entropy

MDPI

Article

Hierarchical Reinforcement Learning Framework in Geographic Coordination for Air Combat Tactical Pursuit

Ruihai Chen [1], Hao Li [2], Guanwei Yan [2], Haojie Peng [1] and Qian Zhang [3,*]

[1] School of Aeronautics, Northwestern Polytechnical University, Xi'an 710072, China; crh@mail.nwpu.edu.cn (R.C.); sc4979@163.com (H.P.)
[2] Chengdu Aircraft Design and Research Institute, Chengdu 610041, China
[3] School of Aerospace, Northwestern Polytechnical University, Xi'an 710072, China
* Correspondence: snzq2020@nwpu.edu.cn

Abstract: This paper proposes an air combat training framework based on hierarchical reinforcement learning to address the problem of non-convergence in training due to the curse of dimensionality caused by the large state space during air combat tactical pursuit. Using hierarchical reinforcement learning, three-dimensional problems can be transformed into two-dimensional problems, improving training performance compared to other baselines. To further improve the overall learning performance, a meta-learning-based algorithm is established, and the corresponding reward function is designed to further improve the performance of the agent in the air combat tactical chase scenario. The results show that the proposed framework can achieve better performance than the baseline approach.

Keywords: hierarchical reinforcement learning; meta-learning; reward design; decision

check for updates

Citation: Chen, R.; Li, H.; Yan, G.; Peng, H.; Zhang, Q. Hierarchical Reinforcement Learning Framework in Geographic Coordination for Air Combat Tactical Pursuit. *Entropy* **2023**, *25*, 1409. https://doi.org/10.3390/e25101409

Academic Editor: Wei Li

Received: 1 August 2023
Revised: 24 September 2023
Accepted: 25 September 2023
Published: 1 October 2023

1. Introduction

The application of reinforcement learning (RL) [1,2] in aerial combat has attracted a lot of attention in recent years, and RL has been used to simulate the behavior of pilots and aircraft and to optimize aerial combat strategies [3,4].

Challenges related to these simulations include establishing the interaction model between pilots and aircraft [5,6]; simulating the behavior of pilots maneuvering the aircraft and its impact [7]; introducing enemy aircraft and weapons; simulating the behavior of the enemy aircraft and its impact [8]; and the simulation of multi-aircraft cooperative combat behavior [9,10]. Of these, confrontation behavior in air combat is complex and variable, with various modes [11], and it is difficult for traditional methods such as state machines and differential games to completely characterize the real-time decision-making state of pilots and devise further optimization according to different situations [12,13]. However, by modeling the air combat process as a Markov process [14], reinforcement learning methods can achieve continuous optimization of decision-making algorithms [15,16].

The first application of RL in aerial combat was proposed by Kaelbling et al. [17]. They proposed a model-based RL approach for controlling an unmanned aerial vehicle (UAV) in a simulated air-to-air combat environment. The UAV was equipped with a simulated radar and missile system, and the RL agent was trained to select the optimal action for the UAV to maximize its chances of survival. The results showed that the RL agent was able to outperform the baseline agent in terms of survival rate. More recently, Hu et al. [18] trained long and short-term memory (LSTM) in a deep Q-network (DQN) framework for air combat maneuvering decisions, and this was more forward-looking and efficient in its decision-making than fully connected neural-network- and statistical-principle-based algorithms [19]. In addition, Li proposed a deep reinforcement learning method based on proximal policy optimization (PPO) to learn combat strategies from observation in an end-to-end manner [20,21], and the adversarial results showed that his PPO agent can beat

the adversary with a win rate of approximately 97%. Based on the deep deterministic policy gradient algorithm framework, Lu designed and implemented an air warfare decision policy and improved the efficiency of the training process via a preferred experience playback strategy [22]. This method was able to achieve fast convergence while saving training costs.

Because of the sparse nature of the air combat environment, the shaping of the reward function has been a key challenge in the application of reinforcement learning to air combat [23,24]. Piao constructed a high-fidelity air combat simulation environment and proposed a critical air combat event reward-shaping mechanism to reduce episodic win–lose signals [25,26], enabling fast convergence of the training process. The implementation results showed that reinforcement learning can generate a variety of valuable air combat tactical behaviors under beyond-visual-range conditions. Hu et al. [27] designed a reward function based on the original deep reinforcement learning method, and the design dimension of the reward included the real-time gain due to the maneuver as well as the final result gain. For the air combat maneuver decision problem with sparse rewards, Zhan et al. [28–30] applied a course-based learning approach to design a decision course of angle, distance, and mixture which improved the speed and stability of training compared to the original method without any course and was able to handle targets from different directions.

In the air combat decision-making process, the combination of various independent states forms a very large situation space which leads to an explosion of state dimensions [31]. Current research focuses on the rationality of the decision logic after the introduction of reinforcement learning training in a specific scenario [32,33], whereas this paper focuses on making the existing decision algorithm rapidly scalable as more and more realistic situations are introduced to quickly adapt to a more realistic air combat countermeasure environment [34,35]. The state space curse of dimensionality problem often leads to insensitivity in the model tracking which eventually fails to converge to a better stable tracking state. Therefore, in this paper, a hierarchical reinforcement learning (HRL)-based air warfare framework is proposed [36], which uses a hierarchical reinforcement learning structure to implement three-dimensional air warfare. Experimental results show that the proposed framework can achieve better performance than existing methods. The main innovations of this study are as follows:

1. We propose a hierarchical reinforcement learning framework in geographic coordination for the training and use of senior and basic policies to solve the MDP in air combat chase scenarios.
2. We propose a meta-learning algorithm applied to the framework proposed in this paper for the complex sub-state and action space learning problem of air warfare. The reward decomposition method proposed in this paper also alleviates the problem of reward sparsity in the training process to some extent.
3. We independently built a three-degrees-of-freedom air combat countermeasure environment and modeled the task as a Markov process problem. Specifically, we defined the key elements of the Markov process, such as state, behavior, and reward functions for this task.
4. We established a quantitative system to evaluate the effectiveness of reinforcement learning methods for training in 3D air combat.

In Section 2, we describe the application of reinforcement learning algorithms to the established air combat environment. In Section 3, we present the algorithm framework, reward function design ideas, algorithm training, and usage process. In Section 4, we establish a standard evaluation method and compare multiple SOTA models. In Section 5, we discuss the experimental results and in Section 6, we summarize the whole paper.

2. Reinforcement Learning for Air Combat

This paper sets out a design for a hierarchical RL algorithm capable of learning effective decision strategies in air combat countermeasure scenarios through interaction

with a simulated environment. The core of the algorithm is the use of Markov decision processes (MDPs) to model the decision process of combat aircraft in the presence of uncertainty and dynamic adversaries [37,38]. In this context, the design of MDPs requires careful consideration of factors such as state space representation, action selection, and reward function design. In addition, the construction of realistic and challenging combat environments is critical to evaluate the performance of the HRL algorithms constructed in this paper [39,40].

2.1. Markov Decision Process

Figure 1 describes the feedback loop; each of the subscripts t and $t + 1$ representing a time step refers to a different state: the state at moment t and the state at moment $t + 1$. Unlike other forms of learning, such as supervised and unsupervised learning, reinforcement learning can only be thought of as a series of sequential state–action pairs [41].

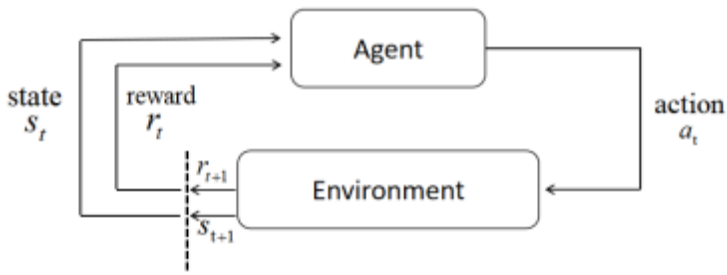

Figure 1. Markov process model.

Intelligence in reinforcement learning requires information from the current state s_{t+1} and also from the previous state s_t to make the best decision that maximizes its payoff [42]. A state signal is said to have Markovianity if it has the information necessary to define the entire history of past states.

Markov decision processes (MDPs) represent decision-makers who periodically observe systems with Markovianity and make sequential decisions [42,43]. They are the framework used for most problems in reinforcement learning. For each state s and action a, the probability that the next state s' may occur is

$$P^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \tag{1}$$

where P denotes the transfer probability, meaning the possible change of air combat situation when a certain behavior a is executed in a specific state s. In this paper, the value of P is fixed, and the expectation value of the next reward value R can be determined as

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s')\right], \forall s \in S, \forall a \in A \tag{2}$$

Intelligence tries to maximize its payoff over time, and one way to achieve this is to optimize its strategy. A strategy π is optimal when it produces better or equal returns than any other strategy, and π specifies the probability distribution of executing a certain decision action in a given air combat situation. The equation for state values states that at any state, strategy π is better than π' if $V^\pi(s) \geq V^{\pi'}, \forall s \in S$. The state value function and the state action value function can be optimized according to the following two equations:

$$V^*(s) = \max_\pi V^\pi(s), s \in S \tag{3}$$

$$Q^*(s,a) = \max_\pi Q^\pi(s,a), \forall s \in S, \forall a \in A \tag{4}$$

The above two equations can calculate the optimal state value $V^*(s)$ and the optimal action value $Q^*(s,a)$ when using the strategy π. The Bellman optimal equation for $V^*(s)$

can be used to calculate the value of states when the reward function $R^a_{ss'}$ and the transfer probability $P^a_{ss'}$ are known without reference to the strategy; similarly, the Bellman optimal equation constructed with the state action value function can be used as follows:

$$\begin{aligned} V^*(s) &= \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \mathcal{W}^*(s') \right] \\ Q^*(s,a) &= \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a'} Q^*(s',a') \right] \end{aligned} \tag{5}$$

The above two equations can calculate the optimal state value $V^*(s)$ and the optimal action value $Q^*(s,a)$ when using the strategy π. Additionally, in the case of no reference strategy, when the reward function $R^a_{ss'}$ and the transfer probability $P^a_{ss'}$ are known, the Bellman optimal equation of $V^*(s)$ can be used to calculate the value of states, representing the expected cumulative returns associated with being in a given situation and subsequently following the best decision strategy throughout the air combat. The Bellman optimal equation constructed with the state action value function can also be used.

