Article

Mastering air combat game with deep reinforcement learning

Jingyu Zhu [1], Minchi Kuang [2,*], Wenqing Zhou [3], Heng Shi [2,*], Jihong Zhu [2], Hongli Zhang [1], Xinkai Li [1] and Xu Han[4]

[1] School of Electrical Engineering, Xinjiang University, Urumqi 830000,China; zhu-jingyu@stu.xju.edu.cn(J.Z.);zhl@xju.edu.cn(H.Z.);lxk@xju.edu.cn.
[2] Department of Precision Instruments, Tsinghua University, Beijing 100190, China; jhzhu@tsing-hua.edu.cn(J.Z.).
[3] Department of Computer Science and Technology, Tsinghua University, Beijing 100190, China; zhouwq14@gmail.com(W.Z.).
[4] Chengdu Aircraft Design & Research Institute, Aviation Industry Corporation of China, Chengdu 610000, China.
* Correspondence: kuangmc@mail.tsinghua.edu.cn; shiheng@tsinghua.edu.cn

Abstract: Reinforcement learning is used for air combat problems in recent years, and the idea of curriculum learning is often used for reinforcement learning, but traditional curriculum learning suffers from the problem of plasticity loss in neural networks. Plasticity loss is the difficulty of learning new knowledge after the network has converged. To this end, we propose a motivational curriculum learning distributed proximal policy optimization (MCLDPPO) algorithm, through which agents trained can significantly outperform the predictive game tree and mainstream reinforcement learning methods. The motivational curriculum learning is to help the agent gradually improve their combat ability by observing the agent's unsatisfactory performance and providing corresponding rewards as a guide. Additionally, a complete tactical maneuver is encapsulated based on existing air combat knowledge, and through the flexible use of these maneuvers, some tactics beyond human knowledge can be realized. In addition, we designed an interruption mechanism for the agent to increase the frequency of decision-making when the agent faces an emergency. When the number of threats the agent receives changes, the current action will be interrupted to reacquire observations and make decisions again. Using the interruption mechanism can significantly improve the ability of the agent. To simulate actual air combat better, we built an air combat environment based on digital twin technology to create an air combat engine core such as missiles and radars. Based on this environment, we develop a distributed intelligent training system that employs Redis as the cache and obtains the data generated in the distributed environment through a multi-threaded TCP/IP protocol, which significantly increases the training data throughput and helps the agent accelerates convergence. The experimental results demonstrate that the agent can fully exploit the situational information to make reasonable decisions and affords air combat tactical adaption, verifying the effectiveness of the algorithmic framework proposed in this paper.

Keywords: Air combat, MCLDPPO, Interruption mechanism, Digital twin, Distributed system

1. Introduction

Air combat is one of the military pilots' most challenging and dangerous missions. As a result, various systems have historically been developed to reduce the pilot's workload in air combat or even replace pilots in autonomous decision-making. With the rapid improvement of sensor technology and computing power, the autonomous decision-making capability of air combat systems has gradually become possible1. Although a fully autonomous air combat system has not been realized, several projects and studies are already underway. However, to face the challenges of future air warfare integrating artificial intelligence and air combat countermeasures technology, and to improve the core

combat capability of the future air and space battlefield2, developing a new intelligent air combat system generation is crucial.

With the rapid development of artificial intelligence technology, traditional methods are struggling to solve some complex and challenging tasks. Nevertheless, although artificial intelligence can solve complex problems, it comprises many black boxes. The National Aeronautics and Space Administration (NASA) built a human empirical knowledge base between the 1960s and 1990s and experimented with artificial intelligence methods to build air combat intelligence systems, aiming to replace humans in air combat decision-making[3-5].

Many scholars have proposed various algorithms to investigate autonomous air combat maneuvers, divided into three categories: game-theoretic, optimization theory-based, and artificial intelligence. The basic assumptions underlying the theory are that decision-makers pursue well-defined exogenous objectives (rational) and consider their knowledge or expectation of other decision-makers (they reason strategically)6. Game theory has been widely applied in military operations, with air combat problems based on game theory mainly divided into differential games and influence diagram games. Differential game theory is primarily used in multivariate controlled continuous dynamic system game confrontation problems and is commonly used in air warfare to design parts of the fugitive pursuit game7. The influence diagram game creates a probabilistic topology based on expert knowledge construction and a series of parameter learning methods to replace the pilot's air combat decisions8. For instance, Perk et al.9 applied the differential game to autonomous decision-making in one-to-one line-of-sight air combat. Alkaher et al.10 improved the differential game and used it in decision-making for over-the-horizon air combat. Additionally, this work introduced the concept of dynamic escape zones. The work of Alkaher et al. was extended by Schachter11 to multiple decision-makers to create a dynamic escape zone. Koller and Milch12 developed multi-agent influence diagrams and introduced the strategy correlation concept to explore the Nash equilibrium in multi-decision-making in noncooperative games. Virtanen et al.13 extended the influence diagram game to dynamic multi-stage decision problems, and Pan14 proposed a state-prediction influence diagram model to solve the close-range air combat problem. Zheng et al.15 suggested an improved fuzzy influence diagram that combines qualitative and quantitative analysis of graphical models for decision problems.

Decision-making techniques for air warfare based on optimization theory belong to the second group. For example, Jonathan et al.16 developed a nonlinear model predictive controller that encodes 3D UAV maneuvers for solving the UAV pursuit problem. Kaneshige and Krishnakumar17 employed an artificial immune mechanism to solve the air combat maneuver decision problem, and Wang18 et al. proposed a robust maneuvering decision method with adaptive target intent prediction. Moreover, JI et al.19 suggested an improved TIMS model that combines sequential correlated data with the TIMS model to generate air combat countermeasures faster. Huang et al.20 considered the air combat game as a Markov process and computed the air combat form utilizing Bayesian inference. Additionally, McGrew21 formulated and solved a functionally approximated dynamic program for seeking air combat strategy.

Artificial intelligence-based decision-making techniques for air combat belong in the third category. Specifically, rule-based expert systems and self-evolving machine learning class techniques are the two primary artificial intelligence types for air warfare decision-making. In 1969, Burgin and Owens created a maneuver decision software that used adaptive maneuver logic, essentially an expert system using IF-ELSE-THEN logic as its primary logic. Wen et al.22 suggested a hybrid tactical decision-making system based on rule sets and a Fuzzy Bayesian Network (FBZ) to deal with uncertainty and afford real-time decisions during an air conflict. Zuo et al.23 coupled heuristic algorithms with reinforcement learning by utilizing expert knowledge as heuristic signals. An intelligent tactical decision-making system based on DQN was proposed by Liu et al.24. Moreover, Yang et al.25 developed an autonomous air combat maneuver decision-learning model for UAVs relying on a deterministic policy gradient (DDPG). Xie et al.26 proposed a decision-

 doi:10.20944/preprints202305.0862.v1

making method combining a dynamic relational weight algorithm and a moving time strategy, incorporating trajectory prediction in maneuver decision-making. Lockheed Martin27 combined a layered architecture with maximum entropy reinforcement learning to form expert knowledge through the strategy's reward and support modularity. This method achieved second place in DARPA's Alpha Dogfight test and defeated graduates of the U.S. Air Force's F-16 Weapons Instructor Course during a competition. However, the competition did not involve an actual missile but used a strike zone setting to simplify the problem. In recent years, research has involved complex environments[28-31]for air combat, with Piao et al.30 suggesting PPO32 algorithms for air combat decisions in complex environments. However, the authors did not use expert systems as a benchmark to judge their capabilities. Most of the UAV air combat algorithms mentioned above are studied in scenarios involving simplified environments that differ significantly from the actual application scenarios.

Therefore, we build a high-precision air combat simulation environment and propose a novel motivational curriculum learning distributed proximal policy optimization algorithm. To be able to fully reflect the effectiveness of the method, we use the predictive game tree of Zhou et al.33 and the mainstream reinforcement learning methods as a comparison. Our method can significantly outperform the predictive game tree and existing mainstream reinforcement learning methods. The main contributions of this paper are as follows.

(1) We propose a motivational curriculum learning distributed proximal policy optimization algorithm, abbreviated as MCLDPPO. This algorithm is different from the idea of traditional curriculum learning, which trains agents by designing courses corresponding to skills. The motivational curriculum learning we propose is to use the agent to actually confront the shortcomings of the performance, and use the corresponding rewards as a guide. By continuously adding guidance rewards, the agent's combat capability is continuously improved.

(2) An interrupt mechanism that changes according to the threat is designed. Hierarchizing the action space into a complete set of tactical maneuvers encapsulated by existing knowledge of air combat enables the algorithm to have human-like hierarchical decision-making capabilities. By using discretized corresponding actions, each action has rich parameters. By combining these actions reasonably, a large number of classic air combat actions can be realized, and some tactics beyond human knowledge can be realized.

(3) Use Unity3D to build a set of simulation scenes close to the real air combat environment, which includes real air combat elements such as aerodynamics, infrared, radar, and missiles. The effectiveness of the proposed algorithm is verified by using this highly digital twin simulation environment and comparing its performance with rule-based finite state machines, predictive game trees, and classical reinforcement learning algorithms.

The rest of this paper is organized as follows. Section 2 introduces the motivational curriculum learning and relevant components of simulation environment architecture, network architecture, system architecture, state space, action space, and reward function. The experimental part is mainly in Section 3. We compare the performance between each algorithm and illustrate the shortcomings of traditional course learning. We use detailed experiments to describe the design process of motivational curriculum learning and effective reward ablation experiments, and describe the evolution of networks and the influence of hyperparameters. Finally, we conclude the paper and outlook for future work in Section 4.

2. Framework and Components

With the rapid development of deep learning technology, combining neural networks and reinforcement learning has become a research hotspot. Deep reinforcement learning is one of the most successful artificial intelligence methods in recent years. Indeed, in 2015 Google's DeepMind team combined deep learning and reinforcement learning in the Atari game and proposed the Deep Q Network34 (DQN), which reached the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

level of human experts. Recently, OpenAI Five35 and AlphaStar36 have reached the top human level in competitive games, demonstrating the full potential of deep reinforcement learning.

This section mainly focuses on the overall design process of a single UAV against air combat agents, including the design of an air combat environment, detailed coding of observation space, interrupted action space design, motivational course learning reward design, and the framework of a training system. The air combat agent obtains observations represented by feature extraction and high-dimensional vectors, inputs them into the policy network and value network, and outputs discretized actions that can have an interruption mechanism. Guided by rewards that motivational curriculum learning, DPPO is used to optimize the strategy network and value network to obtain the optimal strategy. Figure 1 illustrates the air combat agent of reinforcement learning schematic.

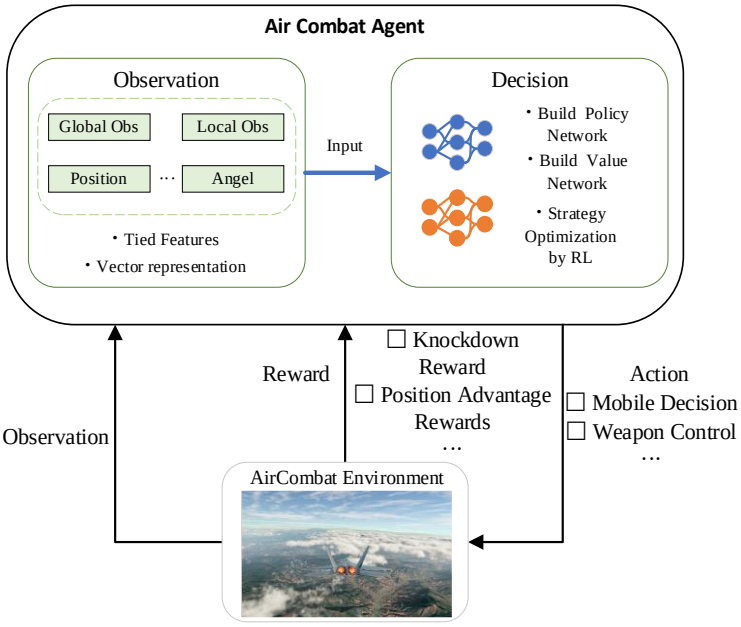


Figure 1 Air combat agent architecture.

