v_{Bx}, v_{Ry}-v_{By}, v_{Rz}-v_{Bz})$
	RS_3	$\phi=(\cos(\varphi_B),\cos(q_B),\cos(\eta_{RB}))$
Energy State	ES_1	$E_V=(v_B^2, v_R^2, v_B^2-v_R^2)$
	ES_2	$E_H=(h_B^2, h_R^2, h_B^2-h_R^2)$
	ES_3	$E_M=(r, h_B, h_R)$

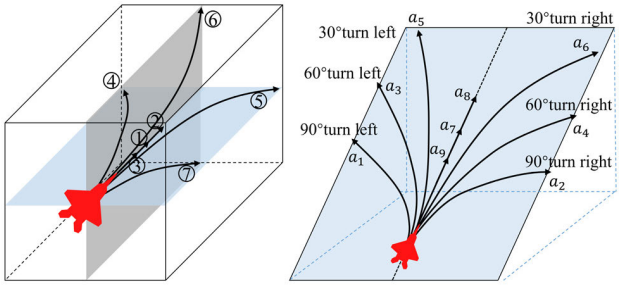
environment directly. The agent state can include basic motion variables but is not limited to them. Other complex variables should be rebuilt for agent to perceive the battlefield situation and make maneuvering decisions. Agent action space can be traditional tactical actions, or basic control actions, but they must be converted into flight control command and passed to the environment. Environmental reward model is an important part of the confrontation framework, and is also a key factor for the agent to accurately obtain the winning regulation.

1) STATE SPACE

We use three parts to represent agent state, independent state, relative state and energy state. The independent state consists of independent motion variables of both parties, such as position and velocity. The relative state makes up of relative motion variables, such as relative distance, relative speed, and relative angle. The energy state is mainly represented by the parameters that are positively related to the kinetic energy, potential energy and the parameters that affect the missile's killing range, Such as H^2 and v^2 (H is height, v is speed). The specific representation of the agent state is shown in Table 1.

So the agent state can be defines as $s_t = (IS_1, IS_2, IS_3, IS_4, RS_1, RS_2, RS_3, ES_1, ES_2, ES_3)$, each element is 1×3 vector. Obviously, the physical meanings of the 10 state variables are different, and there are different dimensions and value ranges, so the preprocessing of the variables is necessary. We use the min-max standardization method to normalize and map them to the space [0, 1]. The normalization formula is as follows:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{19}$$



(a) Basic maneuvering maneuvers (b) Extended maneuvering maneuvers

FIGURE 8. Agent maneuvering maneuvers. Symbols in (a) is that: ① continued stable flight, ② maximum accelerated flight, ③ maximum decelerated flight, ④ turning left with maximum overload, ⑤ turning right with maximum overload, ⑥ climb with maximum overload, ⑦ dive with maximum overload.

TABLE 2. Air combat maneuver.

Symbol	Variation	Maneuver type	Symbol
$\lvert\Delta\psi\rvert$=0	$\lvert\Delta\gamma\rvert$=90°	Turn left 90°	a_1
		Turn right 90°	a_2
	$\lvert\Delta\gamma\rvert$=50°	Turn left 50°	a_3
		Turn right 50°	a_4
	$\lvert\Delta\gamma\rvert$=30°	Turn left 30°	a_5
		Turn right 90°	a_6
	$\lvert\Delta\gamma\rvert$=0°	Keep straight	a_7
		Maximum acceleration straight	a_8
		Maximum deceleration straight	a_9
$\lvert\Delta\gamma\rvert$=0°	$\Delta\psi=\Delta\psi_{\max}$	Maximum overload climb	a_{10}
	$\Delta\psi=\Delta\psi_{\max}$	Maximum overload dive	a_{11}

2) ACTION SPACE

The agent actions can be divided into continuous actions and discrete actions. The continuous actions are mainly composed of flight control commands, and each control variable is continuous in its constraint space. Discrete actions need to set up a library of tactical actions in advance. Each one can be decomposed into a combination of basic actions and completed with its corresponding control instructions. NASA researchers divide the maneuvers into 7 basic maneuvering maneuvers, as shown in Fig. 8(a).

These 7 types of actions are the ultimate actions of the aircraft in various states. This will make the trajectory of two adjacent decision points not smooth and the instruction is difficult to execute. Therefore, we expand the basic actions. Taking a 45° pitch angle as an example, the left and right turns are refined separately, as shown in Fig. 8(b). According to the change of the heading angle after the turn, it can be divided into 30°turn left, 50°turn left, 90°turn left, 30°turn right, 50°turn right and 90°turn right. The specific action division is shown in Table 2.

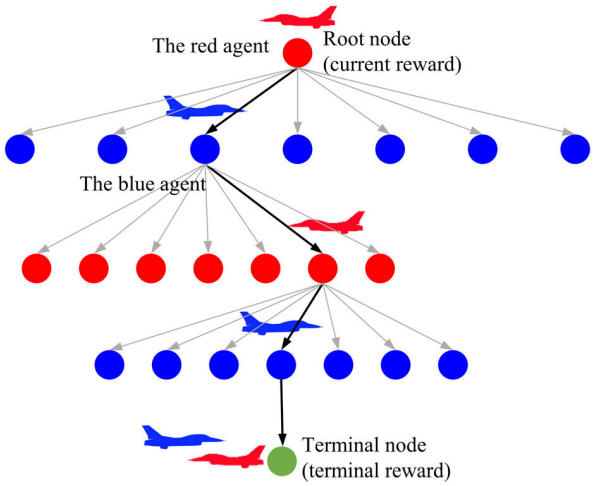


FIGURE 9. Monte Carlo tree search for the reward of root node.

3) REWARD AND PUNISHMENT

The reward and punishment not only needs to consider the effect of agent current situation, but also the future impact in the next few steps caused by the current both sides actions. However, in the process of fighting and gaming, the strategies of both sides are uncertain, and it is difficult to determine the exact impact of the current situation directly. Monte Carlo search is a good method to deal with the difficulty. The simulation experiments begin with a root node which is formed by the agent current state. Each sub-node in the same hierarchy represents different state of the same agent. A complete search path in the tree represents a game, as shown in Fig. 9. The statistical results of the experiment are used to approximate the reward and punishment value of the root node.