2.2. Air Combat Environmental Model

The defined air combat adversarial environment for the MDP is implemented as two simulators $\text{Simu}_i, i \in \{Horizontal, Vertical\}$, where $(S_{next}, R_i) = \text{Simu}_\text{i}(S_i, A_i)$ and A_i is the action of Agent i in state S_i [44]. The simulator $\text{Simu}_{(i)}$ receives the action A_i and then produces the next state S_i and the reward R_i, where the state space S_i consists of the coordinates (x,y,z), velocity v and acceleration Δ of the red and blue sides under the geographic coordinate system:

$$\text{S} = (x_r, y_r, z_r, v_{x_r}, v_{y_r}, v_{z_r}, \Delta_{x_r}, \Delta_{y_r}, \Delta_{z_r}, x_b, y_b, z_b, v_{x_b}, v_{x_r}, v_{y_b}, v_{z_b}, \Delta_{x_b}, \Delta_{y_b}, \Delta_{z_b},) \tag{6}$$

In the next state, the geometric position, the spatial positions of the tracker, and the target are updated after the input actions [45,46]. The action space in *Horizontal* space are discrete, and they are defined as three different actions: forward, left, and right. Again, action space *Vertical* is defined as three different actions: up, hold, and down. In addition, we specifically set rules on height for this simulation to match realistic scenarios so that, during training, if the tracker moves beyond the restricted height range, the simulator limits its further descent or ascent and then receives a new movement [47]. We define rewards R_i for the corresponding environment,$i \in \{Horizontal, Vertical\}$. The role of the reward function is to encourage the tracker to continuously track the movement of the target. It is defined as follows:

$$R_i = \omega_1 SOT(f_{target}, f_{state}) \tag{7}$$

where ω_1 is a parameter and is a positive parameter, f_{target} represents the real position and velocity of the target, and fstate represents the current position and velocity of the tracker. SOT represents the status of tracking between f_{state} and f_{target}. The DQN [14] algorithm is applied to the learning of each agent in the simulation. It learns an optimal control policy $\pi_i : S_i, G_i \rightarrow A_i, i \in \{Horizontal, Vertical\}$.

The horizontal position between the aircraft and the target is indicated by $C = (\varphi_\text{u}, D)$, where D is the azimuth of the aircraft and the distance between the two aircraft, respectively. Figure 2 depicts the position of the tracker relative to the target. The subscripts u and t indicate the tracker aircraft and the target, respectively, and φ_t indicates the azimuth of the tracker relative to the target.

The most important platform capability in air combat countermeasures training systems is flight capability, so this paper presents designs for a set of motion models to model the aircraft platform, which mainly reflect the flight trajectory under the limitation of aircraft flight performance. The six degrees of freedom for aircraft require consideration of the warplane as a rigid body, the complexity of the aircraft structure, and its longitudinal coupling. Here, a three-degrees-of-freedom model is used, ignoring the aircraft as a

rigid body, treating it as a mass, and assuming that the flight control system can respond accurately and quickly to form a maneuver trajectory. The core of the maneuvering decision problem is the rapid generation of the dominant maneuvering trajectory, and the aircraft three-degrees-of-freedom model can meet the solution requirements. The aircraft three-degrees-of-freedom model includes a mass point model of the aircraft platform and a dynamics model; the control model is shown in Figure 3. The specific models are

$$\begin{cases} \dot{V} = g(n_x - \sin(\theta)) \\ \dot{\theta} = \frac{g(n_z\cos(\gamma) - \cos(\theta))}{V} \\ \dot{\varphi} = \frac{g(n_z\sin(\gamma))}{V\cos(\theta)} \\ \dot{X} = V\cos(\theta)\cos(\varphi) \\ \dot{Y} = V\cos(\theta)\sin(\varphi) \\ \dot{Z} = -V\sin(\theta) \end{cases} \tag{8}$$

where x, y, and z denote the position of the aircraft in the geographic coordinate system; V is the flight speed; θ is the velocity inclination angle, i.e., the angle between the velocity direction and the horizontal plane, with upward as positive; φ is the heading angle, i.e., the angle between the velocity direction on the horizontal plane and the due north direction, with clockwise as positive; and where it is assumed that the velocity direction is always in line with the direction in which the nose is pointing, i.e., the angle of attack and the sideslip angle are zero.

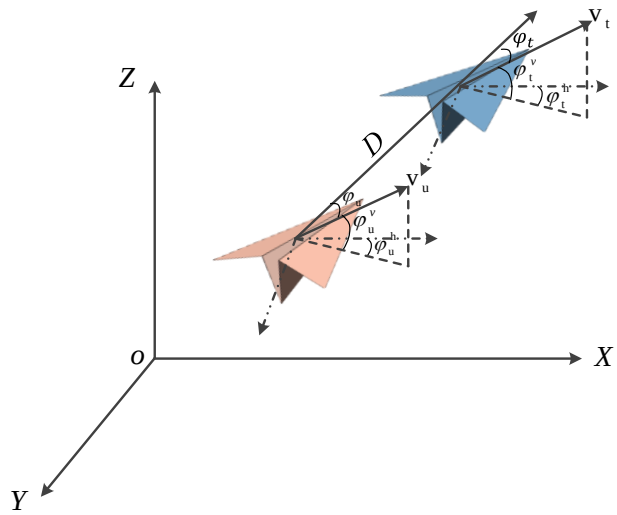

Figure 2. Vehicle situational model.

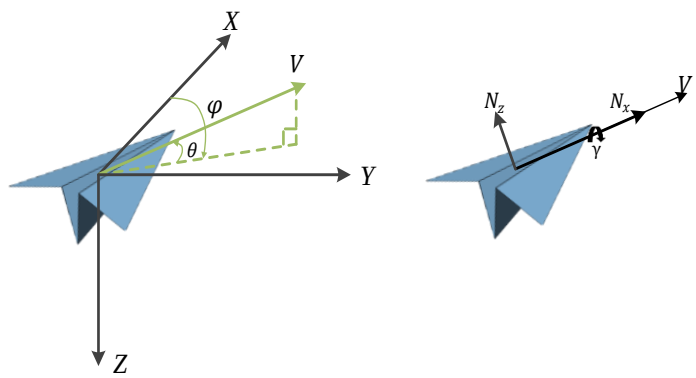

Figure 3. Vehicle control model.

θ, φ, γ denote the trajectory inclination angle, track deflection angle, and roll angle, respectively; n_x and n_z denote the tangential overload along the velocity direction and the normal overload in the vertical velocity direction, respectively; and g is the gravitational acceleration. In the above equation, the first three terms are the mass kinematics model and the last three terms are the aircraft dynamics model; the state variables include x, y, z, θ, φ, γ, ψ, and V; the control variables include n_x, n_z, and γ. Because an ideal mass point model is used in this study, the flight performance, control inputs, and control response parameters are restricted to make the trajectory and maneuvers of the target aircraft reasonable. Specifically, the values of n_x and n_z are limited to within 2 g and 5 g, respectively.

3. Hierarchical Reinforcement Learning Design

In this paper, we propose a hierarchical reinforcement learning training framework that comprises two parts: environment design and framework building. The purpose of the environment design is primarily to define the input and output state data available to the agent and the reward functions that can be obtained, and the framework building is primarily to establish the corresponding hierarchical network structure, realize the reward mapping corresponding to the course learning, and design the optimization algorithm and the corresponding training strategy.

3.1. Geometric Hierarchy in the Aircombat Framework

We formulate the intelligent body motion decision for a 3D air combat 1V1 confrontation as a Markov decision process (MDP), supplemented by a goal state G that we want the two agents to learn. We define this MDP as a tuple (S, G, A, T, μ), where S is the set of states, G is the goal, A is the set of actions, and T is the transition probability function. In this paper, a hierarchical reinforcement learning-based approach called a hierarchical reinforcement learning framework in geographic coordination for air combat, referred to as HRL-GCA, is used to build a shared multilevel structure. The method uses a technique called meta-learning, which learns from a set of tasks and applies this knowledge to new tasks. The algorithm can effectively build a shared multilevel structure, thus improving learning efficiency.

As shown in Figure 4, the global state S is a geometric representation of the tracker and target aircraft in a 3D simulated air combat scenario, including the positions S = (x, y, z) and velocities v = (vx, vy, vz) of both aircraft. At the beginning of each episode of each state s_i in the MDP, for a given initial state s_0 and target g_i, the solution to the sub-policy ω is a control policy π_i: S_i, $G_i \rightarrow A_i$ that maximizes the following value function:

$$v_{\pi_i}(s_i, g_i) \coloneqq E_{\pi_i}\left[\sum_{t=0}^{\infty} \mu \mathrm{i}_t R_t^i \middle| s_0^i = s_i, g_i = G_i\right] \tag{9}$$

An agent consists of an algorithm that updates a parameter vector (θ, ω) defining a stochastic policy $\pi_{\theta,\omega}(s|a)$, where theω parameter is shared among all sub-policies, whereas the θ parameter is learned for each senior policy starting from zero, encoding the state of the learning process on that task. In the considered setup, an MDP is first sampled from the P_S, the agent is represented by the shared parameter ω and the randomly initialized θ parameter, and the agent iteratively updates its θ parameter during the T-step interaction with the sampled MDP. The objective of the HRL-GCA is to optimize the value contributed by the sub-policy over the sampled tasks:

$$V = maximize_{\theta} E_{S \sim P_S, k=0,\ldots,T-1}[v_{\pi}(\mathrm{s}_k, \mathrm{g}_k)] \tag{10}$$

where π consists of a set of sub-policies π_1, π_2, …, π_N, and each sub-policy π_i is defined by a subvector ω_n. The network constructed by the parameter θ works as a selector. That is, the senior policy parameterized by θ selects the most appropriate behavior from index $\mathrm{n} \epsilon \{1, 2, \ldots, \mathrm{N}\}$ to maximize the global value function V.