2.1. Overview: Motivational Curriculum Learning with DPPO

We aim to solve the problem of delayed rewards by motivational curriculum learning distributed proximal policy optimization (MCLDPPO). Motivational curriculum learning differs from traditional course learning, which helps agents learn by specifying tasks ranging from simple to complex, but neural networks have the problem of plasticity loss. During the training process, as the neural network starts to converge, having a large number of neurons die leads to the loss of diversity in the neural network. If the course changes too much, it is difficult for the agent to cross over from this course to another, losing plasticity. To be able to avoid the problem of plasticity loss, we no longer design individual courses, but by add rewards that contain courses. By continually adjusting the curriculum rewards, we are able to guide agents to achieve the effects of traditional course learning. At the same time, by adding multiple curriculum rewards, we can avoid the problem of agents not being able to continue learning other curriculums after learning one curriculum. We empower agents by adding reasonable curriculum reward step by step to enable them to learn multiple courses at the same time.

Moreover, we propose a novel interruption mechanism, which interrupts the current action to obtain the current observation to re-make a maneuver decision when the agent has a significant indicator change in the observation space. With the addition of the interrupt mechanism, the agent's decision frequency is substantially improved. **Figure 2** illustrates the architecture of MCLDPPO.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

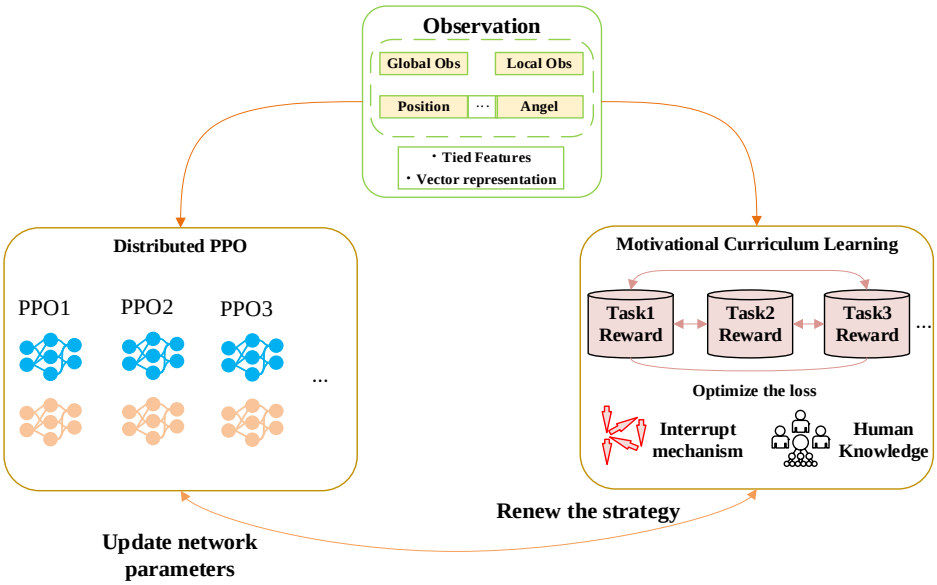


Figure 2.The architecture of MCLDPPO.

For a single UAV combat system, the UAV combat process is modeled as a POMDP problem, which mainly consists of a five-tuple $< S, A, R, P, \gamma >$. In air combat, the state s_t is expressed as the current time t of being able to represent the current state of the UAV. For example, the current position of the UAV, information about fired missiles and enemy alerts. The drone uses the current observation s_t to select the action a_t. By executing the action a_t combined with the set reward function, the reward r_t corresponding to the current action is obtained. After executing action a_t, the drone reaches a new state s_{t+1}, makes a new decision a_{t+1} according to the new state, and gets the reward r_{t+1} for the next state s_t. where S consists of a series of discrete states $s_0, s_1, \dots, s_t, s_{t+1}, \dots$; A is a series of discrete actions $a_0, a_1, \dots, a_t, a_{t+1}, \dots$; R is a series of discrete rewards $r_0, r_1, \dots, r_t, r_{t+1}, \dots$; P is the state transfer function, usually $P(s_{t+1}|s_t, a_t)$ is expressed as the probability distribution of the current state action pair (s_t, a_t) mapping to the successor states s_{t+1} that can be reached; γ is the discount factor, which is used to define the importance of future rewards.

The goal of each agent is to maximize total return, defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k} \tag{1}$$

To maximize the total reward obtained, we use a policy learning approach to learn an optimal policy π^* to make action decisions on agents. For adequately represent the value of the current state s_t and action a_t, the state value function $V^{\pi}(s_t)$ and the action value function $Q_{\pi}(s_t, a_t)$ are used to represent them. The policy-based π state value function $V^{\pi}(s_t)$ is denoted as $V^{\pi}(s_t) = E_{\pi}[G_t|S = s_t]$. The action-value function is defined as $Q_{\pi}(s_t, a_t) = E_{\pi}[G_t|S = s_t, A = a_t]$. The Advantage function is defined as $A_{\pi}(s, a) = Q_{\pi}(s, a) - V_{\pi}(s)$. By parameterizing the agent's policy π_{θ} , an objective function can be designed to measure the superiority of the policy. θ is the parameter of the policy network.

$$J(\theta) = E_{s_0}\left[V_{\pi_{\theta}}(s_0)\right] \tag{2}$$

The objective function is derived concerning the strategy's parameter θ, and after obtaining the derivative, a gradient ascent is used to maximize the objective function, optimizing the strategy. The strategy gradient aims to find an optimal strategy π^* and maximize its expected payoff in the environment.

$$\pi^* = \arg\max_{\pi} E_{\pi}\left\{\sum_{t}^{H} \gamma^t r_{t+1} \mid S = s_0\right\} \tag{3}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

The strategy gradient method mainly updates the parameters iteratively along the gradient direction. However, this algorithm cannot guarantee the size of each update step, potentially deteriorating the strategy due to being too significant, ultimately affecting the training. Therefore Schulman et al. 37developed a Trust region policy optimization (TRPO) algorithm that solved the random policy optimization. This algorithm finds a trust region during the update, and updating this region guarantees policy performance stability. This process relies on the Kullback-Leibler (KL) divergence between the old and new policies to measure the distance between them and overcomes the learning rate limitation, theoretically guaranteeing the monotonicity of the strategy learning performance. However, this is challenging during the TRPO computations, especially the computation of the second-order Hessian matrix, and thus Schulman et al.38 invented the (Proximal policy optimization, PPO) algorithm using first-order derivatives.

The PPO algorithm is implemented as the PPO-Penalty and PPO-Clip, with the latter used more often. The framework used in the PPO algorithm is the Actor-Critic (AC), which ensures that the gap between the new and old parameters is not too large by directly restricting the objective function. Similarly to TRPO, the Actor-network update part of PPO maximizes the "surrogate" objective. Where Equations 4 denotes the probability ratio $r_t(\theta)$ of the current strategy π_θ to the old strategy π_{θ_k}.

$$\begin{aligned} max\ L^{CPI}(\theta) &= \hat{E}_t\left[r_t(\theta)\hat{A}_t\right] \\ r_t(\theta) &= \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_k}(a_t \mid s_t)} \end{aligned} \tag{4}$$

The generalized advantage estimation (GAE) is used to estimate the action's advantage value $\hat{A}_t$ in the state based on the temporal-Difference (TD(λ)) concept, which replaces the GAE's value function with the advantage function to obtain different approximate estimates by adjusting the size of λ in the advantage function.

$$L^{clip}(\theta) = \hat{E}_t\left[min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right] \tag{5}$$

where Equation 6 presents the clip implementation, aiming to limit the policy variation to $[x_{min}, x_{max}]$ that it is not too large, and to significantly simplify the algorithm using KL loss.

$$\text{clip}(x, x_{min}, x_{max}) = max(min(x, x_{max}), x_{min}) \tag{6}$$

Entropy is a measure that represents a random variable's uncertainty. In reinforcement learning, the agent improves its exploration ability utilizing a strategy entropy added to the Actor's loss and multiplied by an Entropy coefficient of 0.01. A strategy's entropy can be expressed as:

$$\begin{aligned} \mathcal{H}\left(\pi(\cdot \mid s_t)\right) &= -\sum_{a_t} \pi(a_t \mid s_t) \log\left(\pi(a_t \mid s_t)\right) \\ &= E_{a_t \sim \pi}\left[-\log\left(\pi(a_t \mid s_t)\right)\right] \end{aligned} \tag{7}$$

The critic network uses the $\text{TD} - \text{Error}$ to update the network parameters, where the state's value function $V_\phi(s_t)$ is estimated. ϕ is the parameter of the value network.

$$TD - Error = \gamma V_\phi(s_{t+1}) + r_{t+1} - V_\phi(s_t) \tag{8}$$

To speed up the agent's convergence, we use a distributed proximal policy optimization technique to train the agent, as presented in the algorithmic pseudo-code.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

Algorithm 1: Distributed proximal policy optimization (chief)

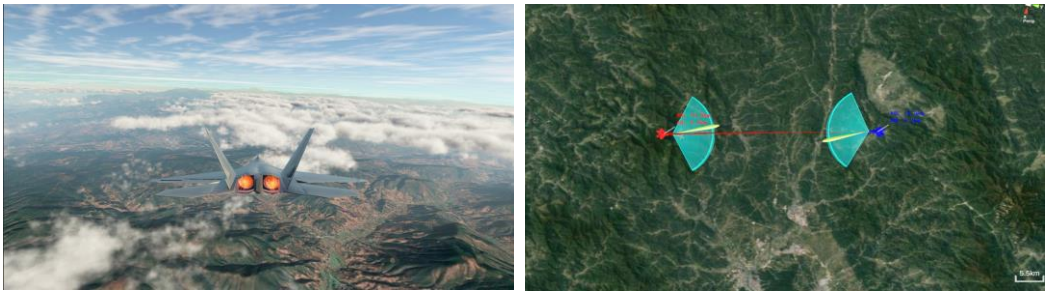
N GPUS, M Workers in One GPU, B iterations.
for $i \in \{1, \cdots, N\}$ **do**
  for $j \in \{1, \cdots, M\}$ **do**
    Wait until setting gradients θ, ϕ are available
    Average gradients and update global θ
    Average gradients and update global ϕ
  end
end

Algorithm 2: Distributed proximal policy optimization (worker)

for $i \in \{i, \cdots, N\}$ **do**
  for $j \in \{1, \cdots, M\}$ **do**
    Run policy π_θ to collect data $\{s_t, a_t, r_t, s_{t+1}, \cdots\}$
    Store partial trajectory information
  end
  $\pi_{old} \leftarrow \pi_\theta$
  for $b \in \{1, \cdots, B\}$ **do**
    get data $\{s_t, a_t, r_t, s_{t+1}, \cdots\}$ from buffers
    Using GAE to calculate Advantage A^{π_θ}
    $J_{ppo}(\theta) = \sum_{t=1}^{T} \left[min\left(r(\theta), clip\left(r(\theta), 1-\epsilon, 1+\epsilon\right)\right) A^{\pi_{\theta_k}}(s,a) \right]$
    Policy update rate $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$
    $V_{loss} = \sum_{t=1}^{T} (-V(s_t) + r_t + \gamma V(s_{t+1}))^2$
    Compute $\nabla J_{ppo}, \nabla V_{loss}$ and send gradient θ, ϕ to chief
  end
end

2.2. Simulation Environment Architecture

In order to simulate real air combat confrontation better, an air combat simulation system was built using the game development engine Unity3D software40. **Figure 3** depicts the digital twin experiment environment.



(a) (b)

Figure 3. Digital twin-air combat environment. (a) F22 in the air combat simulation environment. (b) Two aircraft confront each other in a simulated environment.

