aerospace

MDPI

Article

Autonomous Maneuvering Decision-Making Algorithm for Unmanned Aerial Vehicles Based on Node Clustering and Deep Deterministic Policy Gradient

Xianyong Jing [1,*], Fuzhong Cong [1], Jichuan Huang [2], Chunyan Tian [1] and Zikang Su [3]

[1] Aviation University of Air Force, Changchun 130022, China; congfz67@126.com (F.C.); 8202200529@csu.edu.cn (C.T.)
[2] The 93147th Unit of PLA Air Force, Chengdu 610041, China; jichuan1980@163.com
[3] College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; zk_su@nuaa.edu.cn
* Correspondence: xianyong_1983@163.com; Tel.: +86-18143096755

Abstract: Decision-making for autonomous maneuvering in dynamic, uncertain, and nonlinear environments represents a challenging frontier problem. Deep deterministic policy gradient (DDPG) is an effective method to solve such problems, but it is found that complex strategies require extensive computation and time in the learning process. To address this issue, we propose a node clustering (NC) method, inspired by grid clustering, integrated into the DDPG algorithm for the learning of complex strategies. In the NC method, the node membership degree is defined according to the specific characteristics of the maneuvering decision-making problem, and error handling strategies are designed to reduce the number of transitions in the replay database effectively, ensuring that the most typical transitions are retained. Then, combining NC and DDPG, an autonomous learning and decision-making algorithm of maneuvering is designed. The algorithm flow and the pseudo-code of the algorithm are given. Finally, the NC_DDPG algorithm is applied to a typical short-range air combat maneuvering decision problem for verification. The results show that the NC_DDPG algorithm significantly accelerates the autonomous learning and decision-making process under both balanced and disadvantageous conditions, taking only about 77% of the time required by Vector DDPG. The scale of NC impacts learning speed; the simulation results across five scales indicate that smaller clustering scales significantly increase learning time, despite a high degree of randomness. Compared with Twin Delayed DDPG (TD3), NC_DDPG consumes only 0.58% of the time of traditional TD3. After applying the NC method to TD3, NC_DDPG requires approximately 20–30% of the time of NC_TD3.

Keywords: unmanned aerial vehicle (UAV); autonomous decision-making; maneuvering decision-making; deep deterministic policy gradient (DDPG); node clustering (NC); reinforcement learning (RL)

Citation: Jing, X.; Cong, F.; Huang, J.; Tian, C.; Su, Z. Autonomous Maneuvering Decision-Making Algorithm for Unmanned Aerial Vehicles Based on Node Clustering and Deep Deterministic Policy Gradient. *Aerospace* **2024**, *11*, 1055. https://doi.org/10.3390/aerospace11121055

Received: 30 September 2024
Revised: 17 December 2024
Accepted: 18 December 2024
Published: 23 December 2024

1. Introduction

UAV autonomous maneuvering decision-making is pivotal in various scenarios, including maneuvering target tracking, dynamic route planning, and short-range air combat. Addressing UAV maneuvering decision-making in dynamic, uncertain, and non-linear environments is a formidable frontier challenge [1–3]. Among these, short-range air combat represents a quintessential and particularly challenging application scenario.

Numerous scholars have delved into the autonomous maneuvering decision-making problem of air combat, employing a variety of innovative methods such as differential game [4], impact graphs [5], reinforcement learning (RL) [6–8], neural networks [9], immune systems [10], expert systems [11], genetic fuzzy trees [12], fuzzy reasoning [13], and Approximate Dynamic Programming (ADP) [6]. These studies have made significant strides,

offering valuable references and a theoretical foundation for autonomous decision-making problems. However, these methods also have certain limitations, including the need for an accurate air combat environment model, reliance on substantial "expert knowledge", large decision-making space, and low decision-making efficiency.

Short-range air combat maneuvering decision-making presents several challenges. Firstly, from the perspective of air combat, the nonlinear changes in the state of existence and the stepwise jumps in strategy necessitate powerful tools to capture these nonlinear dynamics. Secondly, establishing a decision model requires a significant amount of empirical data to ensure accurate decision-making across various states. However, transitions in air combat are difficult to obtain, necessitating novel intelligent methods for resolution. Thirdly, maneuvering decision-making is inherently complex, with air combat status influenced by factors such as weapons, targets, environment, aircraft, and target characteristics, making it an exceedingly intricate decision-making problem. Additionally, air combat transitions must be considered from a global and process-oriented perspective, rather than focusing solely on short-term gains and losses. These characteristics of air combat maneuver decision problems pose significant challenges to traditional mathematical methods.