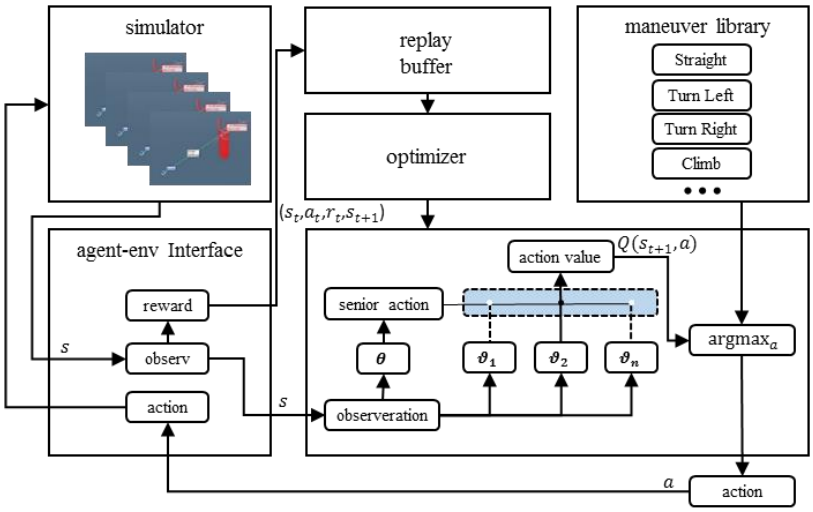

Figure 4. Model structure and training framework.

3.2. Reward Shaping

The senior action reward is used to train senior behaviors, which guide the sub-action to make further behavioral decisions. We take inspiration from the Meta-Learning Shared Hierarchies architecture to train the sub-policy independently, solidify its parameters, and then train senior action adaptively. Our approach is similar to Alpha-Dogfight [48], but we differ in that we implement further layering in the behavioral layer and map global rewards to local rewards by transformations under geographic coordination, and experimental results demonstrate that performance in the behavioral layer is further enhanced.

3.3. Senior Policy Reward

The senior policy performs discrete actions at a frequency five times lower than the sub-policy, which is 1 Hz and is trained using the same DQN as the sub-policy. The state space of the senior policy differs from that of the sub-policy, which is described in detail later in this paper. The reward for senior policy is given by

$$\mathrm{r}_{total} = \alpha \mathrm{r}_{angle} + \beta \mathrm{r}_{dis} \tag{11}$$

where α and β are positive parameters and $\alpha + \beta = 1$.

Firstly, the angle reward r_{angle} can help the model learn how to control the angle of the aircraft toward the target, and φ_{u} is related to the limits of the detection angle of the airborne radar and the off-axis angle of the missile. Specifically, the attack advantage increases the closer φ_{u} is to the desired angle, and r_{angle} reaches its maximum when $\varphi_{\mathrm{u}} = 0°$, i.e., when the velocity is aligned with the target:

$$\mathrm{r}_{angle} = e^{(-abs(\mathrm{f}(\varphi_{\mathrm{u}}) - \mathrm{f}(\varphi_{\mathrm{t}}))/180)} \tag{12}$$

Secondly, the distance redirection r_{dis} is designed based on the distance between the aircraft and the target, which can help the model learn how to control the position of the aircraft to achieve a reasonable position about the target. Specifically, the smaller the distance D between the aircraft and the target, the higher the r_{dis} value:

$$\mathrm{r}_{dis} = e^{(-abs(\mathrm{f}(D))/100)} \tag{13}$$

We used the above rewards for the initial training, and then in subsequent experiments, for comparison with other models, we adjusted the design of the reward to achieve the same state as the baseline. A description of how the three model rewards are adjusted in this paper can be found in Appendix A.

3.4. Sub-Policy Reward

However, the objective of this paper requires the mapping of rewards to the two subtask spaces, and we redistribute rewards for Agent 1 and Agent 2 via transformation in the geometric space. Because Agent 1 and Agent 2 are mainly implemented in two planes of control, as shown in Figure 3, r_{total} is achieved by mapping φ_u and D to the x-y and x-z planes using the function δ to reconstruct the G_i.

$$\begin{aligned} r^h_{total}\left(\delta(\varphi_u)^h, \delta(\varphi_t)^h, \delta\left(\vec{D}\right)^h\right) &= \alpha_h r^h_{angle}\left(\delta(\varphi_u)^h, \delta(\varphi_t)^h\right) + \beta_h r^h_{dis}\left(\delta(\varphi_u)^h, \delta\left(\vec{D}\right)^h\right) \\ r^v_{total}\left(\delta(\varphi_u)^v, \delta(\varphi_t)^v, \delta\left(\vec{D}\right)^v\right) &= \alpha_v r^v_{angle}\left(\delta(\varphi_u)^v, \delta(\varphi_t)^v\right) + \beta_v r^v_{dis}\left(\delta(\varphi_u)^v, \delta\left(\vec{D}\right)^v\right) \end{aligned} \tag{14}$$

Here,

$$r^h_{angle}\left(\delta(\varphi_u)^h, \delta(\varphi_t)^h\right) = e^{(-abs(\mathrm{f}(\varphi^h_u) - \mathrm{f}(\varphi^h_t))/180)}, \tag{15}$$

$$r^h_{dis}\left(\delta(\varphi_u)^h, \delta\left(\vec{D}\right)^h\right) = e^{(-abs(\mathrm{D}^h * f(\varphi^h_u)/100)}, \tag{16}$$

$$r^v_{angle}\left(\delta(\varphi_u)^v, \delta(\varphi_t)^v\right) = e^{(-abs(\mathrm{f}(\varphi^v_u) - \mathrm{f}(\varphi^v_t))/180)}, \tag{17}$$

$$r^v_{dis}\left(\delta(\varphi_u)^v, \delta\left(\vec{D}\right)^v\right) = e^{(-abs(\mathrm{D}^v * \mathrm{f}(\varphi^v_u))/100)}. \tag{18}$$

The redistribution of rewards is achieved by the function δ. The δ function is a spatial projection operator that maps reward elements φ_u, φ_t, and $\vec{D}$ to the x-y and x-z planes, respectively. This ensures that the reward functions r^h_{total} and r^v_{total}, which are used for training in the x-y and x-z planes, have the same expression. However, their auto-covariates are the result of the projection through the δ posterior: φ^h_u, φ^h_t, D^h, and φ^v_u, φ^v_t, D^v, respectively, as detailed in Appendix B. Of these, reward r^h_{total} allows the tracker to follow the target better on the x-y surface, and reward r^v_{total} is used to suppress the altitude difference and, as much as possible, encourage the aircraft to be at the same altitude level as the target at high altitude. In addition, in this paper, the treatment in Equation (7) is also applied to its rewards in the comparisons with other baselines.

3.5. Hierarchical Training Algorithm

In this paper, a course learning approach is used for hierarchical training; the definition of the algorithm is detailed in Appendix C, and the policy network is trained to interact with the environment at a frequency of 10 Hz. The same observation space is used for both policies.

We then explore cooperative learning between Agent 1 and Agent 2 in the training of horizontal control and height control policies. In each iteration of the learning, firstly, Agent 1 moves the tracker on the x-y surface of the 3D geographic coordination scenes; secondly, the next state s^1_{t+1} and the intermediate state s^2_t update after action a^1_t; and thirdly, Agent 2 moves the tracker on the x-z surface. The next state s^2_{t+1} updates after a^2_t.

Initial conditions: These initial conditions are divided into tracking targets that start moving from different positions and take different forms of motion in the height and horizontal planes. Concerning stochastic multistep payoffs, for time–distance learning, multistep payoffs tend to lead to faster learning when appropriately tuned for the number of steps to be used in the future. Instead of tuning a fixed value, we define the maximum number of steps in the future and uniformly sample the maximum value. A common expression for future value is

$$\mathrm{Q}(S', \mathrm{A}) \leftarrow \mathrm{Q}(\mathrm{S}, \mathrm{A}) + \partial\left(\mathrm{R} + \tau \max_{a'} Q(S', a') - \mathrm{Q}(\mathrm{S}, \mathrm{A})\right) \tag{19}$$

The tactical objective of the horizontal plane tracking subtask is to enable the tracker to continuously track the target aircraft in the x-y plane. Formally, motion in the x-y plane is achieved by outputting horizontal motion, successive horizontal left turns, and successive horizontal right turns at each simulation step with a constant steering speed of 18°/s. The initial and termination conditions for the x-y subtasks are designed as shown in Figure 2. The tactical objective of the altitude tracking subtask is to enable the tracker to follow the target aircraft consistently at altitude. The mission can start in any state. Formally, motion in the x-z plane is achieved by outputting horizontal motion, continuous climb, and continuous descent in each simulation step, with a constant climb and descent rate of 20 m/s.

This in turn contains one output, namely, the value of $Q(s, x_i)$. The activation function is the logsoftmax function:

$$Q(s, x_i) = (x_i - x_m) - \log(\sum_{j=0}^{n} e^{x_j - x_m}) \tag{20}$$

and Equation (20) directly outputs the value of each action using the logsoftmax nonlinear function, where x_m is the largest element of $X = (x_1, x_2, \ldots x_n)$.

3.6. Hierarchical Runtime Algorithm

In the hierarchical runtime algorithm, we explore the cooperation of Agent 1 and Agent 2 in a 3D simulated air combat situation. The algorithm is defined in detail in Appendix D. In each iteration of learning, firstly, Agent 1 moves in the x-y plane of the 3D air combat scenario; secondly, the next state s_{t_1+1} and intermediate state s_{t_2} are updated after action; and thirdly, Agent 2 moves up or down in the x-z plane. The next state s_{t_2+1} is updated after a_{t_2}.

For each action mi, a minimum period t = 1500 milliseconds and a maximum period ui = 4 milliseconds are set. When the reinforcement learning intelligence outputs the action m_i (including the stop action) at moment T, it starts to execute m_i if no action is executed at the previous moment T − 1. If moment T – 1 performs action m_j and the execution time is greater than or equal to t, then at moment T, the agent will be allowed to execute m_i to replace the action m_j, otherwise not. If moment T – 1 performs action m_j and the execution time is less than t_j, then the output behavior m_i at moment T is ignored. When the reinforcement learning intelligence outputs no behavior (which is not the same as the stopping behavior) at moment T, if the previous moment T – 1 performed the behavior m_k and the execution time is greater than or equal to ui, then the execution of the no-behavior starts; otherwise, the execution of the behavior m_k continues. The setting of the min-max period can to some extent prevent incorrect behavior of the flight unit.

4. Results

4.1. Experimental Environment Setup

The experiments in this paper use a hierarchical reinforcement learning framework to solve the problem in an air combat simulation environment. The hardware environment used in the experiments is an Intel Core i7-8700K CPU, 16 GB RAM, and an NVIDIA GeForce GTX 4090 Ti graphics card. The size of the 3D space in the experiment is 100 km × 100 km × 10 km; there are 20,000 × 480 s training episodes for each model; and the actual data sampling frequency is 10 HZ. The experimental results show that the performance of the algorithm improves significantly after the training of 20,000 episodes.