The experimental environment of the digital twin also includes missiles, infrared, and radar systems. Among them, a radar system simulates the active phased-array radar used on the most advanced fighter aircraft today. When the radar finds a target, one of the target detection sub-modules provides the AI module with all the relevant information, i.e., target position, velocity, and missile. This radar cannot gather hostile data if it detects electromagnetic interference or does not detect the enemy target. The experimental environment utilizes a human-machine interface that allows human pilots and air combat agents to compete in an entirely recreated real-world scenario. **Figure 4** illustrates the architecture of the air combat simulation environment.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

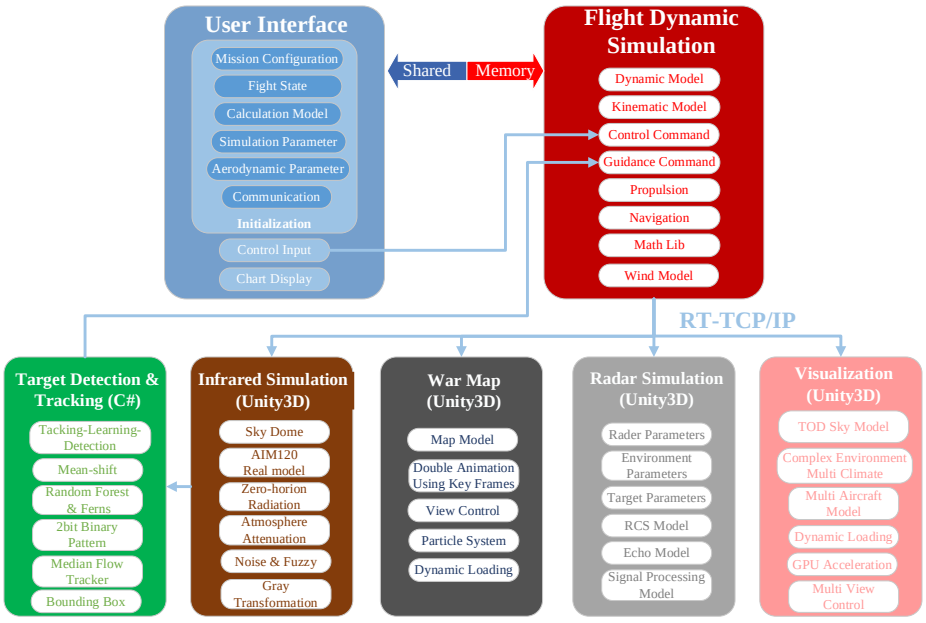


Figure 4. Simulation Environment Architecture.

2.3. *Design of observation space*

In the actual air warfare environment with a vast observation space, encoding the raw observation data and processing the complex observation space into individual vectors is necessary. The observation vector is divided into global and local observation vectors involving each observation embedding. The agent can only obtain the local observation information, while most of the enemy aircraft's information is unavailable. Furthermore, without the enemy's view, the agent has only information about the local machine, and the actor makes decisions through local observations. Simultaneously, the Critic uses global observations to judge the overall current situation, thus providing guidance capabilities to the Actor. **Figure 5** illustrates the specific observation space design.

Local Observation	Global（4）	Aircraft properties						Enemy aircraft properties					
		common(20)	state(18)	rwr(10)	weapons(6*6)	detection(54)	missile(29*6)	common(20)	Target-related(23)				
Global Observation	Global（4）	Aircraft properties						Enemy aircraft properties					
		common(20)	state(18)	rwr(10)	weapons(6*6)	detection(54)	missile(29*6)	common(20)	Target-related(23)	detection(54)	state(18)	weapons(6*6)	missile(29*6)

Global	**4**	**Rader Warning Receiver, RWR**	**10**	elevation angle (EL)	2		
time since air combat started	1	rwr type	4	relative height	1		
airspace anchor points	3	azimuth angel	2	distance between two aircraft	1		
Common	**20**	pitch angle	2	**Missile Information**	**29**		
aircraft ID	1	warning distance	1	Is the missile on target?	1		
aircraft Type	5	threat level	1	Is the missile guided?	1		
current Aircraft Camp	3	**Weapons**	**6**	camp	3		
current Aircraft Position(x,y,z)	3	weapon type	3	azimuth angle of the missile	2		
offset	3	minimum firing distance	1	pitch angle of the missile	2		
movement speed	3	maximum firing distance	1	distance of missiles from aircraft	1		
height above ground	1	is castable?	1	speed of the missile	3		
Elevation	1	**Detection Information**	**54**	current missile guidance phase	3		
The aircraft's own state	**18**	Whether to give missile guidance	1	position of enemy aircraft	3		
current overload	1	last 16 timesteps' locations	48	azimuth angle of enemy aircraft	2		
is it locked by the enemy?	1	current radar level	4	pitch angle of enemy aircraft	2		
current acceleration	3	number of missiles remaining	1	speed of enemy aircraft	3		
time since last fire	1	**Target-related information**	**23**	missile launching time	1		
time since last search for enemy aircraft	1	is the enemy detected?	1	estimated time of missile hit	1		
pitch angle	1	azimuth of the enemy	2	distance of the current missile from enemy aircraft	1		
rolling Angle	1	pitch of the enmey	2				
yaw angle	1	closure rate	1				
current throttle volume	1	maneuver overload ratio	1				
facing angle	3	antenna train angle (ATA)	4				
angle of attack	2	aspect angle (AA)	4				
angle of sideslip	2	horizontal crossover angle(HCA)	4				

Figure 5. Observation space design.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

A significant observation space has different data formats, and it is A significant observation space has different data formats, and it is necessary to sample the observations for continuous data to obtain the discrete values. For some spatially discrete data, the data are merged by concatenating them within each cell. Our method utilizes a fixed encoding format for some disordered sets and combines all the encoded data to create the agent's observation space, as shown in Figure 6.

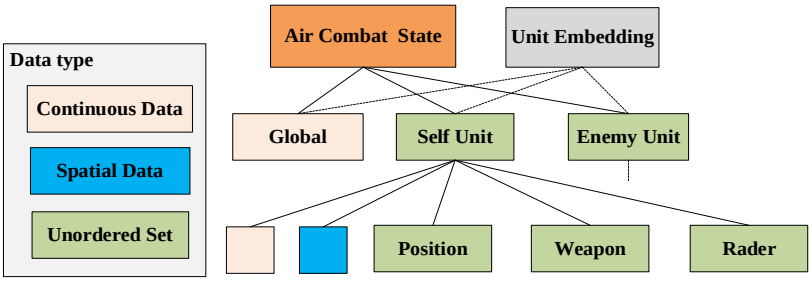


Figure 6. Observation space encode.

The observation space comprises the agent's state, the enemy's state, and their relative postures. To facilitate the network understanding of the posture state, we added some processed posture information that maps the raw data and the enemy's posture based on McGrew's geometric representation of the air combat posture as a reference41. Specifically, we consider antenna column angle (ATA), aspect angle (AA), horizontal crossing angle (HCA), and distance (R) to characterize the enemy-agent advantage (see **Figure 7**). Note that HCA is the angle between the red and blue aircraft heading, and ATA indicates the angle from the blue aircraft heading to the red aircraft. When the AA angle is 0, the blue aircraft trails the red aircraft. In actual air combat, when the AA and ATA are zero, the blue aircraft has a significant advantage over the red aircraft at this time.

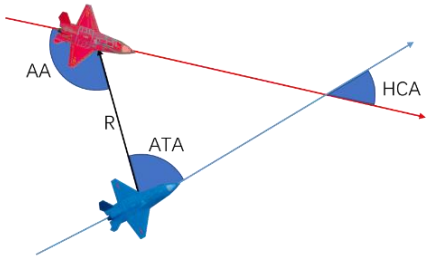


Figure 7. Diagram of air combat angles.

During actual air combat, the information available to current fighters is imperfect because of the fighter fog limitations and the radar detection capability, highlighting the importance of the state space. The system state space comprises the position and speed of the aircraft and the enemy aircraft X_{self}、X_{enemy}、V_{self}、V_{enemy}, attitude aircraft's angle (φ, θ, ϕ), the aircraft-enemy aircraft range R, the antenna column angle ATA, the vertical and horizontal angle AA, the horizontal cross angle HCA, the elevation angle EL, the radar lock signal Lock, Radar Level, radar warning information of missile arrival (RWR), number of remaining missiles M_{left}, and the missile's availability for launch $M_{Castable}$. The total air combat state space is defined as follows.

$$S = [X_{\text{Self}}, X_{\text{Enemy}}, V_{\text{Self}}, V_{\text{Eneny}}, \varphi, \theta, \phi, R, ATA, AA, HCA, EL, \text{Lock}, \text{RaderLevel}, RWR, M_{\text{Left}}, M_{\text{Castable}}] \quad (9)$$

2.4. Interrupted action space design

To fully self-explore the agent, the tactics utilized in air combat and set to the agent are primarily the fundamental maneuvers. Using these basic maneuvers flexibly, the agent can make joint maneuvers during air combat, such as the Immelmann and Cobra

 doi:10.20944/preprints202305.0862.v1

maneuvers. Simultaneously, by limiting the maneuver types, the maneuvers search space is reduced**33**. The design of the action space is shown in Table 1.

Table 1. Action Design

Type	Parameters	Description
Straight	Target pitch angle	There are three types: straight flight, pull-up, and swoop
Track	Target location and orientation	It is usually used to track the enemy or the predicted point
Circle	Joystick pitch and target roll angle	Six sets of fixed parameters were developed for hovering
Loop	Pitch angle	Causes dramatic altitude and speed changes
Attack	Target predicted location	Predict target arrival position from target current position speed
Escape	Alarm information	Adjust the body perpendicular to the alarm direction and down the high rotation

The Straight maneuver considers the aircraft flying straight ahead and is influenced by the aircraft's pitch angle, which is divided into three maneuvers: straight, pull up, and dive. The Circle maneuver involves pulling up the aircraft's nose to make a turn after a roll, where the control quantities are the target roll angle and the joystick parameters. We develop six hovering parameter sets to reduce the maneuver's search space to afford to hover with different radius sizes. Moreover, the loop somersault dramatically changes the aircraft's pitch angle by fixing the joystick's pitch input. An agent combining a somersault and a straight flight can achieve an Immelmann turn, which rolls the aircraft upward 180 degrees before leveling the pitch and turning it horizontally. The Attack maneuver involves firing a missile in the current direction, and the Escape maneuver considers two maneuver types: warning turn and Diving. The former turns the fuselage to fly 90° perpendicular to the warning direction of the incoming missile. The latter rolls down 180 degrees, adjust the fuselage attitude and is typically used to avoid the enemy's incoming missiles.

The action space is defined as follows:

$$A = [\text{ Straight, Track, Circle, Loop, Attack, Escape }] \tag{10}$$

In a natural air combat environment, if the maneuvering frequency changes too fast, it will constrain the airframe's inability to make a good attitude. Considering that an aircraft requires some time to execute and complete the maneuvers, we set a fixed time for each maneuver. Different maneuver types are set between them by artificially setting reasonable values to help the agent reduce the maneuver search space.

During the actual proximity confrontation, because of the large observation space and the action needing to last for some time, the agent has the characteristic of not being able to perceive the change in the main observation quantity. To be able to solve this problem, we propose an interruption mechanism for the change of the main observation quantity. The primary observation consists of the enemy missile warning threat, whether or not the enemy field of view is acquired by the own side and the missile observation of the own side. When each category of observations increases or decreases, it triggers an interruption in the system, allowing the agent to reuse the current observations to make a new maneuver decision. This gives the agent the ability to cope with complex and changing situations at close range. The introduction of the interruption mechanism accelerates the agent's decision frequency and significantly improves the agent's combat performance.

2.5. Motivational curriculum learning reward design

An agent's ultimate goal is to win the game. Therefore, designing the reward function is vital during the agent's training. Traditional reward functions usually reward the winner of the final game. However, such a strategy has very sparse rewards, making it

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

difficult for the algorithm to converge. To overcome this problem, we define the agent's actions during an air battle that lead to the final positive or negative rewards, such as gaining the enemy's field of view and the dominance value between the two aircraft postures. The rewards of the corresponding posture are generated by evaluating the posture in the air battle.

When adding nodal event rewards, a scene has specific rewards that sufficiently encourage the agent. The rewards are divided into nodal event rewards and continuous change rewards.

Nodal event bonus: Close-range dodging enemy missiles gives a bonus. At the same time, the missiles skimming an enemy aircraft at close range force them to maneuver massively to pave the way for subsequent attacks, which are strategically significant and provide a particular reward. Adding nodal event rewards encourages the agent to explore.