When the downward simulation search meets the air battle winning condition or reaches the maximum number of decision step, it is determined as the terminated node, and then the value on each node is calculated retrospectively. The cumulative reward of the initial node is mainly composed of the current state reward and statistical reward from the search tree and terminated node. The current state reward can be divided into four parts, boundary reward R_{Border}, angle reward $R_{Azimuth}$, distance reward R_D and missile attack reward $R_{Missile}$ Boundary reward. Boundary reward gives penalties to agents that exceed the pre-set air battlefield boundary. The reward value is 2 when the agent is within the boundary, and the reward value is -100 when the agent out of the boundary. While agent approaches the boundary, the reward value is determined by the result of Monte Carlo tree search, as shown in Fig. 10(a). In the first search, the agent approaches to the boundary with v_1 speed. When the second step is reached, the agent's speed direction and size may change to v_2 or v_2', which depends on the agent action taken during the search. After the second step, if the agent is out of bounds, the search ends, and if not, the search continues in the same way. At the end of N steps, all the search results are counted to calculate out of bound probability P_{out}. The return value of the current state is as follows:

$$R_{Border} = \begin{cases} 2(1 - P_{out}) - 100P_{out}, \\ \qquad (current\ state\ in\ the\ area) \\ -100, \quad (current\ state\ out\ the\ area) \end{cases} \tag{20}$$

Angle reward. To prevent agents from being in a non- operational state for a long time, angle reward is used to encourage both sides to play the game against each other. We set the value according to the relative angle of the two agents, as shown in Fig. 10(b). Azimuth angle and entry angle are discrete to 4 class, $[-\pi, -\frac{1}{2}\pi]$, $[-\frac{1}{2}\pi, 0]$, $[0, \frac{1}{2}\pi]$, $[\frac{1}{2}\pi, \pi]$. The min penalty is -5 when the two parties are not at war (such as azimuth angle $= \pi$, entry angle $= 0$), and the max reward is 5 when the two are fighting(such as Azimuth angle $= 0$, entry angle $= \pi$). The reward changes with the relative angle of two sides. It should be noted that this penalty cannot conflict with the agent's tactical evasion strategy.

Distance reward. In the same way, to prevent agents from moving further and further apart in the vast battlefield space, which makes the decision meaningless, we put forward corresponding constraints on the distance between the two. As shown in Fig. 10(c), The x-coordinate represents the distance and the y-coordinate represents the return value. D_1 is less than the minimum attack distance of the missile, and D_2 is greater than the maximum attack distance of the missile. The maximum reward value is 5, and the maximum penalty value is -5. This penalty item does not affect the return value of the missile attack.

Missile attack reward. The punishment of the attack zone is the core of the reward model. It reflects whether the agent can destroy the target by launching missile at the current moment. As shown in Fig. 11, the red agent launches a long-range air-to-air missile, and the blue agent will use different maneuver to evade. There are three common choices, no maneuver at all, 90° maneuvering, and 180 maneuvering. We set the probabilities of the choices as 0.25, 0.5, and 0.25. Then, the missile model is used for simulation calculation to determine whether the launched missile can hit the target. Through multiple simulation experiments, the final hit probability $P_{encounter}$ is calculated with statistical result. At the same time the probability $P_{destroyed}$ that the agent is hit by an enemy missile can also be obtained. Finally, missile attack reward is:

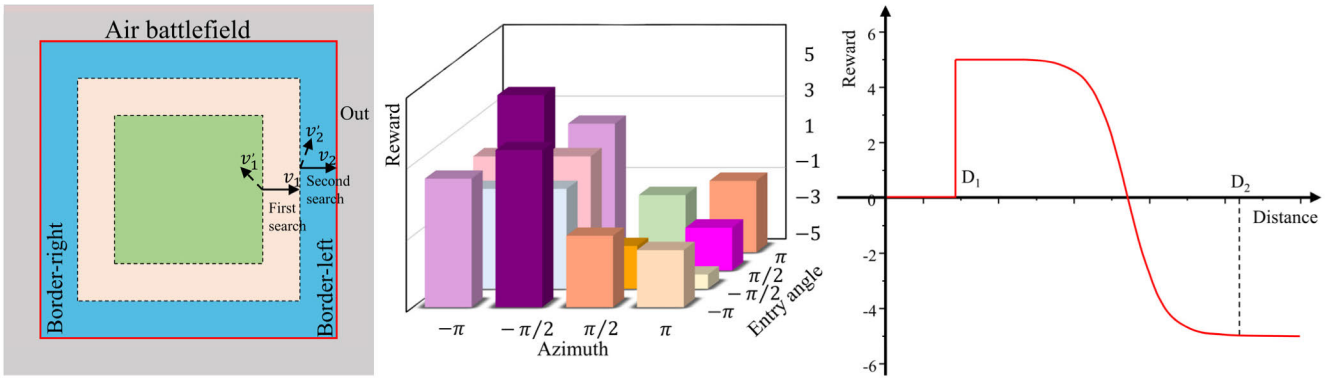
$$R_{Missile} = 100P_{encounter} - 200P_{destroyed} \tag{21}$$

IV. LSTM-DQN ALGORITHM

Based on the agent decision framework in Fig. 7, in this section, we mainly introduce the deep network and training algorithm for agent.

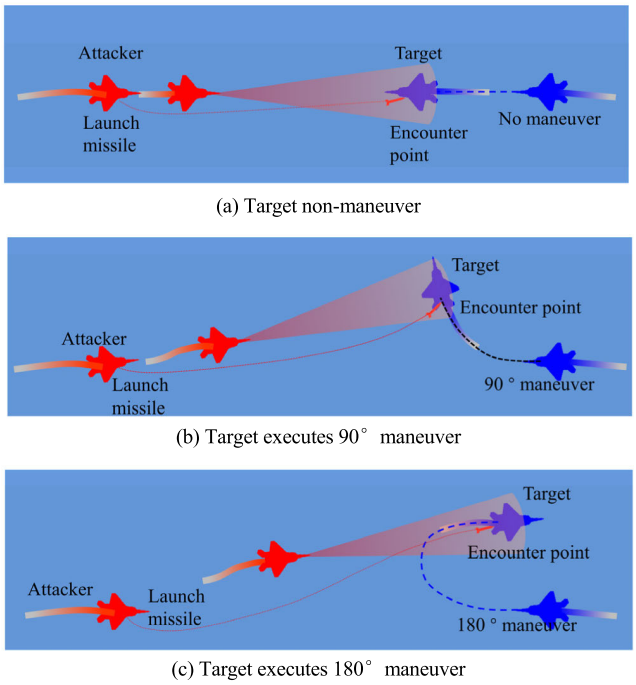
A. ALGORITHMIC REPRESENTATION

As mentioned in Section III B, we use action value function as the criterion of strategy evaluation and maneuvering



(a) The search for boundary reward (b) The angle reward for agent (c) The distance reward for agent

FIGURE 10. The design of reward and punishment for agent.



(a) Target non-maneuver

(b) Target executes 90° maneuver

(c) Target executes 180° maneuver

FIGURE 11. The missile attack reward for agent.

action choice. From (17), we can obtain:

$$Q_\pi(s,a) = E_\pi[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s=s_t, a=a_t] = \frac{1}{m}[\sum_{i=1}^{m}\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s=s_t, a=a_t] \quad (22)$$

Different policy π can make great differences for each maneuvering strategy $\mu_k = (\mu_R(k), \mu_B(k))$, so as to create an influence on global target in (12). We can parameterize the policy and optimize it by learning the parameters θ_t.

$$Q^\pi(s_t, a_t) = Q(s_t, a_t, \theta_t) \quad (23)$$

The depth policy network and the depth target network are respectively constructed to approximate the action-value function, as shown in Figure 12.