4.2. Performance Metrics during Training and Validation

To select the best-performing agent, we create an evaluation metric to compare the training results of various methods. The qualitative and quantitative results demonstrate the usefulness of our proposed model. The tracking performance of the tracker is evaluated when the target is moving at 0–180° relative north in an air combat environment. For com-

parison, we trained 2400 episodes for each angle type, for a total of 11.5×10^6 simulation steps, and tested 500 samples for the corresponding angle types.

The meanings of each indicator are as follows: miss distance represents the average distance between the tracker and the target during the entire tracking process; miss angle represents the average track angle φ_u between the tracker and the target during the entire tracking process; approach time represents the time taken to approach the target for the first time to a certain distance; hold distance time is the length of time that the tracker stays within a certain distance of the target; hold angle time is the time that the tracker stays within a certain angle of the target; and cost time refers to the time spent by each strategy model when outputting the current action command.

$$\mathrm{P(Miss\ Distance)} = \frac{acc(total\ dis)}{t_{epoch\ time}} \tag{21}$$

$$\mathrm{P(Miss\ Angle)} = \frac{acc(track\ angle\ \varphi_u)}{t_{epoch\ time}} \tag{22}$$

$$\mathrm{P(Approach\ time)} = \tau_{TOA} - \tau_0 \tag{23}$$

$$\mathrm{P(Hold\ distance\ time)} = \frac{\tau(\mathrm{dis} \leq \sigma)}{\tau(epoch\ time)} \tag{24}$$

$$\mathrm{P(Hold\ angle\ time)} = \frac{\tau(\mathrm{angle} \leq \partial)}{\tau(epoch\ time)} \tag{25}$$

$$\mathrm{P(Cost\ Time)} = \tau(\theta) \tag{26}$$

4.3. Validation and Evolution of the Hierarchical Agents

In this experiment, we reproduce the models and algorithms in three papers [9,15,49], and apply the hierarchical reinforcement learning framework established in this paper to learn and train them, respectively, while mapping the reward functions shaped in the three papers in the corresponding sub-state spaces; then, in the air combat environment established in this paper, different models are compared in the same test scenarios, and the performance of the three original models is compared with that of the models after applying HRL. We use the benchmark performance comparison method proposed in Section 4.2 to compare the models proposed in the paper, as shown in Table 1. Models 1, 2, and 3 denote the performance of the three models. The experimental results show that the HRL-GCA proposed in this paper can achieve higher scores in all three dimensions under the six test metrics compared with the other three models: the miss distance, miss angle, and approach time decreased by an average of 5492 m, 6.93 degrees, and 34.637 s, respectively, and the average improvement of angle maintenance and distance maintenance time is 8.13% and 16.52%, respectively. Of the other models, Model 2 has the highest hold distance and hold angle time with percentages of 41.12 and 15.44, respectively. In addition, the HRL-GCA model can converge faster and achieve higher accuracy in the training process. Therefore, we conclude that HRL-GCA demonstrates better performance in this experiment.

As shown in Table 1, the implementation of HRL models results in a 40–50% increase in runtime compared to the baseline models. This can be attributed to the fact that HRL models involve more complex computations and require more processing time. This is mostly because HRL incorporates several learning layers. Consequently, the HRL will execute over two extra neural networks in addition to the base models.

Notwithstanding, we consider the time cost to be acceptable based on the comparative results presented in Table 1. For instance, Model 2 benefited from HRL improvement, requiring only a minimum of 87.56 s for Approach Time and making approximately 65 decisions for the approach to the target. In contrast, the corresponding model without HRL improve-

ments required 137.06 s for Approach Time, making about 145 decisions for the approach to the goal. The HRL-improved model achieves its goal with only 65 decisions compared to the original model's 145, resulting in a 55% improvement in decision-making efficiency. This increase in efficiency of the HRL-improved model (55%) offsets the additional time overhead required to execute the model (43.40%).

Furthermore, as an example, Model 2 shows improved Hold Distance Time and Holding Angle Time by 16.33% and 8.24%, respectively, after implementing HRL. Furthermore, compared to the model without HRL improvement, the distance and angle tracking stability are enhanced by 65% and 114%, respectively. In summary, although the computation time spent increased by 43.40%, the HRL improvement resulted in a 55% increase in decision efficiency within the same timeframe. The distance and angle tracking stability also increased by 65% and 114%, respectively. Therefore, this improvement is deemed reasonable.

Table 1. Comparison of experimental results with and without the HRL framework.

Reward Type	Miss Distance (m)			Miss Angle (°)			Approach Time (s)		
	without hrl	with hrl	Ratio of Decrease	without hrl	with hrl	Ratio of Decrease	without hrl	with hrl	Ratio of Decrease
Model 1	44,378.83 (±4020.2)	38,900.41 (±3778.3)	12.34%	32.56 (±5.73)	25.71 (±1.46)	21.04%	141.99 (±19.89)	101.459 (±20.09)	28.54%
Model 2	41,696.70 (±3692.7)	35,797.28 (±3494.9)	14.14%	31.68 (±2.66)	28.726 (±8.39)	9.32%	137.06 (±21.32)	87.56 (±17.83)	36.11%
Model 3	43,526.82 (±2332.4)	38,427.20 (±2371.3)	11.71%	36.69 (±6.93)	25.69 (±6.41)	29.98%	102.89 (±25.18)	89.01 (±18.66)	13.49%
Reward Type	**Hold Distance Time (%)**			**Hold Angle Time (%)**			**Cost Time (ms)**		
	without hrl	**with hrl**	**Ratio of Increase**	**without hrl**	**with hrl**	**Ratio of Increase**	**without hrl**	**with hrl**	**Ratio of Increase**
Model 1	17.15% (±6.39%)	33.14% (±3.87%)	15.99%	5.23% (±0.22)	11.38% (±0.99%)	6.15%	0.97	1.427	47.11%
Model 2	24.79% (±4.99%)	41.12% (±4.67%)	16.33%	7.2% (±0.63%)	15.44% (±1.67%)	8.24%	0.94	1.348	43.40%
Model 3	19.30% (±4.61%)	36.56% (4.51%)	17.26%	4.06% (±1.07%)	14.07% (±1.72%)	10.01%	0.91	1.410	54.95%

5. Discussion

5.1. Trajectory of Air Combat Process

As shown in Figures 5–8, we deploy the algorithm of this paper in a typical air combat scenario and compare its tracking of the target aircraft with a model that does not use this algorithm. During air combat, continuous tracking of the target aircraft in a given scenario is necessary to shoot it down. In the test cases, the target aircraft maneuvers continuously at altitude and moves away from the tracker by turning away from it, as seen in the 3D and 2D tracking trajectories, but the tracker ensures continuous alignment with the target in both altitude and direction. In contrast, the use of the other model fails to achieve continuous tracking of the target in either direction or altitude. Furthermore, the red dashed line in Figures 6 and 8 shows the desired tracking trajectory for the target.

In our experiments, we use a hierarchical reinforcement learning framework to optimize and enhance the vehicle tracking trajectories. The trajectories in Figure 9 show the tracking states of the modified model 2 based on HRL and the model set out in this paper in the XY plane, XZ plane, and XYZ 3D space, respectively. Of these, in Figure 10, the red line is the tracking flight, the blue line is the tracked flight, and the number represents the flight trajectory sequence of both flights. The experimental results show that the use of the hierarchical reinforcement learning framework can effectively improve the accuracy and stability of aircraft tracking trajectories and can effectively reduce their deviation. It is found that Model 3 is more sensitive to the weighting parameters α, β in Equation (11) and has the best test results when the two reward ratios in the original paper are set to 0.5, 0.5. Irrespective of the rewards in Models 1, 2, and 3 or the rewards used in this paper, in Figures 11 and 12, the tracking performance of a single network simultaneously controlling

the motion of the horizontal plane and the motion of the height layer is inferior to that of multiple networks controlling them separately. In addition, the experimental results show that the use of the reinforcement learning method can effectively improve the accuracy of aircraft tracking trajectories, thus improving the timeliness of target tracking.

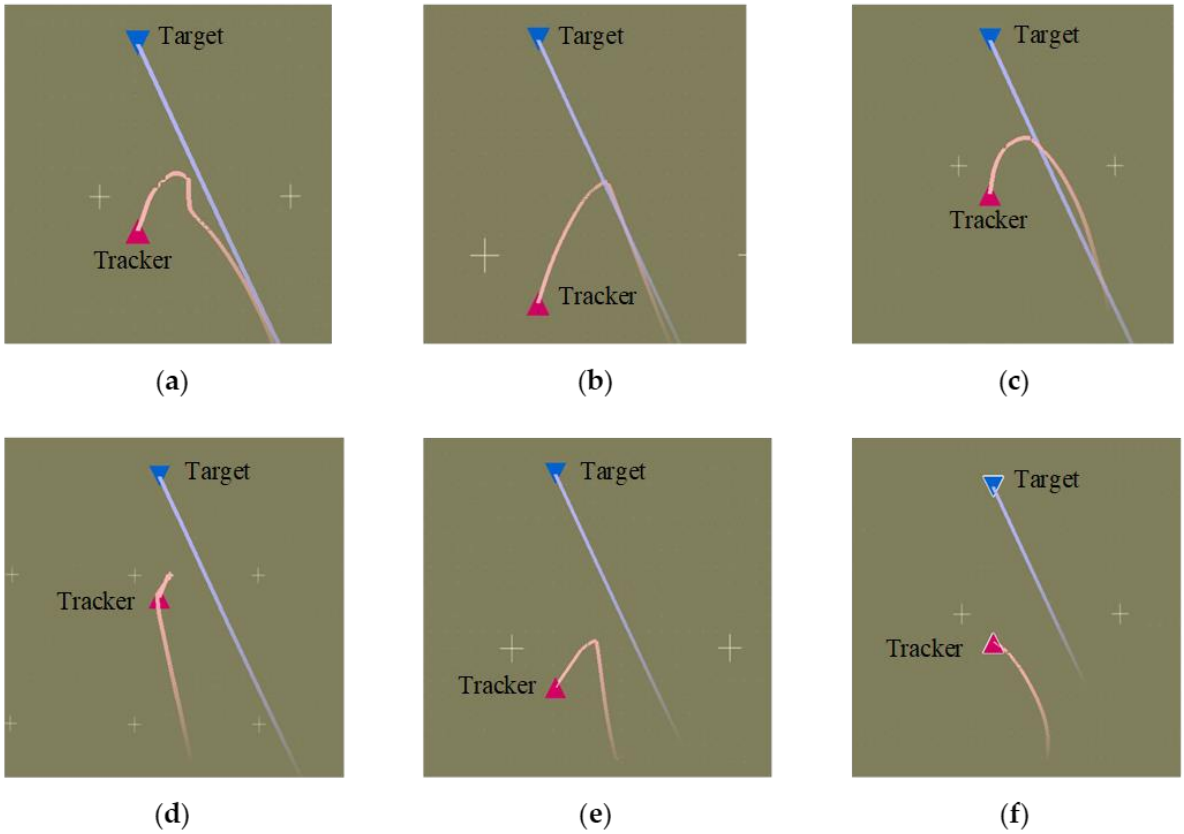

Figure 5. Angle tracking performance: comparison of models ((**a**–**f**) represents the horizontal tracking trajectory of model 1 with HRL framework, the horizontal tracking trajectory of model 2 with HRL framework, the horizontal tracking trajectory of model 3 with HRL framework, the horizontal tracking trajectory of model 1 without HRL framework, the horizontal tracking trajectory of model 2 without HRL framework, and the horizontal tracking trajectory of model 3 without HRL framework, respectively).