Continuous change bonus: When the aircraft maneuvers, many continuous quantities change, such as speed, attitude, and velocity affecting the Speed and sideslip angle limit penalties, relative attitude advantage between aircraft, and missile threat rewards. The continuous bonus is calculated as follows:

$$R_{velocity} = \begin{cases} -80, & V < 80 \\ V - 160, & 80 \le V \le 160 \\ 0, & V > 160 \end{cases} \tag{11}$$

$$R_{\beta} = \begin{cases} -40, & \beta > 20^{\circ} \\ -0.1\beta^{2}, & \beta \le 20^{\circ} \end{cases} \tag{12}$$

In air combat, the speed of the UAV is particularly crucial, and $R_{velocity}$ is used to encourage the agent to maintain a high speed. To keep the UAV in a relatively well attitude, a certain penalty is given to the aircraft's sideslip angle, and R_{β} is employed to limit the aircraft's sideslip angle.

$$R_{Advantage} = \begin{cases} 80e^{-\frac{Angle^{2}}{1300}}\left(\frac{Distance}{1000}\right), & Distance \le 1000 \\ 80e^{-\frac{Angle^{2}}{1300}}\left(\frac{39000 - Distance}{38000}\right), & 1000 < Distance \le 20000 \\ 80e^{-\frac{Angle^{2}}{1300}}\left(\frac{10000}{Distance}\right), & 20000 < Distance \end{cases} \tag{13}$$

$$R_{\text{Threat}} = \begin{cases} -160e^{-\frac{ang^{2}}{144}}\left(1 - \frac{t}{20}\right), & 0 \le t \le 20 \\ 0, & t > 20 \end{cases} \tag{14}$$

To evaluate the attitude advantage $R_{Advantage}$ between UAVs reasonably, altogether the attitude advantage bonus is evaluated from the combination of angle and distance between UAVs. Angle is the angle between two UAVs, Distance is the distance between two UAVs. R_{Threat} is used to evaluate the threat of enemy missiles to UAVs, ang is the angle between missiles and UAVs, t is the missile approach time of the missile approaching the UAV. For a more reasonable evaluation of R_{Threat}, both ang and t take into account a certain amount of future prediction.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

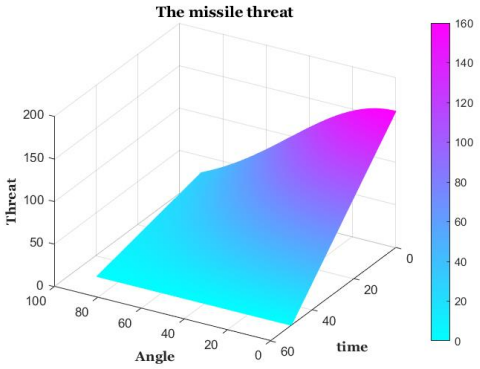


Figure 8. The missile threat.

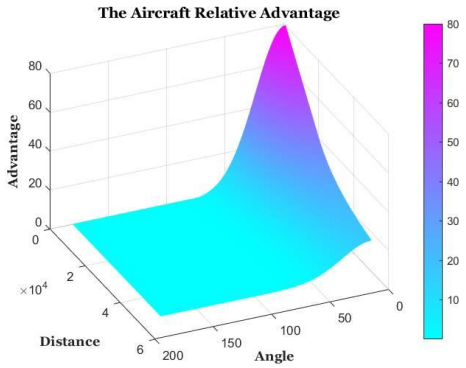


Figure 9. The Aircraft Position Advantage.

Table 2 Reward Design

Type	Reward	Description
Hit	640	The missile hits opposing enemy aircraft
Be hit	-640	Hit by enemy missile
Launch missile	-30 ~ -10	Change according to remaining missiles
Missile Threat	0～160	Calculated from missile field of view and approach time
Position Advantage	0～200	Calculated from relative distances and angles
View	100	Radar acquisition of enemy position through movement
Lost view	-100	Loss of radar to enemy position through movement
Stall	-80~0	Aircraft airframe stall caused by maneuvers
Side slip angle limit	-40~0	Maneuvering leads to excessive side slip angle
Missiles skim close range	50	Fired missiles approach enemy aircraft at close range
Dodge missiles at close range	50	Close-range avoidance of incoming enemy missiles

Table 2 shows the reward function designed to fully use the knowledge of air combat, but the actual performance of the agent was not satisfactory. The initial training was done in a traditional course learning manner, first by using a targeting machine to train attack skills and subsequently by using a state machine to train evasion skills. Although this training approach is effective for deep learning, it is less applicable for reinforcement learning. Because of the problem of plasticity loss of neural networks, we propose motivated course learning based on the idea of course learning. By observing the actual confrontation, we analyze its problems and guide it to do better by designing motivational rewards. And the calculation of missile threat and the size of each reward were redesigned. The specific experimental implementation process of motivational curriculum learning is shown in Section III.C. The curriculum reward function is shown in Table III.

Table 3 Curriculum Reward Redesign

Type	Reward	Description
Hit	100	The missile hits opposing enemy aircraft
Be hit	-100	Hit by enemy missile
Missile Threat	0～20	Calculated from missile field of view and approach time
Position Advantage	0～30	Calculated from relative distances and angles
View	20	Radar acquisition of enemy position through movement

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

Lost view	-20	Loss of radar to enemy position through movement
Stall	-20~0	Aircraft airframe stall caused by maneuvers
Side slip angle limit	-10~0	Maneuvering leads to excessive side slip angle
Missiles skim close range	10	Fired missiles approach enemy aircraft at close range
Dodge missiles at close range	10	Close-range avoidance of incoming enemy missiles

2.6. Design of training system

We constructed a framework for a distributed algorithm, fully exploited its distributed capabilities, and conducted a large number of air combat simulation scenarios in parallel to generate sufficient training data.

The Unity-based simulated air combat environment runs at a default rate of 50 frames per second because it is a highly realistic real-time scenario. The air combat agent runs every ten frames, called a time step, during which the agent acquires a series of observations from the environment. The observations involve all the information available under the current agent's field of view, which is non-perfect information. Then, the agent exploits these observations to make decisions and outputs the actions the current state takes, such as attacking, pursuing, and avoiding a target.

During training, the agent is randomly initialized to any location within the synthetic environment. It should be noted that it is crucial to have a sufficiently diverse training game to ensure that the agent can be fully explored and thus ensure the robustness of the agent's strategy. Additionally, we employ the Deep LSTM core network to handle the temporal data and solve the partially observable problem in the air warfare problem.

This work aims to develop an optimal strategy for air combat, defined as a probability distribution function from the observation to the action space. Thus, we designed a complex network structure to accomplish this goal, with the neural network comprising a Deep 2048-cell LSTM network, we propose an Actor-Critic network architecture affording global and partial observation separability.

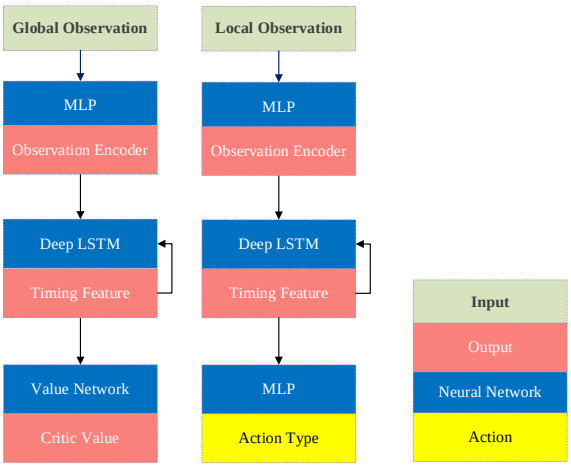


Figure 10. Architecture of the network model.

Given a policy, the agent interacts with the environment game by continuously using the current observation as input and sampling the actions from the output distribution at each time step. The complex multidimensional observations are input to the LSTM network by encoding. The temporal features are extracted using the LSTM network state, followed by predicting the currently executed policy (action and value functions) using a fully connected layer.

To guide the agent to find the optimal strategy quickly, we decompose the core features of air combat concerning the pilot's actual combat experience and design reward functions that contain practical meanings. The latter are used as reward functions for the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

agent to fire missiles against enemy agents and to achieve posture dominance. During the reward function design process, based on the idea of motivational curriculum learning, several effective and practical reward functions are designed to fully alleviate the problem of sparse rewards. Designing such additional reward functions is vital for the agent's successful training.

Specifically, we employ a policy learning approach in reinforcement learning to train the air combat agent and use a distributed proximal policy optimization algorithm to update the network. Due to the delay between launching an air missile and hitting the target, this class of problems is called the credit allocation problem. To solve the latter problem, we use GAE to estimate the dominance of these actions between subsequent actions. Hence, training is accelerated using a standard variance stabilization technique based on the estimated dominance.

The training system comprises four main parts, as shown in **Figure 11**. The adversarial training environment simulates the actual air combat environment through Unity. By conducting multiple parallel battlefields in the air combat environment, scheduling each air combat environment through multi-process and multi-thread management can boost the data generation process of the air combat environment by interacting with the agent hundreds or even thousands of times to accelerate the network's convergence.

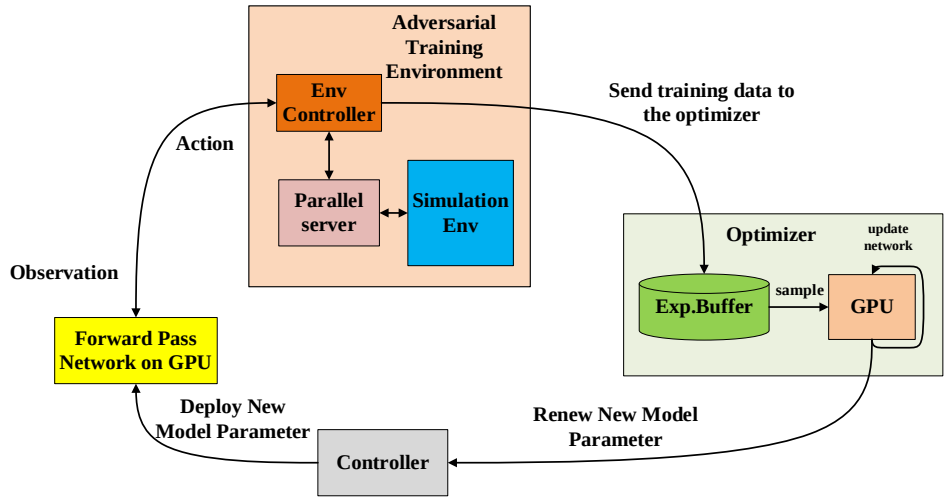


Figure 11. The Training System.

The agent acquires state observations from the air combat simulation environment, encodes them, and then generates maneuvering decisions through forward propagation. The agent is rewarded for changing its state by interacting with the air combat environment based on the acquired maneuver decisions. The data generated by this process is packaged, including the current agent state observations, maneuver decisions, rewards, and LSTM implicit layer $< S, A, R, H >$, into a sequence of 16 steps. The packed data are sent asynchronously to an experience replay pool built from Redis. The GPU of each optimizer samples the data in the experience replay pool to obtain the mini-batch and computes its gradient. The gradients are averaged across multiple optimizers using the MPI standard function of the NVIDIA NCCL2 protocol. The average gradient is returned to each GPU model to perform its gradient descent, ensuring that the GPU models on the distributed system are updated simultaneously.

In each small batch of data with 120 samples, each sample has a sequence of 16 actions and states combination pairs. During training, we use the Adam optimizer to update the model parameters by back-propagating the computed gradient over the samples of 16 actions in the small batch. A global statistician is also designed to determine the distribution of new versions to controllers by setting the number of version updates. Finally, distributed parallel processing enables the model to evolve dynamically and quickly, and global average gradients allow the agent to enhance performance steadily.

 doi:10.20944/preprints202305.0862.v1

3. Experimentation and Evaluation

This paper considers an experimental environment of a reinforcement learning-oriented air combat simulation platform developed by Tsinghua University, which establishes a digital twin air combat environment with a combat unit configuration, batch matchmaking, and other functions , as shown in **Figure 12**. The simulated aircraft is the F22 fighter, which carries a limited number of missiles, i.e., six AIM-120 medium-range missiles that rely on radar guidance homing heads. Besides, the missile's guidance chain can be provided by the aircraft. The experimental equipment is a 5950X + 3080TI host as the front-end that can open multiple adversarial environments simultaneously and establish a socket communication with the back-end model through a TCP/IP protocol. The adversarial environment has a parallel battlefield mechanism, each independently against the others. A single front end can run multiple parallel battlefields simultaneously, significantly increasing the data generation speed and accelerating the agent training process. The back-end host is a 8-way 3090 graphics server that samples the data from the experience pool in real-time and uses proximal policy optimization to calculate the gradient and mean squared deviation to update the Actor and Critic networks.