For the policy network:

$$Q_{Policy} = Q(s_t, a_t, \theta_t) \quad (24)$$

The policy network is used to choose an action for agent with $\varepsilon - greed$ strategy.

$$\pi(a_t|s_t) = \begin{cases} 1-\varepsilon+\varepsilon/N & if\ a_t = \arg\max_a Q(s_t, a_t, \theta_t) \\ \varepsilon/N & if\ a_t \neq \arg\max_a Q(s_t, a_t, \theta_t) \end{cases} \quad (25)$$

For the target network, the core function is to form network loss.

$$Q_{Target} = r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}, \theta_{t+1}) \quad (26)$$

The loss can be represented as:

$$\begin{aligned} \mathrm{L}(\theta_t) &= E[(Q_{Target} - Q_{Policy})^2] \\ &= E[(r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}, \theta_{t+1}) \\ &\quad - Q(s_t, a_t, \theta_t))^2] \end{aligned} \quad (27)$$

The gradient is

$$\frac{\partial L(\theta_t)}{\partial \theta_t} = E[(Q_{Target} - Q(s_t, a_t, \theta_t))\frac{\partial Q(s_t, a_t, \theta_t)}{\partial \theta_t}] \quad (28)$$

In traditional deep reinforcement learning, most researches use full connection as basic deep network cell to approximate value functions. For simple state space problems, this method can be valid. However, the state space in this paper is complex and time-series, so LSTM cell is a best choice to construct two deep networks. For each LSTM cell, it can be divided into forget gate, input gate and output gate. The input data of each cell includes two parts, one is the input of the upper layer, representing the further extraction for state characteristics, and the other is the input of the same layer, representing the sharing of timing information.

The forget gate:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (29)$$

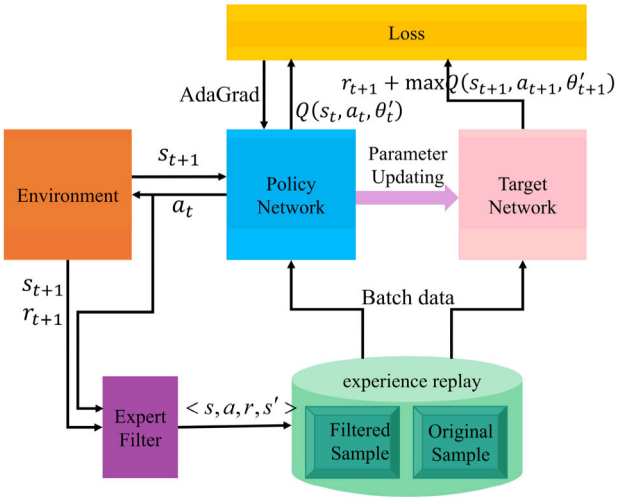


FIGURE 12. The confrontation framework for agent training.

The input gate:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{30}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{31}$$

The output gate:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{32}$$

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{33}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{34}$$

where $\sigma(x) = 1/1 + e^{-x}$, $W.$, $b.$ is network weight and bias.

B. TRAINING ALGORITHM

According to the representation of DQN, we design the agent training structure in Fig. 12. The agent contains two networks with the same structure, a policy network and a value network. The policy network is mainly used to calculate the current actions of the agent, and the target network participates in the construction of training loss. Training samples are obtained from the experience pool whose dates come from environment. Adaptive gradient descent method is used to train the policy network parameters, and after a certain number of training episodes, the parameters are applied to update the target network. The specific process of the algorithm is shown in Algorithm 1.

During training process, agents are prone to generate large numbers of worthless samples. In view of this matter, we design an expert filter based on the experience of BVR to judge and select preliminarily. The filtered samples and original samples are integrated with a certain proportion into the experience playback pool for training. So that we can not only speed up the convergence of agent directionally, but also make the agent have generalization ability.

C. DESIGN OF POLICY NETWORK

As mentioned in the previous section, the structure of the value network and the target network are the same,

Algorithm 1 Agent Training

Initialize replay memory
Build policy network with random weights
Build target network with random weights
For episode = 1, M do
  Chose an initial state s_0
  For step = 1, N do
    Select an action a_t with $\epsilon-$greedy policy
    Otherwise $a_t = \max_a Q^*(s_t, a, \theta)$
    Interact with the environment and get next state s_{t+1} and reward r_{t+1}
    Filter interactive data and store (s_t, a_t, r_t, s_{t+1}) into reply memory
    For k=1, K do
      Sample random minibatch data to policy network
      Training network parameters
    End for
    Every C steps, update target network with policy network parameters
  End for
  Environment reset
End for

but the parameters are different. We propose a deep policy network with two layer, perceptive situation layer and value function fitting layer. The first layer is used to extract the high-dimensional features from the state parameters, and output perception information. The other is to fit the state value and complete the action choice for agent. These two parts can also be regarded as a whole.

1) PERCEPTIONAL SITUATION LAYER

Perceptional situation layer is a multi-layered LSTM network with a continuous sequence of inputs. For each sample data from the memory, we use the environment model to sample T continuous state data which forms an input sequence. As shown in Fig. 13, the current state in k decision step is s_t, we extract last 5 state variables with flight motion model, the time interval between every two adjacent states is 1 second. So that, the input data consists of $(s_{t-5}, s_{t-4}, s_{t-3}, s_{t-2}, s_{t-1}, s_t)$, and $\Delta t = 1\,s$.

As mentioned in Section 3, the agent state is 10×3, in combination with all sequence, the feature structure of the input layer is $T \times 10 \times 3$. The weight and basis of the i layer are respectively W^i, b^i. Each LSTM cell takes input h_{i-1}, c_{i-1} from adjacent cells in the same layer and takes the result of that cell as input for the next LSTM layer. At the end of perceptional situation layer, the input state variables are extract to a higher-dimensional abstract features which contain perceptive information. Then all the data is expanded into a one-dimensional array ($f_{end} \times 1$) used as the input for next fitting network.