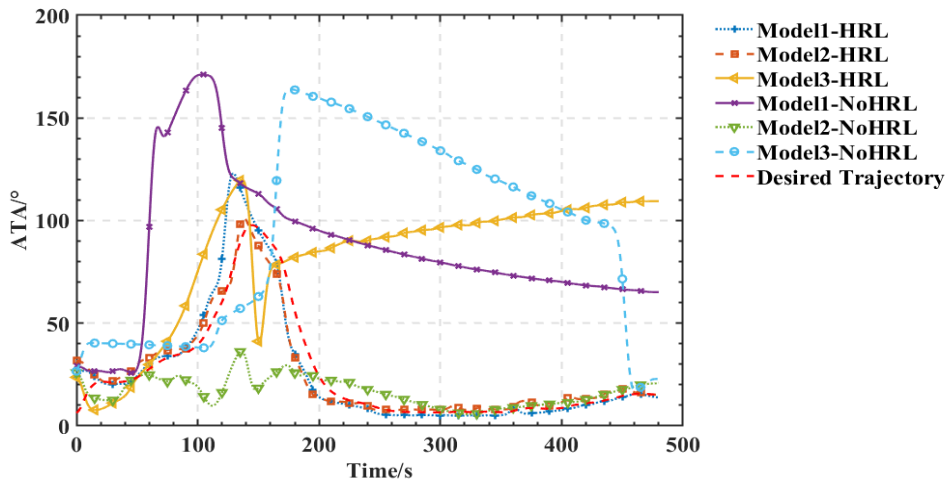

Figure 6. Comparison of angle tracking states of different models in the same scene.

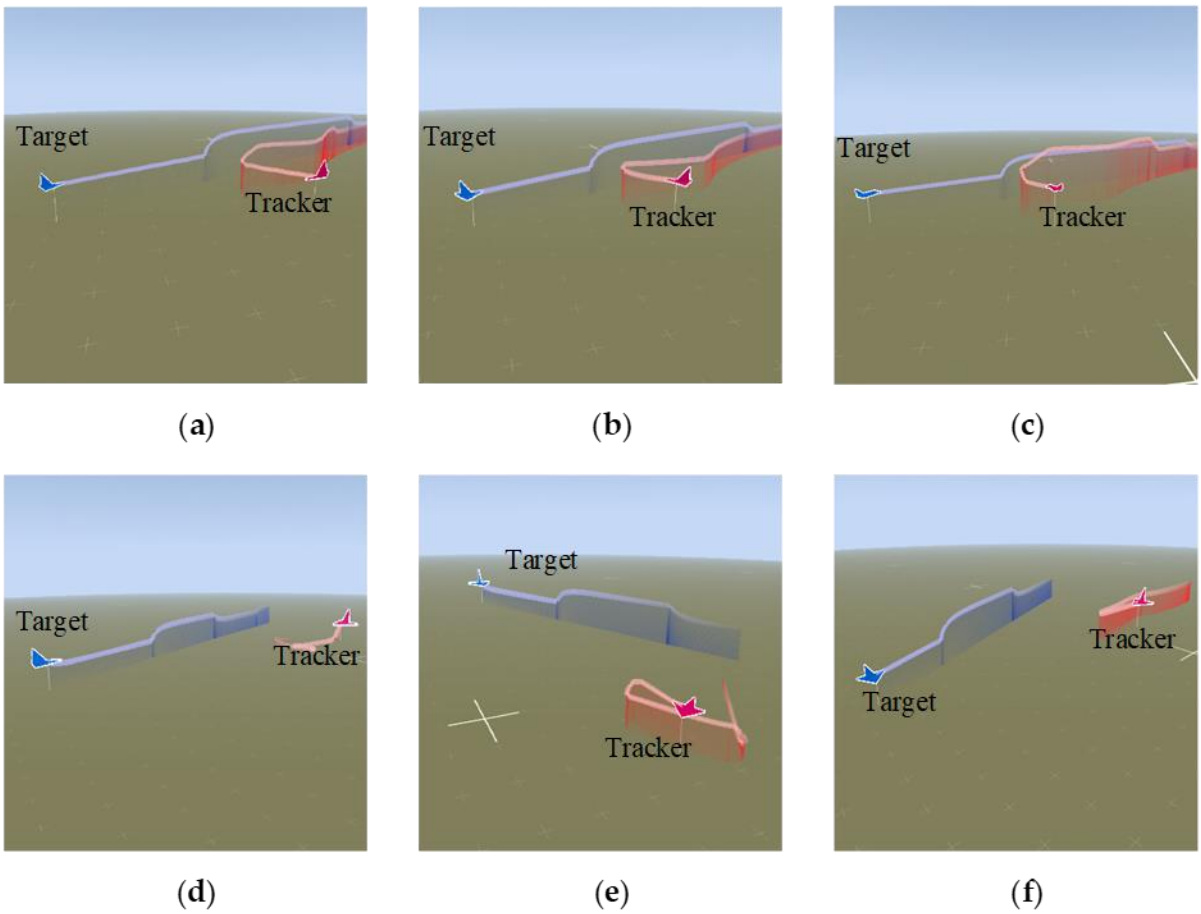

Figure 7. Comparison of the height tracking performance of the models ((**a**–**f**) represents the vertical tracking trajectory of model 1 with HRL frame, the vertical tracking trajectory of model 2 with HRL frame, the vertical tracking trajectory of model 3 with HRL frame, the vertical tracking trajectory of model 1 without HRL frame, the vertical tracking trajectory of model 2 without HRL frame, and the vertical tracking trajectory of model 3 without HRL frame, respectively).

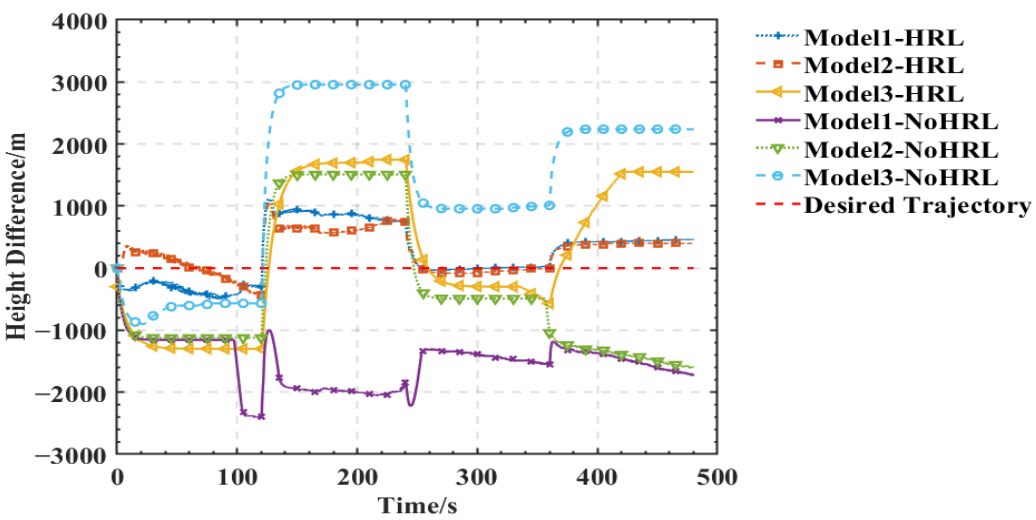

Figure 8. Comparison of different models in the same scene with height-tracked states.

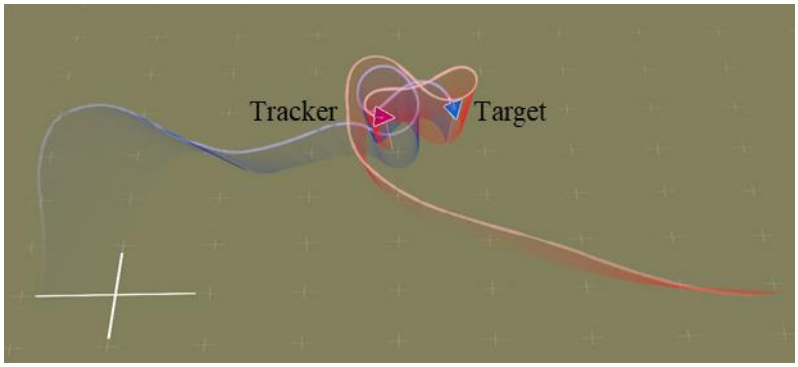

Figure 9. Tracking the trajectory of the HRL modified model 2 agent.