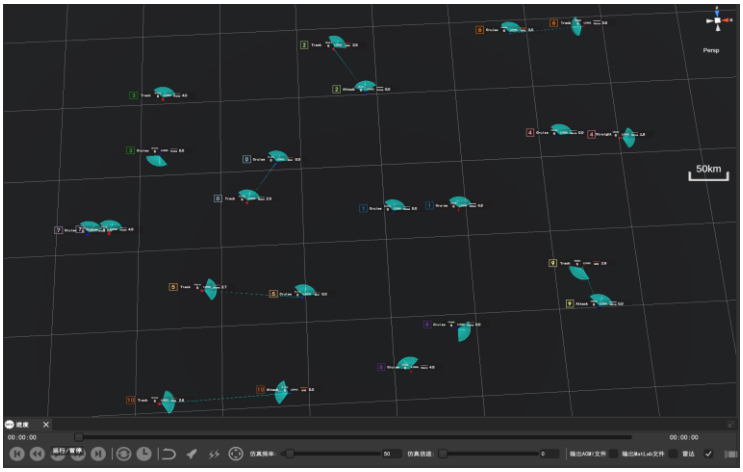


Figure 12. Agent training environment.

3.1. Comparison with expert-level bot

To reasonably evaluate the level of reinforcement learning trained agents, the experiments are conducted using a Finite State Machine (FSM) as a baseline. The FSM is modeled by an expert rule-based air combat state machine modeled on current state-of-the-art air combat knowledge. It is capable of performing complete engagements such as target acquisition, lock, shoot, and defensive tactics.

A total of three win rates are defined: final win rate, real-time win rate, and overall win rate. The final win rate is defined as the win rate per 1,000 matchups after convergence. Real-time win rate is defined as the statistical win rate per 100 games. The overall win rate is defined as the overall win rate from the initialization of the smart body to the final convergence. The specific win rate is calculated as shown in Equation 12, where C is the number of games after convergence. The sum of the rewards is the sum of the rewards of the sampled 120 sequences, and each sequence has rewards r brought by 16 actions.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

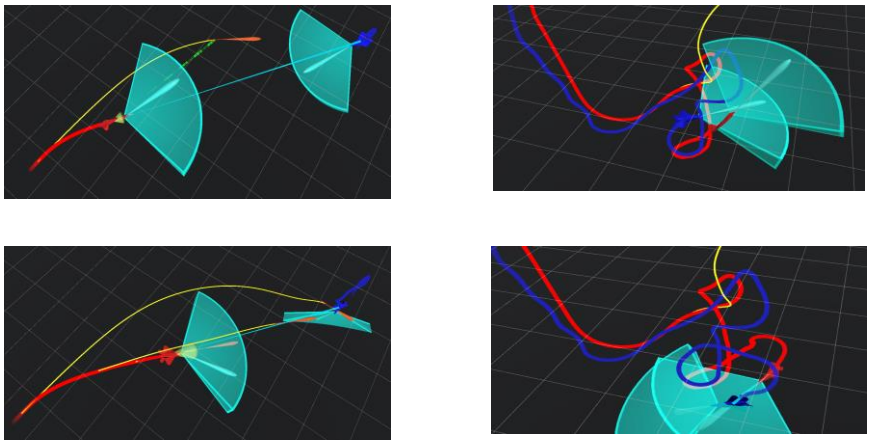
$$
\begin{aligned}
&W_i = \begin{cases} 1, & agent\ wins \\ 0, & agent\ loses\ or\ draws \end{cases} \\
&Final\ Win\ Rate{:}AW_j = \sum_{i=j}^{i+1000} \frac{W_i}{1000},\ (j > C) \\
&Real\ time\ Win\ Rate{:}\ RW_k = \sum_{i=k}^{i+100} \frac{W_i}{100}, \\
&Overall\ Win\ Rate{:}\ OW_i = \sum_{i=1}^{k} \frac{W_i}{k} \\
&Total\ Reward{:}\ R_{sum} = \sum_{i=1}^{120} \sum_{j=1}^{16} r_{ij}
\end{aligned}
\tag{15}
$$

As a comparison, Zhou et al. employed the predictive game tree AI against FSM and conducted 1152 confrontation experiments, where the predictive game tree algorithm achieved 700 wins, with only one tie. The predictive game tree agent achieved a 60.7% win rate against FSM. At the same time, the classic algorithms DQN, Deterministic policy gradient algorithms (DDPG) 42, Soft Actor-Critic (SAC)43 in reinforcement learning are also used to fight against FSM to calculate the final winning rate. The experimental results are shown in **Table 4**.

Table 4. The results of each algorithm against FSM

Algorithm compared	Win Rate	Lose Rate	Tie Rate
Predictive Game Tree	60.7%	39.1%	0.2%
DQN	35%	64%	1%
DDPG	40%	55%	5%
SAC	43%	53%	4%
MCLDPPO (ours)	**66%**	**24%**	**10%**

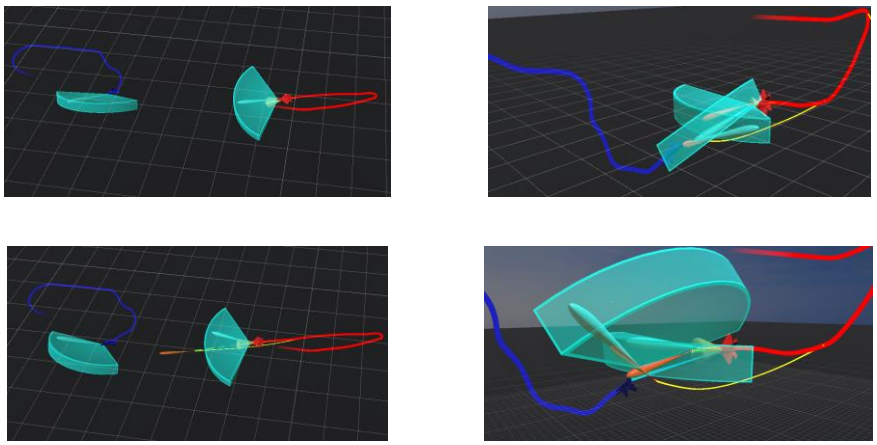
We did a large number of experiments to train the agent, using the MCLDPPO algorithm to train the reinforcement learning agent. The MCLDPPO agent converged after about 40,000 games against FSM, and after convergence the real-time win rate reached a maximum of 82%, the average win rate was about 66%, and the average tie rate was about 10%. The high average tie rate is because the state machine can fly directly out of the agent's attack range after firing all missiles. Moreover, it can be considered that when FSM loses its attack capability, the agent's average win rate exceeds 70%. Compared with the predictive game tree algorithm, our method is improved, and the reward function and action can be further refined, so there is substantial room to improve the agent.



(a) AI initiates attacks proactively

(b) AI gets posturing advantage

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1



(c) AI maintains the guidance chain (d) AI dives first and attacks

Figure 13. Trajectory results of air combat.

The high-resolution view of the training environment is removed in the actual training process to reduce system rendering overhead and improve system resource utilization. **Figure 13**(a) illustrates the agent launching a missile first as a feint to force the enemy to maneuver to avoid the missile by maintaining the guidance chain. A follow-up missile is fired to hit the enemy aircraft during a mass maneuver. In **Figure 13** (b), the red plane has a bad attitude, and playing a constant game with the blue plane gains an altitude advantage. In Figure 13(c), the aircraft actively seeks the enemy's view and thus seizes the opportunity to launch the missile early and maintain the missile's guidance chain to guide it. In Figure 13(d), the agent conducts the diving maneuver to make a quick turnaround and gain attitude advantage, forcing the enemy to conduct a mass maneuver by firing a missile and replacing it with another missile to hit the enemy.

3.2. Traditional Curriculum Learning

Bengio44 introduced the curriculum learning concept, a training idea that mimics the human learning process by starting with simple tasks and gradually increasing difficulty. Based on this concept, in this work, the agent is pitted against a straight-flying target aircraft, and the agent's behavior evolves by optimizing a distributed proximal policy optimization algorithm. The win rate can reach about 90%. After that, we pit the agent against the state machine and achieve a win rate of about 40%. Hence, we observe that the agent's defensive skills are directly discarded after its offensive skills converge. The state machine does not fly in a fixed radius like the target machine but can conduct many violent maneuvers to avoid the agent's offense. Thus, it is difficult for the agent who only learns offensive skills to fight against the state machine.

Loss of Plasticity is the most commonly criticized problem of deep neural networks. Indeed, a warm-starting experiment in 2020 demonstrated that better learning could only be achieved by training on the entire dataset in a one-time learning fashion, discarding what was initially learned. Recently, Sutton45 proposed a continuous backpropagation algorithm in a talk entitled "Maintaining Plasticity in Deep Continual Learning" at CoLLAs 2022. The main idea is to re-initialize some neurons with low utility so that diverse continuous injections can preserve the deep neural network’s plasticity. However, this is challenging to implement.

We mix the target and state machines in a specific ratio for training to solve this problem. The target machines' proportion is higher in the early stage of training and increases as the AI attack capability rise, as shown in Figure 14. By adopting this strategy, the AI's ability to fight against the state machine is significantly improved compared to the AI trained with only the target machine.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

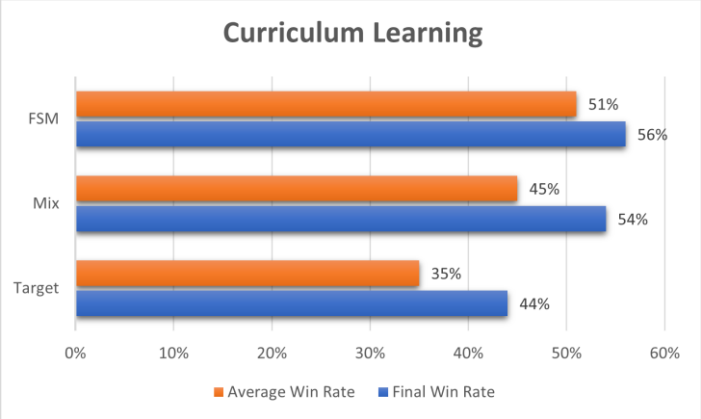


Figure 14. Curriculum Learning.

Nevertheless, since adding the state machine directly to the target machine provides an appealing performance, it is expected that using the state machine directly as a course teacher for the agent should work even better. When the agent directly fights against the state machine and improves its performance compared to the agent with mix training, its average win rate becomes 56% because the initial set of reward values is poorly designed. Specifically, the reward design is shown in Table 2. Therefore, the next section conducts a performance test for each reward to help the agent learn better.

3.3. Motivational Curriculum Learning

The previously used curriculum learning involved designing courses corresponding to skills aiming to help the agents learn. Since neural networks suffer from plasticity loss, they cannot build on existing foundations to learn new skills. Therefore, we reward different aspects for helping the agent strengthen their ability. Specifically, by observing the agent during the actual confrontation, the deficiencies should add guiding rewards to help it strengthen its ability.

For this strategy, first, we only add hit rewards as a benchmark for evaluating the merits of the subsequently added rewards. The hit reward is set to 100 points to standardize the reward scale. At the same time, the rewards and win rates for the agents using only hit rewards against the state machine are shown in, as shown in **Figure 15**-**Figure 16**

Since 120 sequences are collected simultaneously, and one sequence has 16 actions, each action has a corresponding reward. The reward curve is the sum of the rewards for all actions in 120 sequences. By observing the reward curve, we conclude that when the agent converges, the agent trained solely employing the rewards for missile hits on the enemy presented a 47% real-time win rate, calculated by averaging the win rates of the last 100 confrontations.

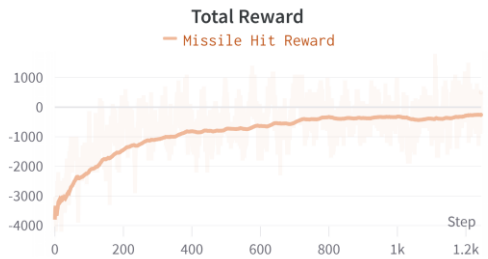


Figure 15. Total Reward.