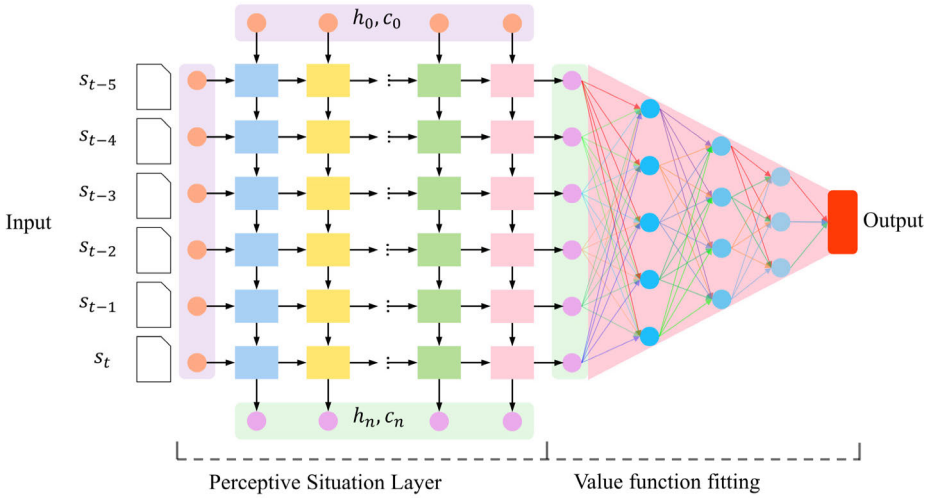


FIGURE 13. The agent policy network with perceptive situation layer and value function fitting layer.

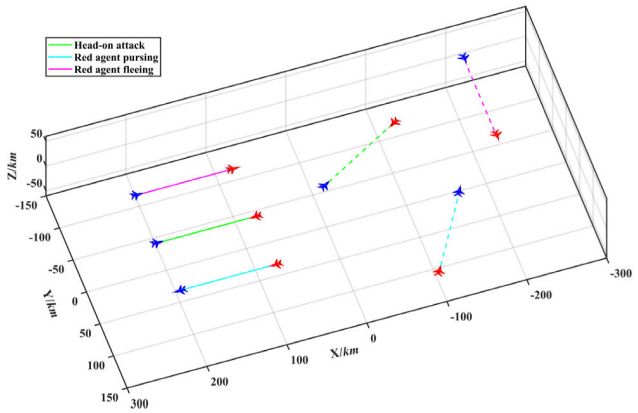


FIGURE 14. The initial relative position for both sides in training.

2) VALUE FUNCTION FITTING LAYER

The fitting layer is the combination of multiple full connection layers. The input is high dimensional characteristic data from the perceptional situation layer. The output of the network is the value of actions. As designed in Section 3, the action space includes 9 tactical actions. So the output of the policy network is a 9×1 vector represents the value of each action in current state. The number of neurons is determined layer by layer.

V. AIR COMBAT SIMULATION

In this section, we train the value network of the agent, and use the results of each training stage to test the agent's decision-making ability. According to the relationship between the positions of the two sides in the initial state, we set three scenarios, namely, the red agent pursing, the red agent heading on the blue side, and the red agent fleeing. To simplify the difficulty of network training, we assume that the blue side keeps the current strategy unchanged during the confrontation. In each scenario, we set the initial agent speed 280 m/s, the maximum agent speed 400 m/s, and the minimum flight speed 200 m/s.

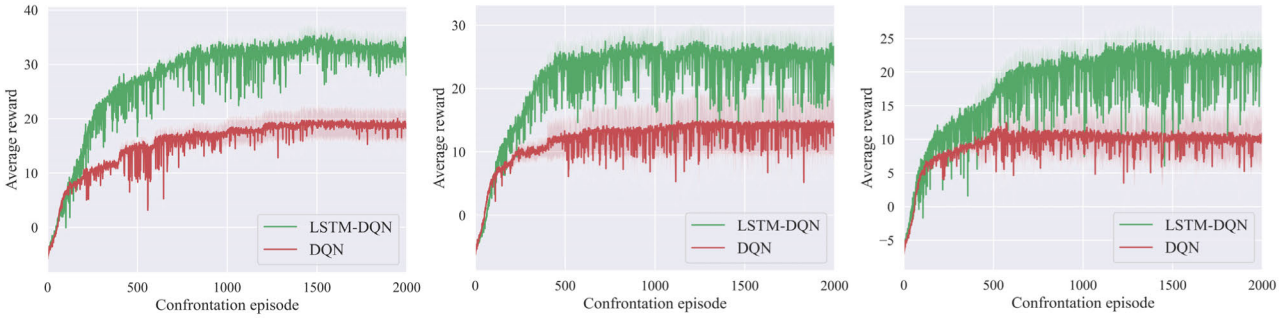
A. SCENARIO DESIGN

In the confrontation simulation, the size of the air battle field is set as $600 \times 300 \times 120km$, the origin is the center of the battlefield. The early initial state and position in the three scenarios is shown in the figure. The red and blue sides are at the same level, 120km apart, and both sides have a speed of 280m/s. In the red agent pursing scenario, the initial position of the red side is (100, 75, 0), the initial position of the blue side is (220, 75, 0), and the velocity direction of the red side is facing the flight direction of the blue side. In the red agent heading on the blue side scenario, the initial position of the red side is (100, 0, 0), the initial state of the blue side is (220, 0, 0), the velocity direction of the red side is head to the blue side, and the velocity direction of the blue side is head to the red side. In the scenario of red agent fleeing, the initial state of the red side is $(100, -75, 0)$, and the initial state of the blue side is $(220, -75, 0)$, the red side is followed by the blue side, and the blue side's velocity direction is positive to the red side.

It should be noted that during the training of each scene, the initial position should be changed randomly every 15 rounds of engagement. If the same initial position is used for each network training, the parameters will become too dependent on the initial value and cannot be generalized at other locations. During the change, the relative positions, relative states and velocities of the two sides of red and blue are kept unchanged. The same translation transformation for the two side is only performed on the three-dimensional coordinates or the rotation transformation is carried out in the *XOY* plane, as shown in the dashed line in the Fig. 14.

B. AGENT TRAINING AND TESTING

In each scene, the red and blue sides fight for 30,000 rounds, and the average reward value is recorded every 15 rounds.



(a) The red agent pursing scenario (b) The red agent head-on attack scenario (c) The red agent fleeing scenario

FIGURE 15. Average reward with different network structure in three scenarios.

The maximum number of decision steps in each round is 20, and the round is terminated when the termination judgment conditions are satisfied or the maximum number of decision steps is reached. The results of the interaction between the agent and the environment are added to the experience pool as playback data for the next network training. The average reward for each round during the training is shown in Fig. 15.