The emergence and development of deep reinforcement learning (DRL) [14,15] have resulted in a powerful tool for tackling air combat maneuvering decision-making problems. DRL integrates two machine learning paradigms: reinforcement learning and deep learning. It retains the "trial-and-error" feedback mechanism of reinforcement learning while leveraging the robust mapping capabilities of deep neural networks. This combination enables an ideal self-learning mode, making DRL naturally suited for air combat maneuvering decision-making challenges.

Many researchers began to deal with air combat maneuvering decision-making problems based on DRL, and several intelligent air combat projects based on such methods have gradually been proposed [16,17]. Many scholars have adopted the Deep Q-network (DQN) method for maneuvering autonomous learning and decision-making, such as Liu Pin [8], Yang Qingming [18], Li Yongfeng [19], etc. However, the action space of the DQN adopts a discrete form of maneuvering action library, which struggles to reflect the maneuvering actions in actual air combat. New deep reinforcement learning algorithms, such as PPO, TPRO, A3C, and TD3, continue to emerge, showcasing significant advantages in terms of convergence speed and accuracy. These algorithms present new options for solving maneuvering decision-making problems, but their applicability requires further testing and verification. The study in [20] focuses on the interpretability of air combat algorithms by combining reinforcement learning and reward decomposition methods, illustrating decision-making processes in various tactical situations. This allows us to understand agents' behavior in greater detail, improving our overall trust in AI applications. It is critical to ensuring that AI agents perform optimally in their intended environment.

The deep deterministic policy gradient (DDPG) algorithm [21] enables autonomous learning and decision-making for problems with continuous states and actions, which aligns well with the characteristics of maneuvering decision-making problems.

However, when facing complex air combat situations, the DDPG algorithm still has difficulties in learning correct strategies due to the serious nonlinearity of the air combat decision state space, as well as learning errors in neural networks. Therefore, the efficiency of the DDPG algorithm needs to be further improved. Focusing on enhancing the efficiency of DDPG for maneuvering decision-making, many scholars have conducted fruitful work. Yang [22] used an optimization algorithm to generate prior air combat maneuver action values, which effectively improved the learning efficiency of DDPG. Jing [23] improved the transition playback strategy of DDPG, controlled the size of the transition database based on the concentration adjustment mechanism, and successfully realized the independent learning of maneuvering strategy under balance and disadvantage conditions. Furthermore, one study [24] proposed an autonomous maneuvering decision-making model based on the Soft Actor-Critic (SAC) algorithm, which uses a small amount of expert transitions to increase sample diversity and reconstructs the transition replay buffer, thereby improving

the exploration and utilization efficiency of maneuver strategy learning. The Twin Delayed deep deterministic policy gradient (TD3) algorithm optimizes the problem of value overestimation and has shown good performance in some continuous control problems. In [25], autonomous learning and decision-making on maneuvering strategies in complex dynamic environments were implemented based on TD3.

Integrating DDPG with other algorithms is an effective method to enhance its efficiency, such as through transfer learning [26] and meta-learning [27]. These approaches can significantly reduce the training difficulty of strategies by leveraging pre-acquired knowledge, although they require the prior acquisition of specialized knowledge. Xie [28] combined DDPG with heuristic rules to investigate a method for UAV autonomous tracking and landing, successfully realizing the learning of optimal strategies for continuous autonomous landing. Li [29] proposed an autonomous air combat maneuvering decision method; a prediction method based on Particle Swarm Optimization Radial Basis Function (PSO-RBF) was designed, followed by an improved DDPG strategy, which enhanced the algorithm's convergence speed. In [30], a new layered decision framework was designed to tackle the six-degrees-of-freedom (6-DOF) aircraft air combat challenge. The decision-making process within this framework was bifurcated into two layers, each addressed separately using reinforcement learning (RL), resulting in improved success rates and efficiency. Liu [31] addressed the sparse reward problem by devising a comprehensive reward function, thereby enhancing the autonomous training efficiency of air combat algorithms through the adoption of centralized training and distributed implementation concepts. In [32], target trajectories were predicted using a Backpropagation (BP) network and a Long Short-Term Memory (LSTM) network. Tracking points were converted into virtual pursuit angles as tracking angle commands using angle guidance laws. A variable-scale reinforcement learning algorithm for short-range air combat was established based on the Proximal Policy Optimization (PPO) algorithm, achieving a high victory rate.