Figure 16. Real-Time Win Rate.

In an actual air combat environment, the greatest threat to an aircraft is the enemy firing a missile. Therefore, we must reasonably assess the enemy missiles' threat to our aircraft. Hence, we develop two scenarios to assess the missile threat. The first one considers only the missile that poses the greatest threat to an aircraft at the current moment. The second scenario considers the sum of all missile threats. The threat value of a missile

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

is set to 20 points from an initially assumed baseline of 160 points. Simultaneously, to enable the agent to learn to conserve missiles, we deduct 5 points from the missile launch action, forcing the agent to learn to launch missiles at the correct location. No points are deducted for the missile launch as a comparison experiment. The following four experiments add a threat bonus to the missile hit bonus, and we use the agent's overall win rate to evaluate the appropriate course for the agent to learn, as shown in **Figure 17**.

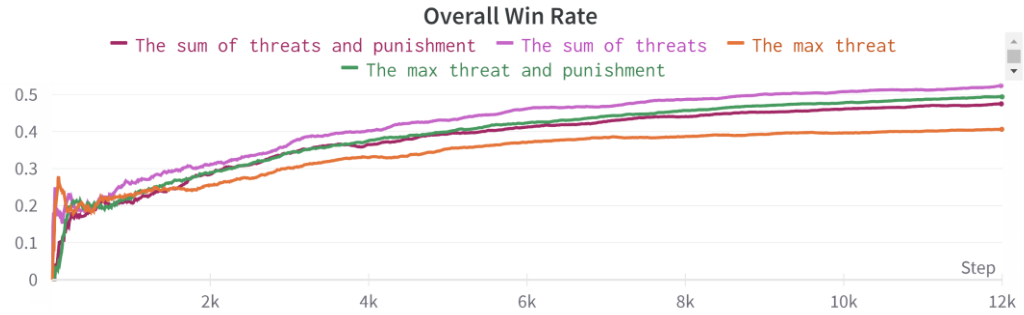


Figure 17. Overall win rate after adding threats.

The experiments reveal that using the threat sum reward affords the best performance. At the same time, after using the missile penalty, the threat sum and threat maximum values decrease, which maximize when using the maximum threat due to a problem using the threat maximum when the aircraft is in a better attitude. However, the threat value of the previously fired missile is larger than the threat of firing at the current moment, causing the agent to deduct points for firing the missile and thus affecting its actual combat capability. The agent achieve 60% win rate in real-time by adding a threat sum bonus to the missile hit bonus, as shown in **Figure 19**.

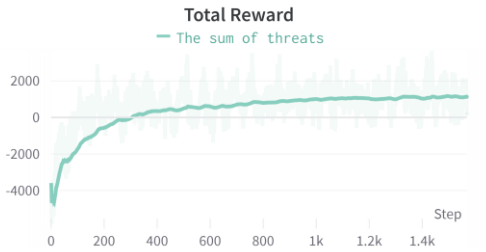


Figure 18. Total reward after adding threats

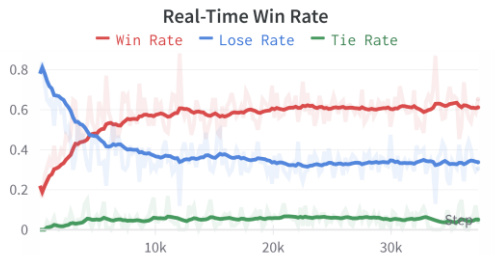


Figure 19. Real-Time win rate after adding threats

In addition to the missile threat bonus, the aircraft's attitude relative to the enemy aircraft is essential during the confrontation process. This is because if an aircraft has a superior attitude, it has an advantageous position to launch its missiles, significantly enhancing its winning chances. Therefore, the attitude bonus is added, which is set from the baseline of 80 to 30 points. By combining the missile hit and attitude bonuses, the training agent attains an improved win rate against the state aircraft, proving the effectiveness of adding the attitude bonus. We also evaluate other rewards, i.e., a close skimming bonus, a close dodging enemy missile bonus, a gaining enemy field of view bonus, a speed penalty bonus baseline, and a sideslip angle penalty, as shown in **Figure 20**. Table 3 reports that the bonuses improve only when agents use a missile hit bonus.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

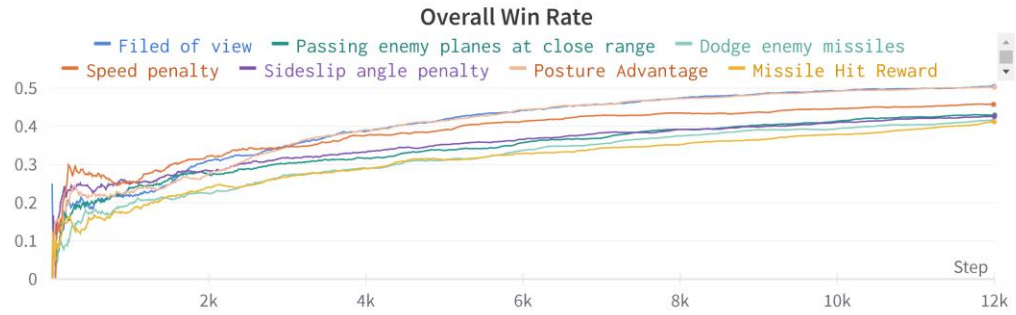


Figure 20. Overall win rate after adding individual rewards.

When using all favorable rewards to train the agent, the agent achieves a **66%** win rate and a 10% draw rate, as shown in **Figure 21**(b). The higher draw rate is mainly due to the aircraft running out of missiles and thus imposing a direct flight without confrontation. This situation occurs when the enemy aircraft loses its combat capability. In this case, the missile hit rate is 15%, and the end-guidance rate is 25%, similar to the actual air combat situation. When considering together, the trained agents' actual win rate exceeded 70%.

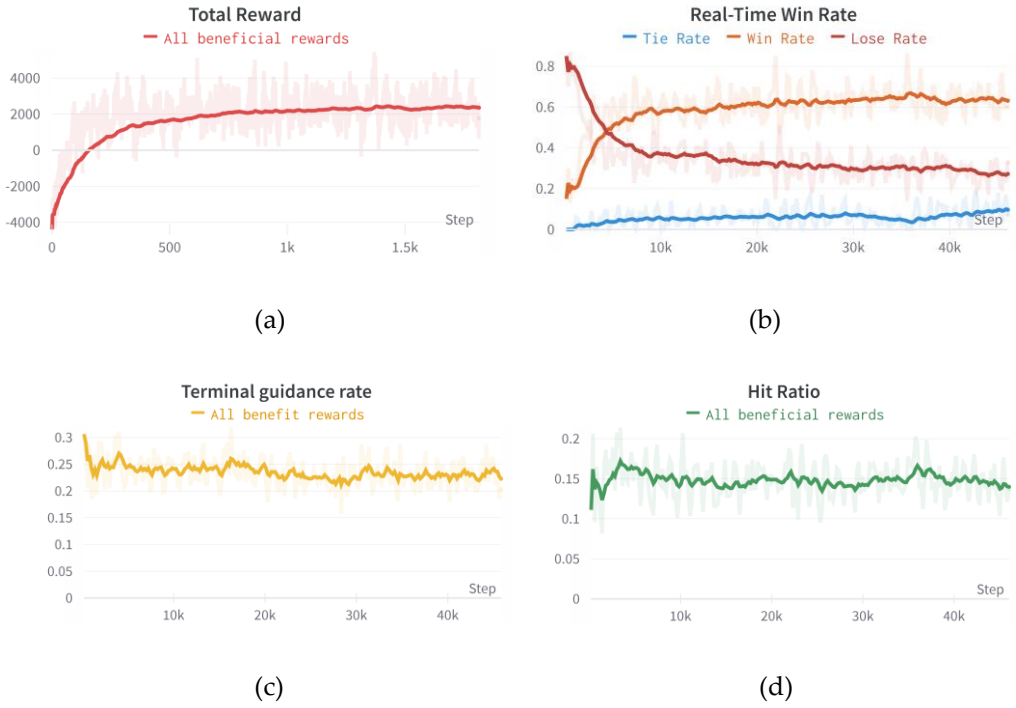


Figure 21. Reward results when combining all benefits.

The current agents' action parameters and types are relatively simple. Therefore, future works can explore layered reinforcement learning46 to expand the action parameters and types or allow the agents to control the number of rods directly, significantly expanding the agents' exploration space and potential. Nevertheless, expanding the movement space will exponentially increase the cost and time to train the agent.

Adding all valid rewards on the overall ability must be evaluated further by considering an ablation experiment. In the case of utilizing all rewards, their effect on the agent is reduced. **Figure 22** highlights that some of the reward reductions did not affect much performance. The rewards that have the most significant impact are the missile hit, missile threat, and posture rewards. Although the other rewards slightly improve performance from a data point of view, their actual match performance reveals that these rewards can improve their shortcomings.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

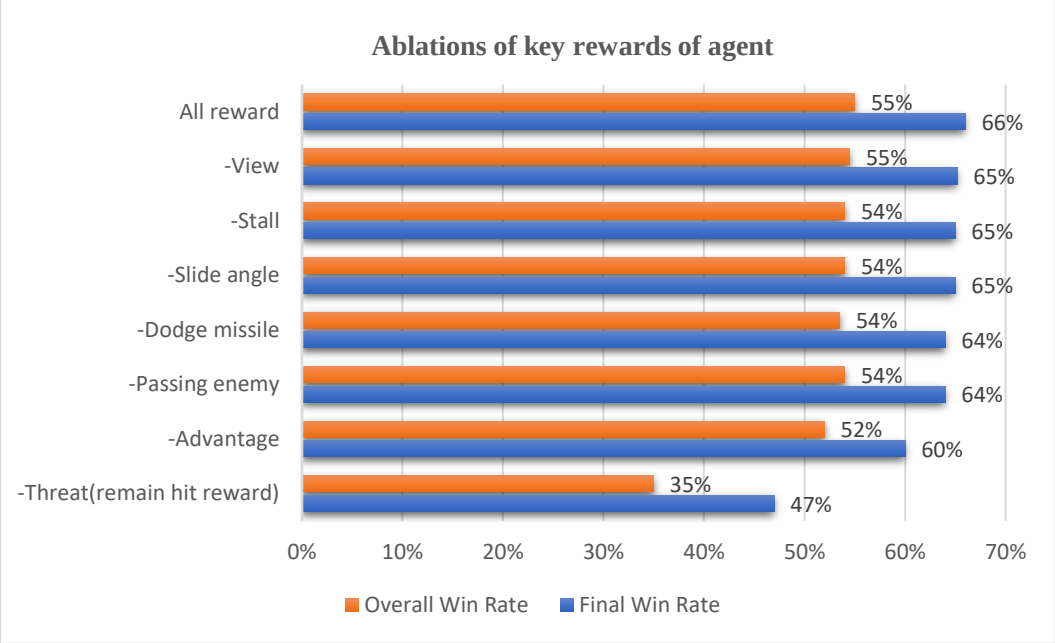


Figure 22. Ablation experiments for key rewards.

When using all the bonuses to train the agents from the actual confrontation process, in most cases, the attitude advantage can be an effective air combat confrontation. However, the win rate is not too high because all aircraft respawn randomly. Sometimes the state aircraft respawn might be in a good position when the agent can hardly avoid the missiles, even if it tries hard to dodge them. Agents trained using random location respawn are evaluated in a fixed location scenario. In this case, the win rate is around 80%. However, in an actual air combat environment, it is impossible to guarantee that all positions are relatively fair for dueling. Therefore, we use random position respawning to evaluate the agent's performance. The ablation experiment considering the rewards demonstrates that the added rewards positively affected the agent's training.

It is not only the guided rewards designed for motivational curriculum learning that affect the performance of the agent, but also the interruption mechanism designed for the agent. The interruption mechanism is designed to enhance the agent's decision-making ability in the face of an emergency situation when complex maneuver changes are required during actual air combat confrontations. The trigger for the interruption mechanism is a change in the number of threats to the agent, which is defined as a change in the number of enemy missile alerts, disappearance and reappearance of the enemy's field of view, and a change in the information available to his side. When the number of these different types of threats changes, the agent interrupts the current action and re-makes a maneuver decision using the current observations. Four algorithms were designed to do ablation experiments based on the bootstrap reward and interruption mechanisms, and the performance of the algorithms was evaluated by the final win rate. The comparison is illustrated in **Table 5**.