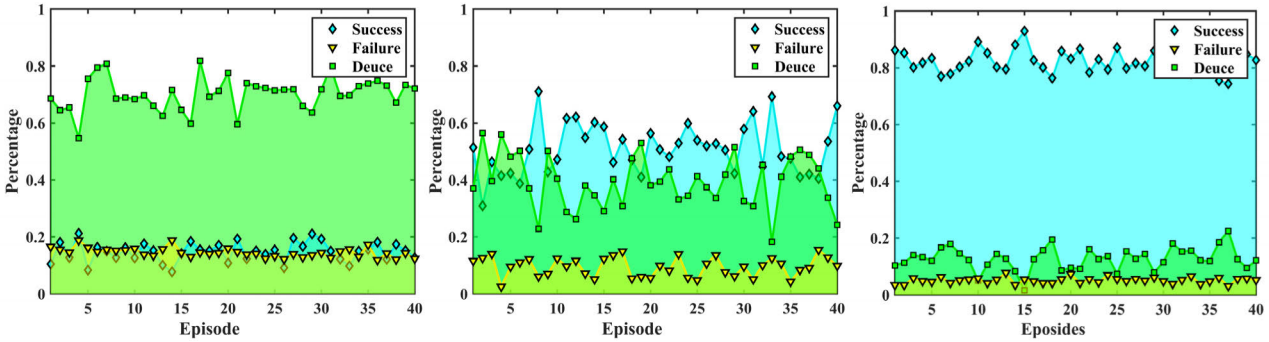
The experimental results in the Fig. 15 show that, under the three scenarios, the LSTM-DQN network proposed in the paper can generally get higher scores than the traditional QDN network. It proves that the proposed policy network can make agent obtain more profit from the environment and the perceptive situation layer is valid. Judging from each result of the three scenarios, the final average reward of the first scenario in Fig. 15(a) fluctuates around 32 within a narrow range. In the second scenario as shown in Fig. 15(b), the final average reward is not higher than the first scenario and varies slightly around 26.5. It indicates that the advantage of the red initial state in second scenario is weaker than that in first scenario, and the task is hard to control. From the result of the third scenario in Fig. 15(c), the red side converges slowly, the final average reward only reach 22.6, and the fluctuations are obvious. From the comparison of the scores between the three scenes, the difficulty of the agent mission is increasing sequentially.

In the early, middle and late stages of the training, 40 network models recorded during the process are randomly selected respectively to test. Each model is tested for 100 rounds of engagement and the wining round numbers, losing round number are counted. So that the probability of winning and losing can be approximately replaced by the frequency. The test probability of three scenarios are shown in Fig. 16.

Fig. 16 shows the result of the first scenario. In the early stage of the training, the two sides are mainly tied, whose probability is very high in Fig. 16(a). We can draw a conclusion that the red agent is in a random maneuvering state and failed to make good use of the advantage of the initial state. In the middle stage of the training, the percentage of success has gone up, while the one of deuce is still high, as shown in Fig. 16(b). It confirms that the agent begins to master the initial superiority of relative position and the meaningless maneuver is gradually reducing. Also, it proves that the agent can take advantage to win the game. In the later stage of training, as the result in Fig. 16(c), the agent can win with a high probability, and the probability of failure or draw reduces to the lowest. We record and reply a confrontation trajectory in Fig. 19. There are already no random or worthless actions for trained agent. The advantage situation is always maintained until missile launch conditions are met.

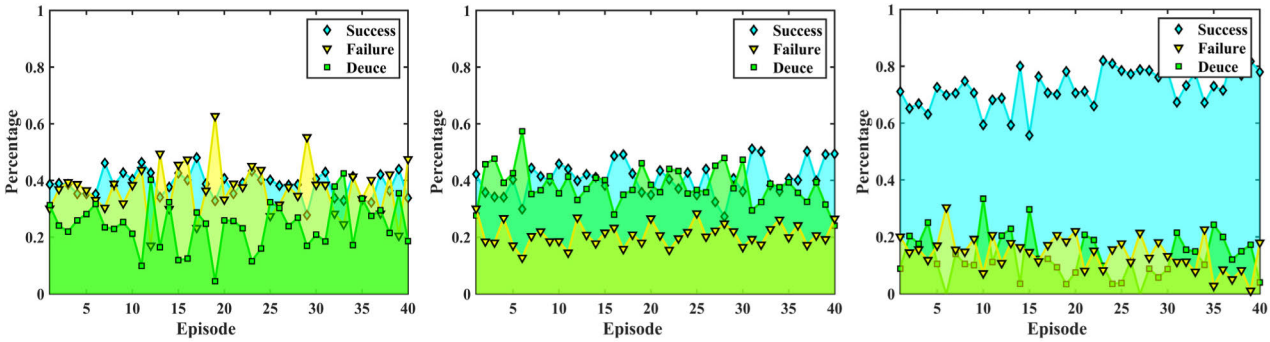
Fig. 17 is the training result in red and blue head-on attack scenario. Considering the initial state, both sides situations are equal. The probability of success is as high as the probability of failure in precious training in Fig. 17(a). From a training trajectory in Fig. 20(a), the agent chooses with no purpose and always under the missile attack zone of the target. After previous training, as shown in Fig. 17(b), the number of failure decreases sharply. The agent gradually masters the turning maneuvers to avoid being attacked. However, it is also apparent that the percentage of deuce also increases, which can be explained in Fig. 20(b). The red agent turns left to escape from the missile of the blue agent, while it also loss the wining chance of itself. The combat result has improved considerably in later stage of training. As shown in Fig. 17(c) and Fig. 20(c), the agent can transform into attacking the red side after avoiding the attack area, transforming disadvantages into advantages, and increase the probability of winning.

During the training of red agent fleeing scenario in Fig. 18, the red has a disadvantage in initial state and is hard to get away from the pursuit of blue agent, as shown in Fig. 21(a). The percentage of failure is very high compared with other scenarios. The middle training result in Fig. 18(b) shows that the probability of survival for agent is 0.3. Climb and dive maneuver are firstly learnt to preserve itself. It is universally acknowledged that climbing or diving is a good solution when pursing by others in combat. Effectively, the network can learn the experience, which can be replied in Fig. 21(b). The red escape from the blue and gather own energy simultaneously. In the later stage of training, the red agent can effectively avoid the blue attack area and gradually turn the



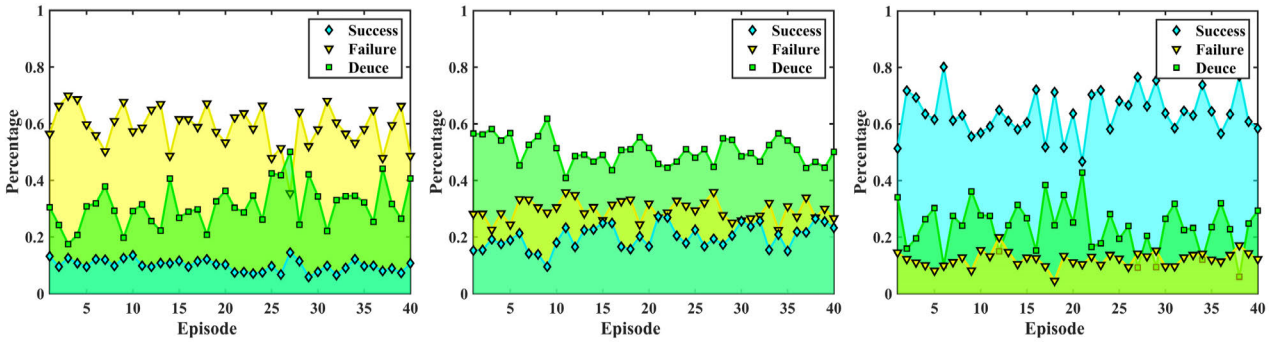
(a) Early training results in red agent pursing scenario (b) Middle training results in red agent pursing scenario (c) Late training results in red agent pursing scenario

FIGURE 16. The test result during different training stage in red agent pursing scenario.