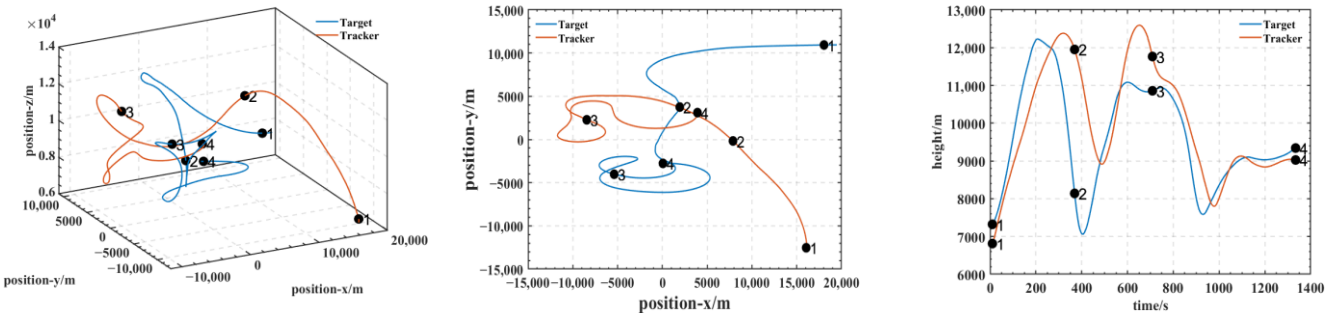

Figure 10. Three-dimensional and 2D trajectories when the target and tracker are using the HRL agent.

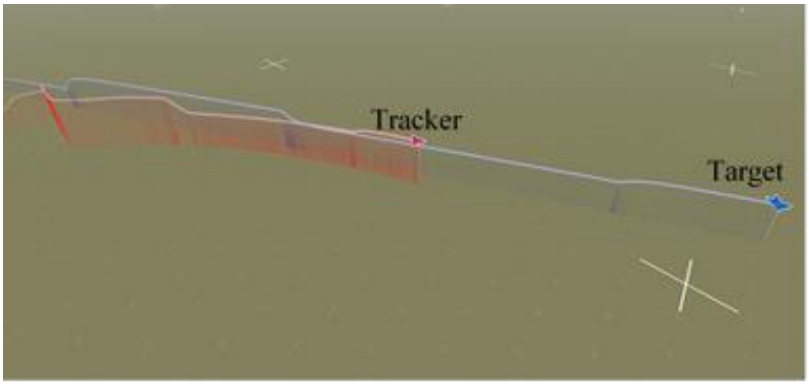

Figure 11. Tracking the trajectory of the HRL-free model 2 agent.

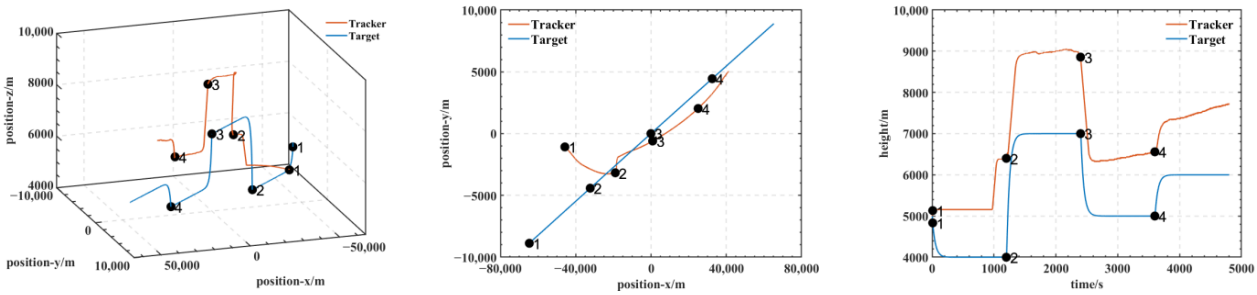

Figure 12. Three-dimensional and 2D trajectories when the target and tracker are not using the HRL agent.

5.2. Training Process

The analysis of the experimental results in this paper shows that we can compare the changes in reward and loss of Models 1, 2, and 3 with the HRL-GCA model during the training process. From the experimental results, the reward and loss of HRL-GCA converge as the episodes increase and reach their optimal state after stabilization. In Figure 13, from the change in reward, our research model reward reaches its maximum at episode 21, whereas the rewards of the three standard models still show large fluctuations at episode 21, indicating that the reward of our research model has better convergence performance. Figure 14 illustrates the loss parameters during training after normalization. From the change in loss, the loss of our research model reaches its minimum at episode 592, whereas the losses of the three standard models still show a large fluctuation at episode 592, indicating that our research model has a better convergence performance of loss.

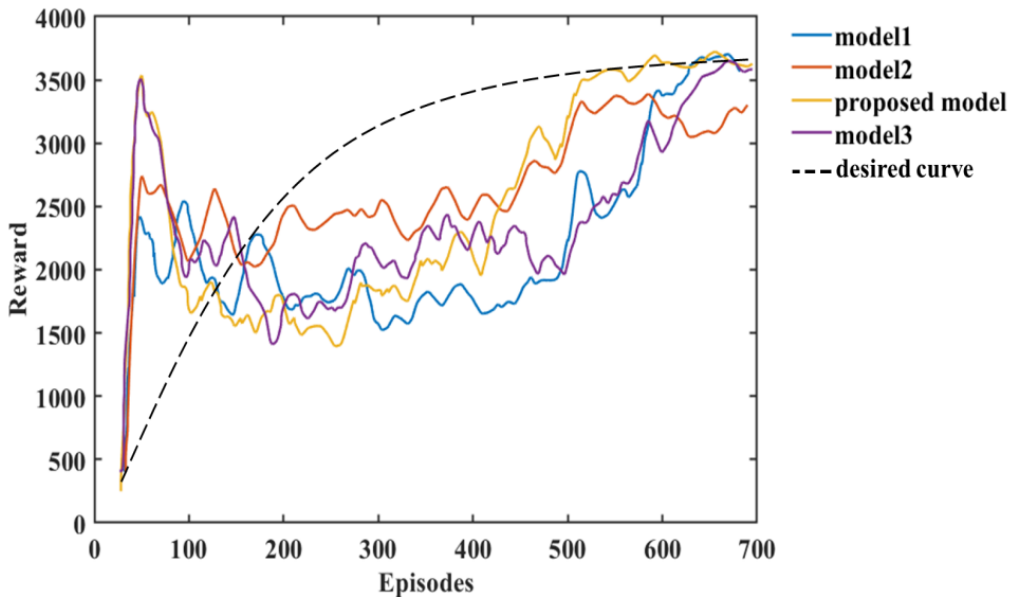

Figure 13. The relationship between the reward and episodes of the models.

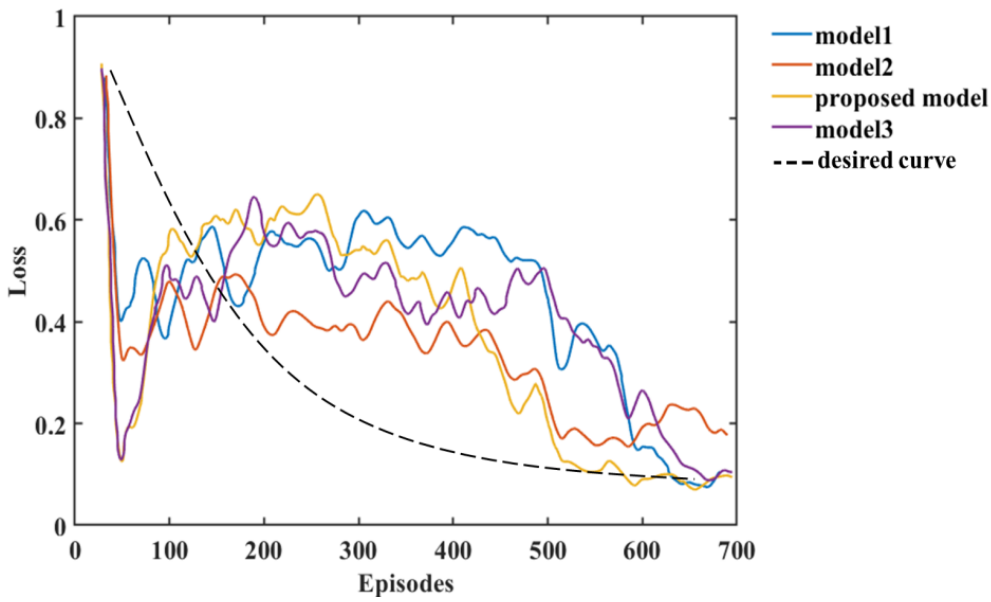

Figure 14. The relationship between the loss and episodes of the models.

In summary, Figures 13 and 14 show that our research model has the best convergence performance during the training process, as well as an optimal state after stabilization. Therefore, when introducing a new sub-policy, the framework in this paper can achieve fast adaptation in training and learning for the corresponding task. Furthermore, in Figure 13, Model 3 and the proposed model decrease significantly after the first peak around episode 21; such behavior makes these models inferior to Model 2. This is mainly due to overfitting. From Equations (9) and (10) in Section 1, it can be deduced that each sub-policy π_i can quickly learn its corresponding subvector ω_n after the initial learning phase, but ω_n learns only a tiny portion of the state space, and it needs to further learn the θ corresponding to the selector to maximize the global value function V.

In addition, the θ corresponding to the selector can be learned from the ω_n subvector, but due to the large amount of training data required to train θ, as $\pi_{\theta,\omega}(s|a)$ performs stochastic exploration, the variance is large, resulting in a decrease in $v_\pi(\mathrm{s}_k, \mathrm{g}_k)$ until $S \sim P_S$ accumulates sufficiently to train a valid θ parameter. Furthermore, model 2 can maintain a more stable exploration-utilization capability throughout the training process, but the model proposed in this paper has a higher final reward value compared to the other models due to a better memory of what has been learned. Finally, we use black dashed lines in Figures 13 and 14 to show the variation of reward and loss with the training process in the ideal case.

6. Conclusions

The hierarchical reinforcement learning framework in geographic coordination for air combat proposed in this paper trains two types of neural networks using distance reward, angle reward, and a combination of both to control the vehicle in multiple dimensions. The model has achieved good results in tracking targets in multiple dimensions. In thousands of tests, it achieved an average improvement of 8.13% in angle tracking and 16.52% in distance tracking over the baseline model, demonstrating its effectiveness. However, the model has limitations, especially in complex environments or when the goal is to perform complex maneuvers, and it is not yet able to achieve optimal control. Future research should focus on improving the tracking performance in such scenarios, along with exploring additional reward functions to improve stability and accuracy. Furthermore, numerous challenges remain, such as addressing two-agent game combat and extending to 2v2 and multi-agent combat scenarios in air combat pair control, which warrant further exploration. In addition, the application potential of this model in other real-world scenarios should be investigated.