Table 5 The setting of ablation experiment

Algorithm	Hit Reward	Curriculum Reward	Interruption
MCLDPPO	●	●	●
MCLDPPO-R	●	○	●
MCLDPPO-I	●	●	○
Vanilla DPPO	●	○	○

Note: ● means that the mechanism is included, ○ means that the mechanism is not included.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

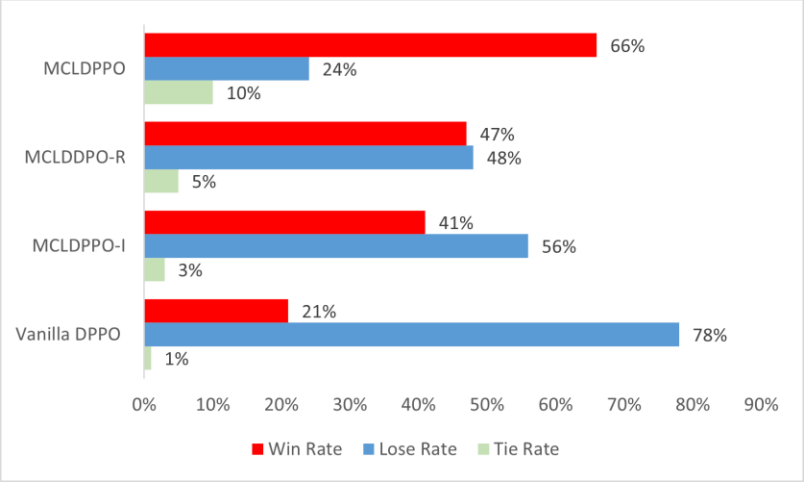


Figure 23. Performance comparison of different algorithms.

Figure 23 shows a comparison of the performance of the experimental algorithms. The addition of the course reward bootstrap and the interrupt mechanism to the traditional DPPO algorithm both have a large degree of improvement in the ability of the agents, with some differences in the degree of impact due to the different mechanisms at play. Specifically, the Vanilla DPPO trained agent with only hit rewards has difficulty against the FSM, and can beat the FSM only if the random position initialization is better. the MCLDPPO-I trained agent without the interruption mechanism is better in long-range confrontation, but appears to have insufficient decision frequency in close-range confrontation. For MCLDPPO-R, which lacks a course reward, the MCLDPPO-R trained agent can only barely beat the FSM about evenly because there is only a hit reward, the global reward is too sparse, and there is a lack of clarity for many favorable moves. MCLDPPO combining course reward and interruption mechanism can beat FSM with a relatively large advantage, which verifies the effectiveness of the proposed algorithm and can better solve the decision problem of single aircraft air combat confrontation.

3.4. Impact of network structure

Figure 24 illustrates the network's architecture evolution. Initially, the network a solely relied on local observations, and the Value Network cannot accurately evaluate the situation from the non-perfect information of the current agent compared to using the perfect information rating. Therefore, network b combines global inputs and local observations to give the Value Network an output rating of the current situation. Deep LSTM does not extract the global observation output in network b, and since the network design is deep, its training process challenging. Therefore, a relatively shallow network architecture c is designed and the global and local observables are encoded independently by Deep LSTM.

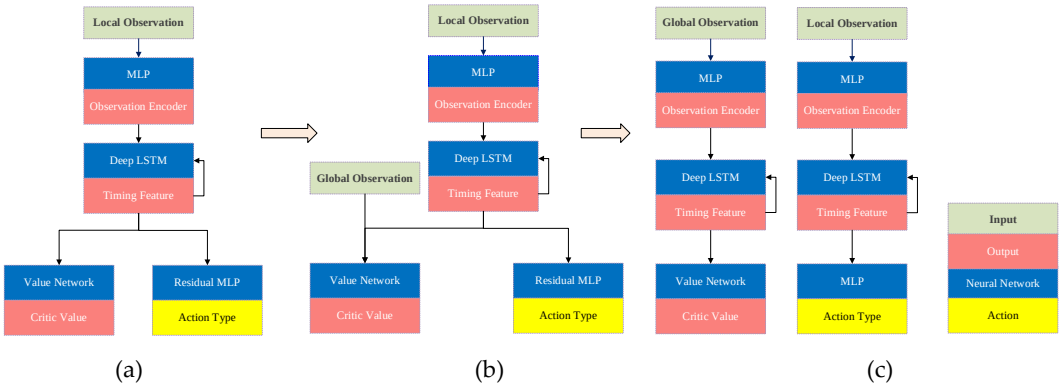


Figure 24. Network Architecture Evolution Process.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

The performance of the network architecture was evaluated by using these three networks to train the agent against the FSM. When the agent's capabilities cannot be improved, we determine that the network has converged at that point. As can be seen from **Table 6**, the agent trained by network c not only have better final win rates than networks a and b, but also have far less convergence time.

Table 6. Performance of different network architectures

	Win Rate	Lose Rate	Tie Rate	Convergence Time
Network a	57%	37%	6%	29h
Network b	60%	32%	8%	34h
Network c(ours)	66%	24%	10%	**20h**

Regarding the interplay between the number of LSTM layers and performance, it should be noted that handling the agent's local observations is important. To overcome that concern, the forward network first encodes the observations before passing them on to the deep LSTM network for processing. We conduct various tests to evaluate the impact of the correct number of LSTM layers, with **Figure 25** presenting the results. **Figure 25** highlights that the initial win rate of the LSTM network exceeds the network without LSTM, and the LSTM network with two layers performs better than using deeper LSTM architectures. Overall, training LSTMs with more than three layers is challenging and significantly increases the training expenses and duration.

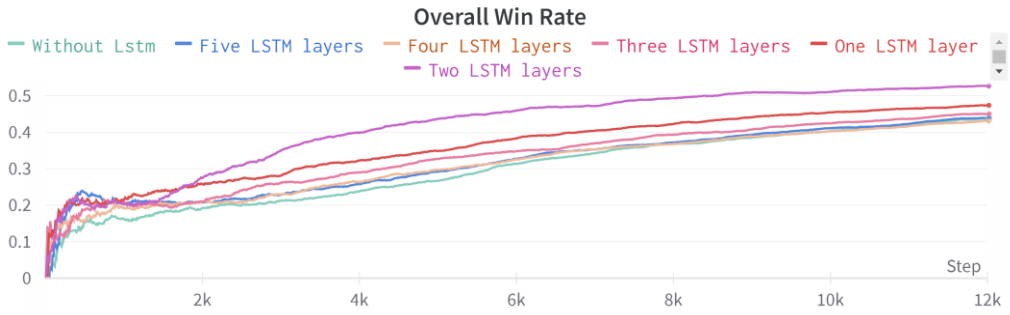


Figure 25. Performance impact of the number of layers of the LSTM network.

Stored state and Burn-in47 processing are the two main methods used in LSTM implicit layer processing.

Store State: The hidden layers of LSTM-based networks must be saved for training. All intermediate hidden layers and the trajectory data in a scene are stored while the network is initialized with a zero initial state during the initial sequence creation. The network is initialized during training using the preserved implicit layers, compensating for the drawback using the direct zero-state approach. However, if the implicit layers produced by a sufficiently ancient network are sampled, they might be biased because they differ too much from the present ones.

Burn-in: Since the sequence comprises 16 actions, some problems using Store State exist. Thus, we use the stored implicit layer state as the starting state to initialize the network. The subsequent networks constantly re-update the implicit layer state, so the network accurately outputs the state of the hidden layer. **Figure 26** illustrates an experimental comparison of the network using Burn-In concerning Store State, demonstrating a slight improvement. Since the Buffer size in the experimental environment is relatively small, the gap between the two models is small, and their performance difference is negligible.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

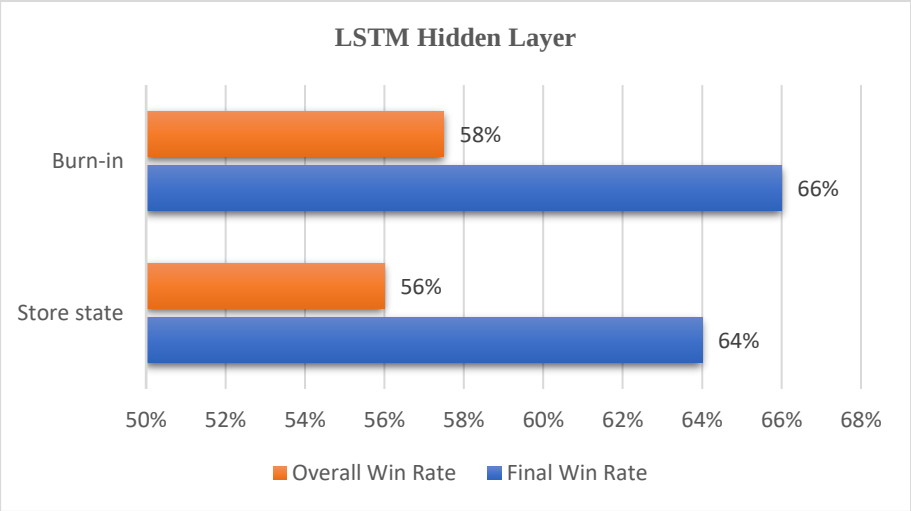


Figure 26. Results for different hidden layer usage

3.5. Impact of key parameters

The real performance and training speed during the actual training are impacted by many hyperparameters. Thus, this section conducts small-scale experiments to assess how several important parameters affect the agent's actual performance. **Table 7** lists the hyperparameters applied.

Table 7. Parameters in the reinforcement learning phase

Parameters	Setting
Number of optimizer GPUs	5
Batch Size	120
Total Buffer Size	480
Decay factor	0.99
GAE λ	0.95
PPO clipping	0.2
Sample Reuse	1
Model update per Iteration	2
Actor learning rate	5e-4
Critic learning rate	2e-4
Entropy coefficient	0.01

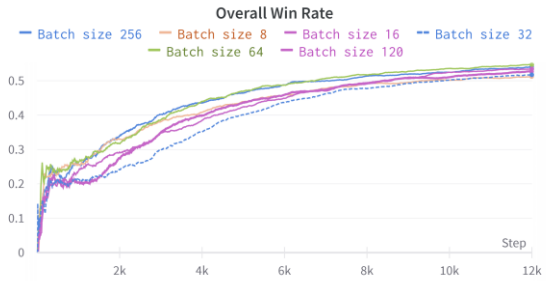
Batch Size: Increasing the batch size during the training process requires adding more rollouts to the experience pool and more GPU resources to optimize the larger batches. Simultaneously, utilizing a bigger batch size can speed up the model's convergence, and theoretically, using twice the GPU requires half the training time. In practice, the linear acceleration impact only begins at the beginning of training, and the acceleration effect reduces with the subsequent iterations, as shown in Figure 27(a). However, increasing the batch size saves a significant amount of training time.

Data Quality: Data reuse is defined as the number of times data can be reused, and a usage rate of 1 indicates that only the data produced by the current model are used. Data provided by the prior model affect the model's maximum capacity. Although the top limit is greater for the 0.5 case than the 1 case, the former requires twice the time for practical testing, which is costly, as shown in Figure 27(b).

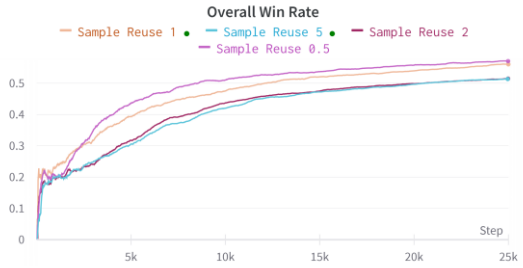
Impact of sequence length: Multiple adversarial settings are simultaneous in a distributed environment, each involving numerous parallel battlefields, generating many state-action reward pairings. In the backpropagation utilizing GAE to determine the action advantage, the advantage must be truncated at the proper place, raising the question of how many actions comprise a more sensible sequence. In this plenary experiment, the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 May 2023 doi:10.20944/preprints202305.0862.v1

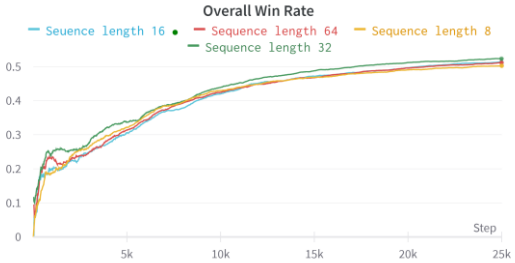
default sequence length is 16 actions. As depicted in the Figure 27(c), excessively long or short sequences are not very beneficial to practical training.