(a) Early training results in red agent head-on attack scenario (b) Middle training results in red agent head-on attack scenario (c) Late training results in red agent head-on attack scenario

FIGURE 17. The test result during different training stage in red agent head-on attack scenario.



(a) Early training results in red agent fleeing scenario (b) Middle training results in red agent fleeing scenario (c) Late training results in red agent fleeing scenario

FIGURE 18. The test result during different training stage in red agent fleeing scenario.

disadvantage into advantage. The percentage of success is more than that of failure and deuce in Fig. 18(c). The height energy is used to attack blue sides in Fig. 21(c).

VI. COMPARISON AND DISCUSSION

In this paper, we propose an LSTM-DQN algorithm and a deep network to solve the BVR maneuver planning problem. In this section, we mainly compare and discuss the effectiveness and complexity of the algorithm. The target of the maneuver planning is to provide a reasonable action plan for air combat aircrafts, so that agent can accumulate situational advantages from the current status to launch air-to-air missiles, and effectively attack enemy targets. The essence of maneuver planning is to give the action strategy, and its effectiveness is reflected on the result of combat and the real-time performance of calculation. Through comparing

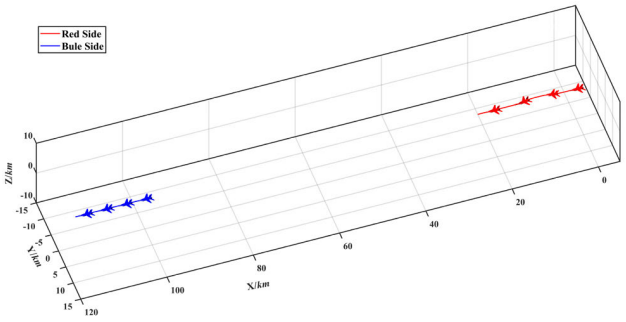
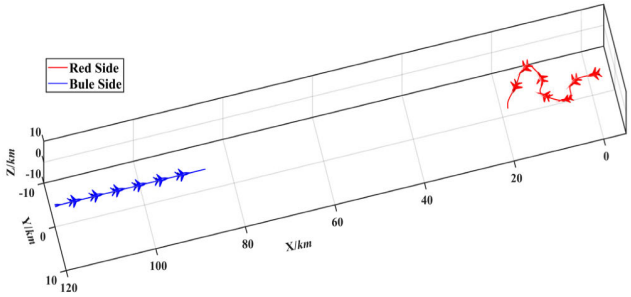
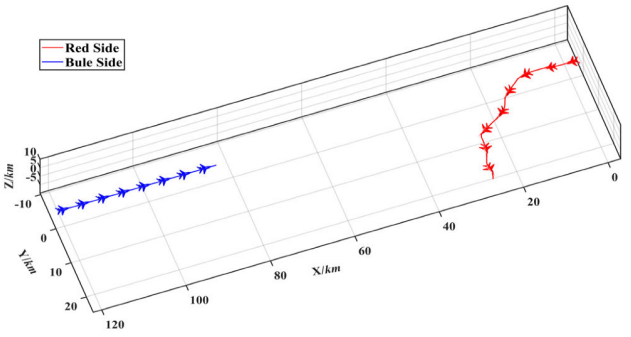


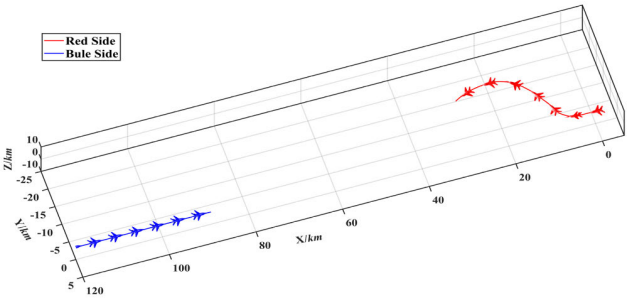
FIGURE 19. One confrontation trajectory after training in first scenario.



(a) Confrontation trajectory during early stage of training



(b) Confrontation trajectory during middle stage of training



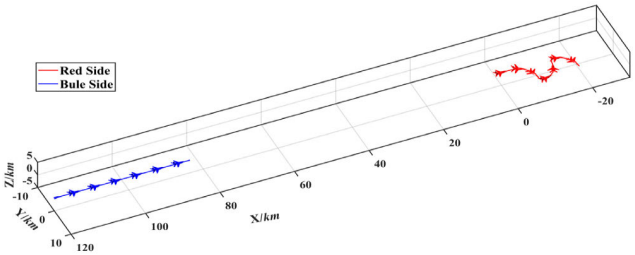
(c) Confrontation trajectory during later stage of training

FIGURE 20. Confrontation trajectories in second scenario.

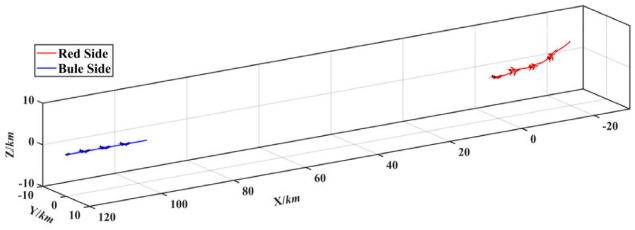
our methods with others, the effectiveness and timeliness of the algorithm can be analyzed.

A. EFFECTIVENESS COMPARISON

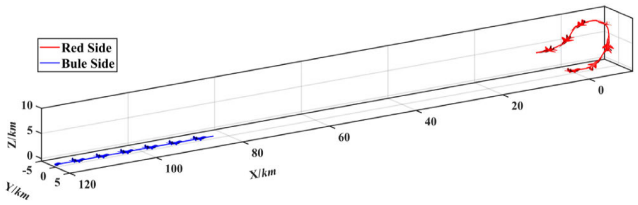
The effectiveness of the planning solutions is most directly reflected in the change of the agent's situation and the result of the attack on the enemy. There are two criterions to measure. The first one is reward which is obtained from the



(a) Confrontation trajectory during early stage of training



(b) Confrontation trajectory during middle stage of training



(c) Confrontation trajectory during later stage of training

FIGURE 21. Confrontation trajectories in second scenario.

environment model. Reward is the evaluation of the agent's current status and maneuvering actions, reflecting the relative advantage with the opponent. The other is winning probability, which can be counted from results of testing experiments. We do some comparative experiments with other trained methods to explain the effectiveness of our methods.