Scholars from various fields have contributed improvements to address the limitations of the DDPG algorithm. Qi [33] targeted the low sampling efficiency during free exploration by augmenting the transition replay database with a batch of teaching data, which significantly enhanced learning efficiency and notably improved the success rate of crawling tasks. In [34], Zhang integrated the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm and fuzzy logic to enhance learning efficiency and convergence. These improvements were made across agent exploration strategies, network training algorithms, and overall algorithm implementation, leading to the proposal of an improved DDPG method based on a double-layer Backpropagation (BP) neural network. Yu [35] introduced DDPG for device selection and training configuration in Federated Learning (FL), proposing a clustering-based aggregation strategy within FL that achieved significant improvements over existing benchmarks. Hu [36] proposed an Adaptive Parameter Space Noise with Parallel Experience Replay, Double Critics, and Double Actors DDPG (AP-D4PG) algorithm. This novel approach demonstrated faster convergence and enhanced robustness compared to the traditional DDPG algorithm.

It is evident that improving the efficiency of the deep deterministic policy gradient (DDPG) algorithm is an issue of wide concern. During the operation of DDPG, the agent generates "trial-and-error" transitions through continuous interaction with the environment. These transitions are then used to create batch sets for network training via the replay mechanism. However, due to the complexity of air combat maneuvering decision-making, obtaining the optimal strategy in certain situations is challenging, leading to a greater number of "trial-and-error" processes and the generation of a multitude of transitions. As the algorithm operates, a vast accumulation of transitions occurs in the replay database. This not only demands a significant amount of operational memory, but also prolongs the transition replay process, severely impacting the algorithm's efficiency.

In general, there are still some problems in the research of autonomous maneuvering decision-making algorithms for UAVs:

- Enhancing the autonomous learning capability and efficiency of the algorithms under complex conditions, reducing the computational burden, and ultimately shortening the learning process;
- Improving the modeling of autonomous maneuvering problems in short-range air combat to include a more comprehensive set of influencing factors, thereby reducing the difficulty of convergence.

In response to these challenges, we propose an autonomous learning and decision-making algorithm for UAV maneuvering in short-range air combat that incorporates node clustering and DDPG. The main contributions are as follows:

- We propose an improved close-range air combat description model, enhancing the state representation in accordance with the characteristics of close-range air combat;
- A node fuzzy clustering method is introduced, which processes the transition pool to eliminate similar transitions and retain the most representative ones;
- By integrating the problem model, node clustering algorithm, and DDPG, we design an autonomous learning and decision-making algorithm for short-range air combat maneuvers. The algorithm flow and main steps are detailed in the pseudocode provided, and extensive simulation verification is conducted.

2. Methodology

According to the characteristics of short-range air combat, the height variable is introduced into the state vector to improve the problem description model of short-range air combat and better reflect the situations of short-range air combat.

2.1. State Description

The correct description of the problem state is a prerequisite for achieving ideal results. The state vector of the problem is represented by vector S, which should contain all factors affecting the future state.

The spatial relative position of the two sides can be described based on the following variables: the azimuth v_1 and the pitch angle μ, the azimuth v_2 and the pitch angle $-\mu$ of U_1 with respect to U_2, and the relative distance d. The definitions of each variable are shown in Figure 1, $OXYZ$ is an inertial ground coordinate system. Among them,

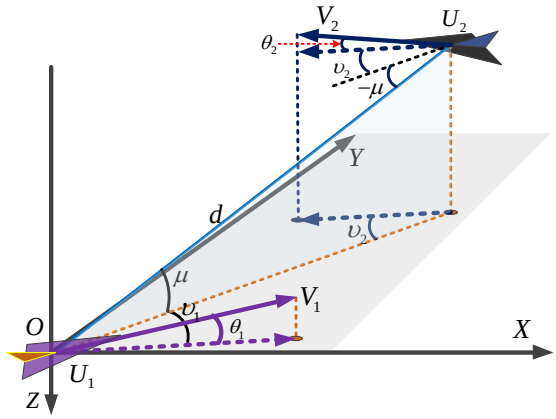

Figure 1. Spatial position diagram of the two sides in short-range air combat.

v_1 is the angle between the projection of V_1 in the horizontal plane OXY and the projection of target line U_1U_2 in the OXY plane;

v_2 is the angle between the projection of V_2 in the OXY plane and the projection of U_1U_2 in the OXY plane;

μ is the angle between U_1U_2 and the OXY plane.