(a) Performance of different Batch sizes



(b) Performance of different sample reuse



(c) Performance of different sequence lengths

Figure 27. Effect of different hyperparameters on agent performance.

4. Conclusions

This paper studies deep reinforcement learning based on single UAV aerial combat. We propose a distributed proximal policy optimization algorithm based on motivational curriculum learning to solve to solve the problem of neural network plasticity loss in traditional course learning. Agents trained by MCLDPPO can beat expert systems by a large margin, and outperform predictive game tree and mainstream reinforcement learning methods.

The main idea of the motivational course learning is to help the agent to gradually improve its own combat ability by observing the inadequacy of the agent's actual air combat capability and providing corresponding rewards as a guide. To solve the problem that the action space is too large, a whole set of tactical maneuvers is encapsulated based on the existing knowledge of air combat, and some tactics beyond human knowledge can be realized through rational use of these maneuvers. At the same time, by designing an interruption mechanism for the agent, the decision-making frequency of the agent is improved when it faces an emergency. When the number of threats perceived by the agent changes, it will interrupt the current action to re-acquire observations and make maneuvering decisions again. By using the interruption mechanism, the ability of the agent can be greatly improved. To simulate the real air combat confrontation better, we build a simulation environment close to a real air combat scenario using the digital twin technique. The simulation environment fully simulates the main components of air warfare-missiles, radar, optoelectronics, and infrared.

Moreover, we propose a distributed proximal policy optimization-based air combat agent framework that fully exploits the distributed architecture features, enabling multiple adversarial environments simultaneously. Furthermore, our method sends the data generated by each adversarial environment to the Redis cache asynchronously via a TCP/IP protocol, exponentially increasing the data throughput and accelerating the agent's training. We code the agent's local and global observations for the complex air warfare observation environment. Besides, an Actor-Critic network architecture using

 doi:10.20944/preprints202305.0862.v1

deep LSTM is designed where the Actor and Critic use separate networks, and the Actor makes decisions by inputting local observations. In contrast, the Critic uses global observations to determine the value of the current state.

Based on the idea of motivational course learning, by observing the actual performance of the agents in air combat, the corresponding rewards are given to guide them, and the experiments prove that all added rewards are instructive to the agents. We also analyze the influence of network structure and hyperparameters on the training process. Moreover, we suggest a reasonable network structure and a set of appropriate hyperparameters. Our experimental findings suggest that the agent's capability trained by reinforcement learning substantially exceeds the performance of the FSM. If the initial poses are fair, the agent already has a significant advantage, which verifies the effectiveness of this paper's algorithmic framework.

Due to computational resources and training time cost limitations, the currently available actions are not comprehensive enough. Therefore, we aim to enhance the agent's action space in future research. The idea of hierarchical reinforcement learning will be used to expand the action space, which accurately restores the real situation of actual air combat and improves the agent's combat capability.

Author Contributions: Conceptualization, J.Z. and W.Z.; methodology, J.Z.; software, J.Z and W.Z.; validation, J.Z., W.Z. and M.K.; formal analysis, J.Z.; resources, Jihong.Z.; data curation, J.Z.; writing—original draft preparation, J.Z.; writing—review and editing, H.S., M.K., H.Z. and X.L.; visualization, J.Z.; supervision, X.H.; project administration, M.K. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: The data used to support this study have not been made available.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ryan, H. Air Combat Evolution (ACE) [Internet]. 2019 Jun[cited 2023 May 1]. Available from: https://www.darpa.mil/program/air-combat-evolution.
2. Sun, Z.; Yang, S.; Piao, H. A survey of air combat artificial intelligence. *Acta Aeronautica Sinica*, **2021**, *42(8)*:525799.
3. Burgin, G.; Owens, A. An adaptive maneuvering logic computer program for the simulation of one-to-one air-to-air combat. *Volume 2: Program Description*. NASA, 1975:5-10.
4. Burgin, G. Improvements to the adaptive maneuvering logic program. NASA,1986:2-6.
5. Mcmanus, J.; Goodrich, K. Application of artificial intelligence (AI) programming techniques to tactical guidance for fighter aircraft. *Guidance, Navigation and Control Conference*. 1989: 3525.
6. Osborne, M.; Rubinstein, A. *A course in game theory*. MIT Press, 1994:1-2.
7. Xu, G.; Liu, Q.; Zhang, H. The application of situation function in differential game problem of the air combat. *Chinese Automation Congress (CAC). IEEE*, 2018: 1190-1195.
8. Virtanen, K.; Karelahti, J.; Raivio, T. Modeling air combat by a moving horizon influence diagram game. *Journal of guidance, control, and dynamics*, 2006, *29(5)*: 1080-1091.
9. Park, H.; Lee, B.; Tahk, M. Differential game based air combat maneuver generation using scoring function matrix. *International Journal of Aeronautical and Space Sciences*, 2016, *17(2)*: 204-213.
10. Alkaher, D.; Moshaiov, A. Dynamic-escape-zone to avoid energy-bleeding coasting missile. *Journal of Guidance, Control, and Dynamics*, **2015**, *38(10)*: 1908-1921.
11. Shachter, R. Evaluating influence diagrams. *Operations research*, **1986**, *34(6)*: 871-882.
12. Koller, D.; Milch B. Multi-agent influence diagrams for representing and solving games. *Games and economic behavior*, 2003, *45(1)*: 181-221.
13. Virtanen, K.; Raivio, T.; Hamalainen, R. Modeling pilot's sequential maneuvering decisions by a multistage influence diagram. *Journal of Guidance, Control, and Dynamics*, **2004**, *27(4)*: 665-677.
14. Pan, Q.; Zhou, D.; Huang, J. Maneuver decision for cooperative close-range air combat based on state predicted influence diagram. *IEEE International Conference on Information and Automation (ICIA). IEEE*, **2017**, 726-731.
15. Zheng, H.; Deng, Y.; Hu, Y. Fuzzy evidential influence diagram and its evaluation algorithm. *Knowledge-Based Systems*, 2017, *131*: 28-45.
16. Sprinkle, J.; Eklund, J.; Kim, H. Encoding aerial pursuit/evasion games with fixed wing aircraft into a nonlinear model predictive tracking controller. *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601). IEEE*, 2004, *3*: 2609-2614.
17. Kaneshige, J.; Krishnakumar, K. Artificial immune system approach for air combat maneuvering. *Intelligent Computing: Theory and Applications V. SPIE*, **2007**, *6560*: 68-79.
18. Wang, Y.; Huang, C.; Tang, C. Research on unmanned combat aerial vehicle robust maneuvering decision under incomplete target information. *Advances in Mechanical Engineering*, **2016**, *Vol. 8(10)*: 1–12.
19. Ji, H.; Yu, M.; Han, Q. Research on the Air Combat Countermeasure Generation Based on Improved TIMS Model. *Journal of Physics: Conference Series. IOP Publishing*, **2018**, *1069(1)*: 012039.
20. Huang, C.; Dong, K.; Huang, H. Autonomous air combat maneuver decision using Bayesian inference and moving horizon optimization. *Journal of Systems Engineering and Electronics*, 2018, 29(1): 86-97.
21. McGrew, J. Real-time maneuvering decisions for autonomous air combat. *Massachusetts Institute of Technology*, 2008:3-10.
22. Geng, W.; Ma, D. Study on tactical decision of UAV medium-range air combat. *The 26th Chinese Control and Decision Conference (2014 CCDC). IEEE*, 2014: 135-139.
23. Zuo, J.; Yang, R.; Zhang, Y.; Li, Z. Intelligence decision-making in air combat maneuvering based on heuristic reinforcement learning. *Acta Aeronautica et Astronautica Sinica*, 2017, 38(10): 321168.
24. Ping, L.; Yao, M. A deep reinforcement learning based intelligent decision method for UCAV air combat. *Asian Simulation Conference. Springer, Singapore*, 2017: 274-286.
25. Yang, Q.; Zhu, Y.; Zhang J. UAV air combat autonomous maneuver decision based on DDPG algorithm. *2019 IEEE 15th International Conference on control and automation (ICCA). IEEE*, 2019: 37-42.
26. Xie, L.; Ding, D.; Wei, Z.; Xi Z.; Tang A. Moving Time UCAV Maneuver Decision Based on the Dynamic Relational Weight Algorithm and Trajectory Prediction. *Mathematical Problems in Engineering*, 2021:25-31.
27. Pope, A.; Ide, J.; Mićović, D. Hierarchical reinforcement learning for air-to-air combat. *2021 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE*, 2021: 275-284.
28. Liu, T.; Tian, B.; Ai, Y.; Li, L.; Cao, D.; Wang, F. "Parallel Reinforcement Learning: A Framework and Case Study," *IEEE/CAA J. Autom. Sinica*, **July 2018**, *vol. 5, no. 4*, pp. 827-835.
29. Shi, W.; Feng, Y.; Cheng G. Research on multi-aircraft cooperative air combat method based on deep reinforcement learning. *Acta Automatica Sinica*, **2021**, *47(7)*: 1610−1623.
30. Piao, H.; Sun, Z.; Meng, G. "Beyond-Visual-Range Air Combat Tactics Auto-Generation by Reinforcement Learning," 2020 *International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1-8.
31. Ma, Y.; Du, X.; Sun X. Adaptive modification of turbofan engine nonlinear model based on LSTM neural networks and hybrid optimization method. *Chinese Journal of Aeronautics*, **2022**, *35(9)*: 314-332.

32. Schulman, J.; Wolski, F.; Dhariwal, P. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
33. Zhou,W.; Zhu, J.; Kuang M. Multi-UAV cooperative swarm algorithm in air combat based on predictive game tree. *Sci Sin Tech*, **2016**, *46*: 79–90.
34. Mnih, V.; Kavukcuoglu.; K, Silver, D. Human-level control through deep reinforcement learning. *Nature*, **2015**, *518(7540)*: 529-533.
35. Berner, C.; Brockman, G.; Chan, B. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680, 2019.
36. Vinyals, O.; Babuschkin, I.; Czarnecki, W. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, **2019**, *575(7782)*: 350-354.
37. Schulman, J.; Levine, S.; Abbeel, P. Trust region policy optimization. *International conference on machine learning. PMLR*, 2015: 1889-1897.
38. Schulman, J.; Wolski, F.; Dhariwal P. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
39. Schulman, J.; Moritz, P.; Levine, S. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438, 2015.
40. Zhou,W.; Zhu, J.; Kuang M. An unmanned air combat system based on swarm intelligence. *Sci Sin Tech*, **2020**, *50(3)*: 363-374.
41. McGrew,J.; How J.; Williams B. Air-combat strategy using approximate dynamic programming. *Journal of guidance, control, and dynamics*, **2010**, *33(5)*: 1641-1654.
42. Silver, D.; Lever, G.; Heess, N. Deterministic policy gradient algorithms. *International conference on machine learning. PMLR*, 2014: 387-395.
43. Haarnoja, T.; Zhou, A.; Abbeel, P. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International conference on machine learning. PMLR*, 2018: 1861-1870.
44. Bengio, Y.; Louradour, J.; Collobert, R. Curriculum learning. *Proceedings of the 26th annual international conference on machine learning*. 2009: 41-48.
45. Richard, S. Maintaining Plasticity in Deep Continual Learning. *Conference on Lifelong Learning Agents*; 2022, August 22-24; McGill University, Canada, Montreal; 2022.
46. Barto, A. Mahadevan S. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, **2003**, *13(1)*: 41-77.
47. Kapturowski, S.; Ostrovski, G.; Quan, J. *Recurrent experience replay in distributed reinforcement learning. International conference on learning representations.* **2018**.