Author Contributions: Conceptualization, R.C. and Q.Z.; methodology, R.C. and H.L.; software, R.C.; validation, R.C. and H.P.; formal analysis, R.C.; investigation, G.Y.; resources, H.L.; data curation, H.P.; writing—original draft preparation, R.C.; writing—review and editing, Q.Z.; visualization, R.C.; supervision, G.Y.; project administration, H.L.; funding acquisition, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by a Project funded by China Postdoctoral Science Foundation (Grant No. 61932012), and in part by the Natural Science Basic Research Program of Shaanxi (Program No. 2022JQ-061).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

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

Appendix A. Design of Reward Functions

We adjusted the reward functions in the three models appropriately according to the characteristics and goals of the tasks. Reward functions play a crucial role in reinforcement learning in that they guide model learning and decision-making. However, in complex

real-world tasks, it is very challenging to design ideal reward functions that apply to all situations. To realize the model in this paper to adapt the reward functions of the three models [9,15,49], we mainly adjust the $f(\varphi_u)$, $f(\varphi_t)$, and $f(D)$ in Equations (12) and (13).

In model 1:

$$f(\varphi_u) = \begin{cases} 1 - \frac{|\varphi_u|}{150} 0 \leq |\varphi_u| \leq 30 \\ 030 \leq |\varphi_u| \leq 180 \end{cases}$$

$$f(\varphi_t) = -\frac{(180 - |\varphi_t|)\pi}{180}, -180 \leq \varphi_t \leq 180$$

$$f(D) = \begin{cases} 1D < 10 \\ -\frac{D-10}{30} 10 \leq D \leq 40 \\ 0D > 40 \end{cases}$$

In model 2:

$$f(\varphi_u) = 0$$

$$f(\varphi_t) = \frac{360 - |\varphi_u| - |\varphi_t|}{360}, 0 \leq |\varphi_u| \leq 180, 0 \leq |\varphi_t| \leq 180$$

$$f(D) = \begin{cases} 0D \geq 80 \\ -\frac{D-60}{40} 60 < D \leq 80 \\ -\frac{D-40}{20} 40 < D \leq 60 \\ 120 < D \leq 40 \\ 0D \leq 20 \end{cases}$$

In model 3:

$$f(\varphi_u) = -(\varphi_u - 30)^2, 0 \leq |\varphi_u| \leq 180$$

$$f(\varphi_t) = -(\varphi_t - 60)^2, 0 \leq |\varphi_t| \leq 180$$

$$f(D) = -(R/80)^2$$

Appendix B. The Spatial Projection

The main role of the spatial mapping operator δ is to map the base auto-morphisms of the reward function to the x-y and x-z planes, such that the x-y and x-z planes share the same reward function form but have different auto-morphisms.

Where $\overrightarrow{V_u}$ and $\overrightarrow{V_t}$ represent the velocity vectors of the tracker and target in the geographic coordinate system, respectively, with scalar forms $(v_{u,x}, v_{u,y}, v_{u,z})$ and $(v_{t,x}, v_{t,y}, v_{t,z})$. $\vec{D}$ represents the vector expression of the line that connects the center of gravity of the tracker and the center of gravity of the target in the geographic coordinate system, and its scalar form is (d_x, d_y, d_z). As shown in Figure 2, φ_u is the angle between vector $\overrightarrow{V_u}$ and $\vec{D}$, while φ_t is the angle between vector $\overrightarrow{V_t}$ and $\vec{D}$.

In x-y plane:

$$\varphi_u^h = \delta(\varphi_u)^h_{x,y,z\to x,y} = \delta\left(\arccos\frac{\overrightarrow{V_u} * \vec{D}}{\left|\overrightarrow{V_u}\right| * \left|\vec{D}\right|}\right) = \arccos(v_{u,x}d_y + v_{u,y}d_x)$$

$$\varphi_t^h = \delta(\varphi_t)^h_{x,y,z\to x,y} = \delta\left(\arccos\frac{\overrightarrow{V_t} * \vec{D}}{\left|\overrightarrow{V_t}\right| * \left|\vec{D}\right|}\right) = \arccos(v_{t,x}d_y + v_{t,y}d_x)$$

$$D^h = \delta\left(\vec{D}\right)^h_{x,y,z\to x,y} = \sqrt{d_x * d_x + d_y * d_y}$$

In x-z plane:

$$\varphi_u^v = \delta(\varphi_u)_{x,y,z\to x,y}^v = \delta\left(\arccos\frac{\overrightarrow{V_u}*\overrightarrow{D}}{\left|\overrightarrow{V_u}\right|*\left|\overrightarrow{D}\right|}\right) = \arccos(v_{u,x}d_z + v_{u,z}d_x)$$

$$\varphi_t^v = \delta(\varphi_t)_{x,y,z\to x,y}^v = \delta\left(\arccos\frac{\overrightarrow{V_t}*\overrightarrow{D}}{\left|\overrightarrow{V_t}\right|*\left|\overrightarrow{D}\right|}\right) = \arccos(v_{t,x}d_z + v_{t,z}d_x)$$

$$D^v = \delta\left(\overrightarrow{D}\right)_{x,y,z\to x,y}^v = \sqrt{d_x * d_x + d_z * d_z}$$

Appendix C. Algorithm A1

The training algorithm for hierarchical reinforcement learning proposed in this paper involves the steps of hierarchical policy optimization, subtask policy optimization, hierarchical reward design, and network parameter training. Among them, the reward function of each layer can be adjusted and optimized according to the objectives and characteristics of the task. By reasonably designing the reward function, the hierarchical reinforcement learning network can be guided to learn decision-making and behavioral strategies suitable for the task. Through iteration and optimization of these steps, we can obtain a hierarchical reinforcement learning model adapted to the complex task.

Algorithm A1 The hierarchical training algorithm

Initialize a one-on-one air combat simulation environment
Initialize replay buffer R1,R2 to capacity N
Initialize the action-value function Q with random weights
Initialize Agent DQN1with (Q,R1), DQN2 with (Q,R2)
for episode = 1, MAX do
  Initialize state s_t^1 = env.reset()
  for t = 1, T do
    Agent 1 samples action $a_t^1 = a_{t,\theta 1}(s_t^1) + \varepsilon_t$using DQN1
    Execute a_t^1 in an air combat simulation environment
    The aircraft observes s_{t+1}^1, reward r_t^1
    Agent 2 samples action $a_t^2 = a_{t,\theta 2}(s_t^2) + \varepsilon_t$using DQN2
    Execute a_t^2 in an air combat simulation environment
    The aircraft observes s_{t+1}^2, reward r_t^2
  end for
Store $(s_t^1, a_t^1, r_t^1, s_{t+1}^1)$ into R1, Store $(s_t^2, a_t^2, r_t^2, s_{t+1}^2)$ into R2
if the Update condition is reached, then
  Sample random mini-batch of m from the replay buffer
  Calculate the target Q value for each both DQN in each transition $\hat{Q}_m = r_m + \lambda max_{a_{\theta'}} Q_m(s_m^{t+1}, a_{\theta'}(s_m^{t+1}); \theta)$
  Compute gradient estimation $\Delta\theta_1$ and $\Delta\theta_2$
  Update the parameters of DQN1 and DQN2 based on the optimizer using $\Delta\theta_1$ and $\Delta\theta_2$
  end for
end for

Appendix D. Algorithm A2

In practice, the hierarchical reinforcement learning algorithm proposed in this paper gradually improves the performance and decision-making ability of the whole system through the steps of hierarchical structure, policy execution, reward signaling, and parameter updating. Through decision-making and learning at different levels, the hierarchical reinforcement learning algorithm can be more flexible in solving complex, high-dimensional tasks and achieve improved adaptability and generalization ability.

Algorithm A2 The hierarchical running algorithm

Load trained neural networks $Q_1(\mathrm{s}, \mathrm{a}, \theta_1), Q_2(\mathrm{s}, \mathrm{a}, \theta_2)$
Initialize the state of the tracker and target $(s^u_{T=0}, s^t_{T=0})$
Initialize target maneuver strategy $\pi(\mathrm{a}|\mathrm{s})_{\mathrm{t}}$
for step = 1 to max step do
  for i in (θ_1, θ_2) do
    Calculate $(\varphi^h_u, \varphi^h_t, \mathrm{D})$
    Execute $a_t \sim \pi_{\theta_1}(a_t|s^u_T)$
    $\left|\Delta X^h_T, \Delta Y^h_T, \Delta Z^h_T\right|$is obtained according to (2)
    Get the next state s^u_{T+1}
    Calculate $(\varphi^v_u, \varphi^v_t, \mathrm{D})$
    Execute $a_t \sim \pi_{\theta_2}(a_t|s^u_{T+1})$
    $|\Delta X^v_T, \Delta Y^v_T, \Delta Z^v_T|$is obtained according to (2)
    Get the next state s^u_{T+2}
  $s^u_T = s^u_{T+2}$
  end for
The target moves to the next state s^t_{T+1} according to the strategy $\pi(\mathrm{a}|\mathrm{s})_{\mathrm{t}}$
  $s^t_T = s^t_{T+1}$
end for