In section V, we propose three scenarios. The first scenario is red agent pursing. The second scenario is red agent head-on attack. The third scenario is red agent fleeing. We compare our method with AC [28] and DDPG [29] in the three scenarios. During training, three methods use same environment models and reward models proposed in the paper. The difference is the decision algorithm for agent during choosing actions. After 30000 rounds training, we make 200 rounds test in three scenarios. The statistical result of reward is shown in Table 3, and the winning probability in Table 4.

As can be seen from Table 3 and Table 4, for Scenario 1, the mission is relative simple and the winning probability is relatively close between three methods. The average reward value of the methods stabilizes at high levels, which is 25.86, 25.14 and 29.73 respectively. However, as the maneuvering mission becomes more difficult, for example, in Scenario 2 and 3, the algorithm AC and DDPG can't achieve high winning probabilities. The effectiveness and decision-making ability of LSTM-DQN is superior to others.

In Scenario 2, it can be proved from Table 2 that AC and DDPG have weak ability to accumulate situation. The reward obtained from the environment is smaller than the method

TABLE 3. Testing Reward with different methods.

Scenario	Method	The min reward	Average reward	Variance
The first scenario	AC	-9.58	25.86	1.86
	DDPG	-8.56	25.14	1.53
	LSTM-DQN	-8.26	29.73	1.25
The second scenario	AC	-11.72	14.46	3.78
	DDPG	-10.08	15.22	3.26
	LSTM-DQN	-9.86	20.38	3.04
The third scenario	AC	-13.48	8.81	3.12
	DDPG	-12.56	10.32	3.15
	LSTM-DQN	-11.85	15.68	3.76

TABLE 4. Testing results with different methods.

Scenario	Method	Probability		
		Win	Deuce	Loss
The first scenario	AC	0.86	0.13	0.01
	DDPG	0.87	0.12	0.01
	LSTM-DQN	0.88	0.11	0.01
The second scenario	AC	0.69	0.16	0.15
	DDPG	0.70	0.16	0.14
	LSTM-DQN	0.79	0.13	0.08
The third scenario	AC	0.52	0.27	0.21
	DDPG	0.53	0.27	0.20
	LSTM-DQN	0.71	0.19	0.10

proposed in this paper, and the variance is larger, AC and DDPG algorithm is unstable and fluctuates greatly.

In Scenario 3, the loss probability with AC and DDPG is up to 0.21 and 0.20. Such a high probability of failure is difficult to accept in air combat. Although the variance of the reward of these two algorithms is smaller than LSTM-DQN, the main reason is the overall reward value obtained by this algorithm is relatively small.

B. COMPLEXITY ANALYSIS

There is little research on the complexity of deep reinforcement learning, the main reason is that the convergence is difficult to prove theoretically. In this section we briefly analyze the computational complexity of LSTM-DQN in the view of the algorithm framework.

Considering with the confrontation framework in Fig.12, we can divide the algorithm into three parts, action selection, state transition and network training. The 7th and 8th lines of Algorithm 1 are used for action selection, including the calculation of a network and the sorting of value functions $\arg\max_a Q(s_t, a_t, \theta_t)$. Assuming the complexity of network calculating forward is $O(lstm)$. So the action selection complexity can be represented as $O(lstm) + O(n)$, where n is the number of actions. The 9th line in Algorithm 1 is mainly for state transition. We set the time interval between two adjacent decision steps as ΔT, and the simulation step time as 0.1s. According to the differential equation mentioned in Section II, we use the Rungo-Kutta method to solve the equation. So the time complexity of state transition can be expressed as $e = O(\Delta T/0.1 \cdot k^3)$. From the 13th line to 16th line, the deep network is training with interactive data sampling from replay experience. The computational complexity of network training can be represented as $net = O(E \cdot D/B \cdot L \cdot n_f)$. E is the training epochs. D is the size of sample data, which is also the scale of replay experience. B is batch size for iterative training. L is the complexity of each layer of network and n_f is layer number. Comparing the three sections, $O(lstm) + O(n)$ can be ignored due to n is little in this paper. Therefore, the complexity of Algorithm 1 is $O(M \cdot N \cdot e + M \cdot N \cdot net)$.

When using the method to solve the maneuvering planning problem, we mainly adopt offline learning and online decision making. Offline learning is used to trained deep network with database, so as to reduce computational time during online decision process which does not need network training again. The consumption of time in the actual use of the model is mainly reflected in online decision making, as $O(lstm)$. The time cost of network calculation can be reduced by controlling the number of parameters of the deep network. The calculation of the Q value in the paper is controlled within 0.01s, which can meet the requirements of the simulation step.

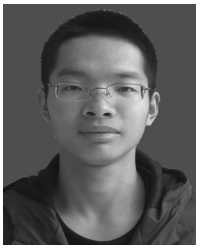
VII. CONCLUSION

In this study, we built a BVR flight motion simulation model and air combat confrontation framework. The proposed LSTM-QDN is capable of situational awareness and action decision making which is the key in maneuver planning. We design a continuous state space and 9 actions space for the combat agent and a detailed reward model is used for agent training. The experiment shows that the reward model and the network structure in the paper can make the training converge effectively, and the test of three scenarios demonstrates and effectiveness and the feasibility of intelligent confrontation with our method.

REFERENCES

[1] X. An, C. Xing, L. I. Zhanwu, and H. U. Xiaodong, "A method of situation assessment for beyond-visual-range air combat based on tactical attack area," *Fire Control Command Control*, vol. 45, no. 9, pp. 97–102, 2020.
[2] H. U. Zhaohui, L. Y. U. Yue, and X. U. An, "A threat assessment method for beyond-visual-range air combat based on situation prediction," *Electron. Opt. Control*, vol. 27, no. 3, pp. 8–12, 2020.
[3] W. H. Wu, S. Y. Zhou, L. Gao, and J. T. Liu, "Improvements of situation assessment for beyond-visual-range air combat based on missile launching envelope analysis," *Syst. Eng. Electron.*, vol. 33, no. 12, pp. 2679–2685, 2011.
[4] H. Luo, "Target detection method in short coherent integration time for sky wave over-the-horizon radar," *Sādhanā*, vol. 45, no. 1, pp. 1–9, Dec. 2020.
[5] T. Liu and R.-W. Mei, *Over-the-Horizon Radar Impulsive Interference Detection With Pseudo-MUSIC Algorithm*. Shanghai, China, 2019, p. 7.
[6] H. U. Dongyuan, Y. Rennong, Y. A. N. Mengda, Y. U. E. Longfei, Z. U. O. Jialiang, and W. Ying, "Real-time calculation of missile launch envelope based on auto-encoder network," *Acta Aeronautica et Astronautica Sinica*, vol. 41, no. 4, pp. 231–247, 2020.