References

1. Sutton, R.S.; Barto, A. *Reinforcement Learning: An Introduction, Nachdruck*; Adaptive Computation and Machine Learning; The MIT Press: Cambridge, MA, USA, 2014; ISBN 978-0-262-19398-6.
2. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]
3. Hu, D.; Yang, R.; Zuo, J.; Zhang, Z.; Wu, J.; Wang, Y. Application of Deep Reinforcement Learning in Maneuver Planning of Beyond-Visual-Range Air Combat. *IEEE Access* **2021**, *9*, 32282–32297. [CrossRef]
4. Jiang, Y.; Yu, J.; Li, Q. A novel decision-making algorithm for beyond visual range air combat based on deep reinforcement learning. In Proceedings of the 2022 37th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Beijing, China, 19–20 November 2022; IEEE: New York, NY, USA, 2022; pp. 516–521. [CrossRef]
5. Shi, W.; Song, S.; Wu, C.; Chen, C.L.P. Multi Pseudo Q-learning Based Deterministic Policy Gradient for Tracking Control of Autonomous Underwater Vehicles. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3534–3546. Available online: http://arxiv.org/abs/1909.03204 (accessed on 3 April 2023). [CrossRef] [PubMed]
6. Byrnes, C.M.W. Nightfall: Machine Autonomy in Air-to-Air Combat. *Air Space Power J.* **2014**, *28*, 48–75.
7. Kim, C.-S.; Ji, C.-H.; Kim, B.S. Development of a control law to improve the handling qualities for short-range air-to-air combat maneuvers. *Adv. Mech. Eng.* **2020**, *12*, 168781402093679. [CrossRef]
8. Xu, J.; Zhang, J.; Yang, L.; Liu, C. Autonomous decision-making for dogfights based on a tactical pursuit point approach. *Aerosp. Sci. Technol.* **2022**, *129*, 107857. [CrossRef]
9. Li, W.; Shi, J.; Wu, Y.; Wang, Y.; Lyu, Y. A Multi-UCAV cooperative occupation method based on weapon engagement zones for beyond-visual-range air combat. *Def. Technol.* **2022**, *18*, 1006–1022. [CrossRef]
10. Kong, W.; Zhou, D.; Du, Y.; Zhou, Y.; Zhao, Y. Hierarchical multi-agent reinforcement learning for multi-aircraft close-range air combat. *IET Control Theory Appl* **2022**, *17*, cth2.12413. [CrossRef]
11. Ernest, N.; Carroll, D. Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions. *J. Def. Manag.* **2016**, *06*, 2167-0374. [CrossRef]
12. Li, Q.; Jiang, W.; Liu, C.; He, J. The Constructing Method of Hierarchical Decision-Making Model in Air Combat. In Proceedings of the 2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 22–23 August 2020; IEEE: New York, NY, USA, 2020; pp. 122–125. [CrossRef]
13. Mulgund, S.; Harper, K.; Krishnakumar, K.; Zacharias, G. Air combat tactics optimization using stochastic genetic algorithms. In *SMC'98 Conference Proceedings, Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218), San Diego, CA, USA, 14 October 1998*; IEEE: New York, NY, USA, 1998; Volume 4, pp. 3136–3141. [CrossRef]
14. Lee, G.T.; Kim, C.O. Autonomous Control of Combat Unmanned Aerial Vehicles to Evade Surface-to-Air Missiles Using Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 226724–226736. [CrossRef]
15. Li, Y.; Lyu, Y.; Shi, J.; Li, W. Autonomous Maneuver Decision of Air Combat Based on Simulated Operation Command and FRV-DDPG Algorithm. *Aerospace* **2022**, *9*, 658. [CrossRef]
16. Cao, Y.; Kou, Y.-X.; Li, Z.-W.; Xu, A. Autonomous Maneuver Decision of UCAV Air Combat Based on Double Deep Q Network Algorithm and Stochastic Game Theory. *Int. J. Aerosp. Eng.* **2023**, *2023*, 3657814. [CrossRef]
17. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
18. Wang, Y.; Ren, T.; Fan, Z. Autonomous Maneuver Decision of UAV Based on Deep Reinforcement Learning: Comparison of DQN and DDPG. In Proceedings of the 2022 34th Chinese Control and Decision Conference (CCDC), Hefei, China, 21–23 May 2022; IEEE: New York, NY, USA, 2022; pp. 4857–4860. [CrossRef]

19. Chen, Y.; Zhang, J.; Yang, Q.; Zhou, Y.; Shi, G.; Wu, Y. Design and Verification of UAV Maneuver Decision Simulation System Based on Deep Q-learning Network. In Proceedings of the 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV), Shenzhen, China, 13 December 2020; IEEE: New York, NY, USA, 2020; pp. 817–823. [CrossRef]
20. Li, L.; Zhou, Z.; Chai, J.; Liu, Z.; Zhu, Y.; Yi, J. Learning Continuous 3-DoF Air-to-Air Close-in Combat Strategy using Proximal Policy Optimization. In Proceedings of the 2022 IEEE Conference on Games (CoG), Beijing, China, 21–24 August 2022; IEEE: New York, NY, USA, 2022; pp. 616–619. [CrossRef]
21. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.063472017. Available online: http://arxiv.org/abs/1707.06347 (accessed on 7 April 2023).
22. Lu, J.; Zhao, Y.-B.; Kang, Y.; Wang, Y.; Deng, Y. Strategy Generation Based on DDPG with Prioritized Experience Replay for UCAV. In Proceedings of the 2022 International Conference on Advanced Robotics and Mechatronics (ICARM), Guilin, China, 9–11 July 2022; IEEE: New York, NY, USA, 2022; pp. 157–162. [CrossRef]
23. Wei, Y.-J.; Zhang, H.-P.; Huang, C.-Q. Maneuver Decision-Making For Autonomous Air Combat Through Curriculum Learning And Reinforcement Learning With Sparse Rewards. *arXiv* **2023**, arXiv:2302.05838. Available online: http://arxiv.org/abs/2302.05838 (accessed on 7 March 2023).
24. Hu, Y.; Wang, W.; Jia, H.; Wang, Y.; Chen, Y.; Hao, J.; Wu, F.; Fan, C. Learning to Utilize Shaping Rewards: A New Approach of Reward Shaping. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 15931–15941.
25. Piao, H.; Sun, Z.; Meng, G.; Chen, H.; Qu, B.; Lang, K.; Sun, Y.; Yang, S.; Peng, X. Beyond-Visual-Range Air Combat Tactics Auto-Generation by Reinforcement Learning. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; IEEE: New York, NY, USA, 2020; pp. 1–8. [CrossRef]
26. Wang, A.; Zhao, S.; Shi, Z.; Wang, J. Over-the-Horizon Air Combat Environment Modeling and Deep Reinforcement Learning Application. In Proceedings of the 2022 4th International Conference on Data-driven Optimization of Complex Systems (DOCS), Chengdu, China, 28–30 October 2022; IEEE: New York, NY, USA, 2022; pp. 1–6. [CrossRef]
27. Hu, J.; Wang, L.; Hu, T.; Guo, C.; Wang, Y. Autonomous Maneuver Decision Making of Dual-UAV Cooperative Air Combat Based on Deep Reinforcement Learning. *Electronics* **2022**, *11*, 467. [CrossRef]
28. Zhan, G.; Zhang, X.; Li, Z.; Xu, L.; Zhou, D.; Yang, Z. Multiple-UAV Reinforcement Learning Algorithm Based on Improved PPO in Ray Framework. *Drones* **2022**, *6*, 166. [CrossRef]
29. Narvekar, S.; Sinapov, J.; Stone, P. Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 2536–2542. [CrossRef]
30. Schmidhuber, J. Learning to generate subgoals for action sequences. In Proceedings of the IJCNN-91-Seattle International Joint Conference on Neural Networks, Seattle, WA, USA, 8–12 July 1991; IEEE: New York, NY, USA, 1991; Volume II, p. 453. [CrossRef]
31. Rane, S. *Learning with Curricula for Sparse-Reward Tasks in Deep Reinforcement Learning*; Massachusetts Institute of Technology: Cambridge, MA, USA, 2020.
32. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, USA, 24 September 2017; IEEE: New York, NY, USA, 2017; pp. 23–30. [CrossRef]
33. Comanici, G.; Precup, D. Optimal Policy Switching Algorithms for Reinforcement Learning. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, Montreal, QC, Canada, 10–14 May 2010; Volume 1, pp. 709–714.
34. Frans, K.; Ho, J.; Chen, X.; Abbeel, P.; Schulman, J. Meta Learning Shared Hierarchies. *arXiv* **2017**, arXiv:1710.09767. Available online: http://arxiv.org/abs/1710.09767 (accessed on 20 February 2023).
35. Zhao, H.; Stretcu, O.; Smola, A.J.; Gordon, G.J. Efficient Multitask Feature and Relationship Learning. *PMLR* **2020**, *115*, 777–787.
36. Barto, A.G.; Mahadevan, S. Recent Advances in Hierarchical Reinforcement Learning. *Discret. Event Dyn. Syst.* **2003**, *13*, 41–77. [CrossRef]
37. Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R. Planning and acting in partially observable stochastic domains. *Artif. Intell.* **1998**, *101*, 99–134. [CrossRef]
38. Sutton, R.S.; Precup, D.; Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.* **1999**, *112*, 181–211. [CrossRef]
39. Eppe, M.; Gumbsch, C.; Kerzel, M.; Nguyen, P.D.H.; Butz, M.V.; Wermter, S. Intelligent problem-solving as integrated hierarchical reinforcement learning. *Nat Mach Intell* **2022**, *4*, 11–20. [CrossRef]
40. Wen, Z.; Precup, D.; Ibrahimi, M.; Barreto, A. On Efficiency in Hierarchical Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6708–6718.
41. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602. Available online: http://arxiv.org/abs/1312.5602 (accessed on 7 April 2023).
42. Littman, M.L. A tutorial on partially observable Markov decision processes. *J. Math. Psychol.* **2009**, *53*, 119–125. [CrossRef]
43. White, D.J. A Survey of Applications of Markov Decision Processes. *J. Oper. Res. Soc.* **1993**, *44*, 1073–1096. [CrossRef]
44. Wang, L.; Wei, H. Research on Autonomous Decision-Making of UCAV Based on Deep Reinforcement Learning. In Proceedings of the 2022 3rd Information Communication Technologies Conference (ICTC), Nanjing, China, 6–8 May 2022; IEEE: New York, NY, USA, 2022; pp. 122–126. [CrossRef]

45. Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; Abbeel, P. Benchmarking Deep Reinforcement Learning for Continuous Control. *PMLR* **2016**, *48*, 1329–1338.
46. Vogeltanz, T. A Survey of Free Software for the Design, Analysis, Modelling, and Simulation of an Unmanned Aerial Vehicle. *Arch. Comput. Methods Eng.* **2016**, *23*, 449–514. [CrossRef]
47. Chandak, Y.; Theocharous, G.; Kostas, J.E.; Jordan, S.M.; Thomas, P.S. Learning Action Representations for Reinforcement Learning. *PMLR* **2019**, *97*, 941–950.
48. Pope, A.P.; Ide, J.S.; Micovic, D.; Diaz, H.; Twedt, J.C.; Alcedo, K.; Walker, T.T.; Rosenbluth, D.; Ritholtz, L.; Javorsek, D. Hierarchical Reinforcement Learning for Air Combat At DARPA's AlphaDogfight Trials. *IEEE Trans. Artif. Intell.* **2022**, *Early Access*. [CrossRef]
49. Chen, W.; Gao, C.; Jing, W. Proximal policy optimization guidance algorithm for intercepting near-space maneuvering targets. *Aerosp. Sci. Technol.* **2023**, *132*, 108031. [CrossRef]