[7] Y. A. N. Mengda, Y. Rennong, Z. U. O. Jialiang, H. U. Dongyuan, Y. U. E. Longfei, and Z. Yu, "Real-time computing of sir-to-air missile multiple capture zones based on deep learning," *Acta Armamentarii*, to be published.
[8] S.-J. Han, "Analysis of relative combat power with expert system," *J. Digit. Converg.*, vol. 14, no. 6, pp. 143–150, Jun. 2016.
[9] Z. Gacovski and S. Deskovski, "Modelling of combat actions via fuzzy expert system," Tech. Rep., 2020.
[10] K. Zhou, R. Wei, Z. Xu, and Q. Zhang, "A brain like air combat learning system inspired by human learning mechanism," in *Proc. IEEE CSAA Guid., Navigat. Control Conf. (GNCC)*, Aug. 2018, pp. 1–6.
[11] M. Guang-lei, L. Yuan-Qiang, L. Xiao, and X. Yi-Nin, "Air combat decision-making method based on dynamic Bayesian network," *Command Control Simulation*, vol. 39, no. 3, pp. 49–54, 2017.
[12] Y. Luo, "Reserch on air-combat decision based on dynamic Bayesian network," M.S. thesis, Shenyang Aerospace Univ., Shenyang, China, 2018.
[13] H. U. A. N. G. Changqiang, D. O. N. G. Kangsheng, H. Hanqiao, T. A. N. G. Shangqin, and Z. H. A. N. G. Zhuoran, "Autonomous air combat maneuver decision using Bayesian inference and moving horizon optimization," *J. Syst. Eng. Electron.*, vol. 29, no. 1, pp. 86–97, Mar. 2018.
[14] A. Sathyan, N. D. Ernest, and K. Cohen, "An efficient genetic fuzzy approach to UAV swarm routing," *Unmanned Syst.*, vol. 4, no. 2, pp. 117–127, Apr. 2016.
[15] N. Ernest and D. Carroll, "Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions," *J. Defense Manage.*, vol. 6, no. 1, p. 2167, 2016.
[16] Z. Yang, D. Zhou, H. Piao, K. Zhang, W. Kong, and Q. Pan, "Evasive maneuver strategy for UCAV in Beyond-Visual-Range air combat based on hierarchical multi-objective evolutionary algorithm," *IEEE Access*, vol. 8, pp. 46605–46623, 2020.
[17] J. Poropudas and K. Virtanen, "Game theoretic approach to air combat simulation analysis," Tech. Rep., 2020.
[18] C. H. E. Jing and Z. Feng-qi, "Simulation of pursuit-evasion resistance based on differential game," *Flight Dyn.*, vol. 32, no. 4, pp. 372–375, 2014.
[19] J. McGrew, J. How, L. Bush, and B. Williams, "Air-combat strategy using approximate dynamic programming," *J. Guid. Control Dyn.*, vol. 33, no. 5, pp. 1641–1654, 2010.
[20] Y. Ma, X. Ma, and X. Song, "A case study on air combat decision using approximated dynamic programming," *Math. Problems Eng.*, vol. 2014, pp. 1–10, Sep. 2014.
[21] H. E. Xu, J. Xiaoning, and F. Chao, "Air combat maneuver decision based on MCTS method," *J. Air Force Eng. Univ.*, vol. 43, no. 3, pp. 34–39, 2018.
[22] C. Heinze, B. Hanlon, M. Turner, K. Bramley, J. Rigopoulos, D. Marlow, and K. Bieri, "The artemis air-to-air combat model," Tech. Rep., 2020.
[23] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and S. Dieleman, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[24] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.
[25] Z. Qiang, R. N. Yang, L. X. Yu, T. Zhang, and J. L. Zuo, "BVR air combat maneuvering decision by using Q-network reinforcement learning," *J. Air Force Eng. Univ.*, vol. 19, no. 6, pp. 8–14, 2018.
[26] C. U. Chithapuram, A. K. Cherukuri, and Y. V. Jeppu, "Aerial vehicle guidance based on passive machine learning technique," *Int. J. Intell. Comput. Cybern.*, vol. 9, no. 3 pp. 255–273, 2016.
[27] Q. Yang, J. Zhang, G. Shi, J. Hu, and Y. Wu, "Maneuver decision of UAV in short-range air combat based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 363–378, 2020.
[28] X. Zhang, G. Liu, C. Yang, and J. Wu, "Research on air combat maneuver decision-making method based on reinforcement learning," *Electronics*, vol. 7, no. 11, p. 279, Oct. 2018.
[29] B. Kurniawan, P. Vamplew, M. Papasimeon, R. Dazeley, and C. Foale, "An empirical study of reward structures for actor-critic reinforcement learning in air combat manoeuvring simulation," presented at the Adv. Artif. Intell., 32nd Australas. Joint Conf., Adelaide, SA, Australia, Dec. 2019.
[30] Q. Yang, Y. Zhu, J. Zhang, S. Qiao, and J. Liu, "UAV air combat autonomous maneuver decision based on DDPG algorithm," in *Proc. IEEE 15th Int. Conf. Control Autom. (ICCA)*, Jul. 2019, pp. 37–42.
[31] K. Jiao, Y. M. Pi, and J. Z. Lu, "A fast calculation method of launch envelope of air-to-air missile," *Fire Control Command Control*, vol. 35, no. 6, pp. 100–102, 2010.
[32] F. Xueyi, L. I. U. Junxian, and Z. Deyun, "Background interpolation for on-line simulation of capture zone of air-to-air missiles," *Syst. Eng. Electron.*, vol. 41, no. 6, pp. 1286–1293, 2019.



DONGYUAN HU was born in Hubei, China, in 1994. He received the M.S. degree from Air Force Engineering University. He is currently pursuing the Ph.D. degree in the research of intelligent decision and artificial intelligence. His research interests include deep reinforcement learning, intelligent air combat, and maneuver decision.

RENNONG YANG was born in 1969. He is currently a Professor and a Ph.D. Tutor with Air Force Engineering University. His research interests include intelligent decision, expert systems, and autonomous planning.

JIALIANG ZUO was born in Nanchang, in 1986. He received the Ph.D. degree from Air Force Engineering University. He is currently an Associate Professor and a Researcher. His research interests include intelligent decision, intelligent air combat, and deep learning.

ZE ZHANG was born in Shaanxi, Xi'an, China. She is currently pursuing the master's degree in management science and engineering with Air Force Engineering University. Her current research interests include maneuver planning and artificial intelligence.

JUN WU was born in Hunan, China. He is currently a Professor and a Master's Tutor with Air Force Engineering University. His current research interests include dynamic planning, expert systems, and autonomous planning.

YING WANG was born in China, in 1984. She is currently an Associate Professor and a Master's Tutor with Air Force Engineering University. Her current research interests include autonomous planning, optimization theory, and artificial intelligence.