The variables that describe the motion state of both sides mainly include velocity, i.e., V_1 and V_2, and their track inclination angles, represented by θ_1,θ_2.

In fact, environmental factors will also affect the maneuvering decision-making of the aircraft. When the flight altitude changes, the air density will also change, and the available overload of the aircraft will change. The available overload will limit the scope of decision-making. Therefore, the flight altitudes H_1 and H_2 should also be added to the current state.

According to the above analysis, the current state can be expressed as

$$S = [d, v_1, v_2, \mu, v_1, v_2, \theta_1, \theta_2, H_1, H_2] \tag{1}$$

2.2. Expression of the Attack Area and Achievement of the Attack Conditions

Generally, the launch of infrared short-range missiles needs to meet the following three conditions (as shown in Figure 2):

(1) U_1 needs to enter a certain zone around U_2, denoted by D;
(2) U_2 also needs to be in a certain zone around U_1, denoted by $D\prime$;
(3) The distances between the two sides $d \in [d_{\max}, d_{\min}]$, $d_{\min}$, and $d_{\max}$ are the minimum and maximum launching distances of the missile, respectively.

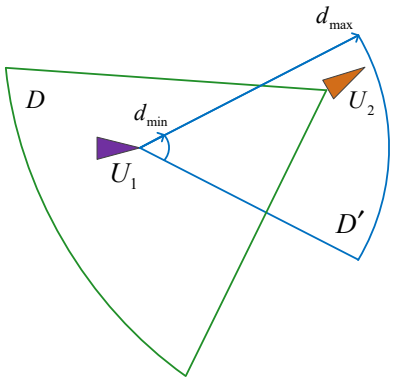

Figure 2. Schematic diagram of achieving attack conditions.

The calculation of missile attack zone is very complex and dynamically changing, but not the focus of this article, so a simplified attack zone is adopted by setting D, $D\prime$, $d_{\min}$, and $d_{\max}$ to be constant.

2.3. UAV Motion Dynamics Model

In modern air combat, it is necessary to consider the energy loss and replenishment of UAVs during maneuvering, so a particle motion model, dynamic model, and aerodynamic model are established, respectively.

2.3.1. Particle Motion Model

A three-degree-of-freedom particle motion model of a UAV is established in the inertial ground coordinate system $OXYZ$, as Figure 1 shows.

Suppose that θ is the track inclination and ψ is the track deflection angle of V_1. The particle motion equation can be expressed as

$$\begin{cases} \dot{x} = V\cos\theta\cos\psi = v_x \\ \dot{y} = V\cos\theta\sin\psi = v_y \\ \dot{z} = V\sin\theta = v_z \end{cases} \tag{2}$$

2.3.2. Aerodynamic Model

Maneuvering actions will change the motion state of the UAV, so the aerodynamic force must be calculated in the problem model to reflect this basic law. In the velocity coordinate system $(OXYZ)_V$, lift force F_L and resistance f are calculated as follows:

$$\begin{cases} F_L = \frac{1}{2}\rho V^2 S_{Uav} C_F(V,a) \\ f = \frac{1}{2}\rho V^2 S_{Uav} C_x(V,a) \end{cases} \tag{3}$$

ρ is the air density and S_{Uav} is the equivalent wing area; $C_F(V,\alpha)$ and $C_x(V,\alpha)$ are the lift and resistance coefficients, respectively.

In order to simplify the model, sideslip is ignored, so there is only attack angle α between the aircraft coordinate system $(OXYZ)_p$ and $(OXYZ)_v$, that is, the angle between OX_p and OX_v, as shown in Figure 3. Therefore, the transformation matrix from $(OXYZ)_p$ to $(OXYZ)_v$ can be simplified as follows:

$$T_{v,p} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{4}$$

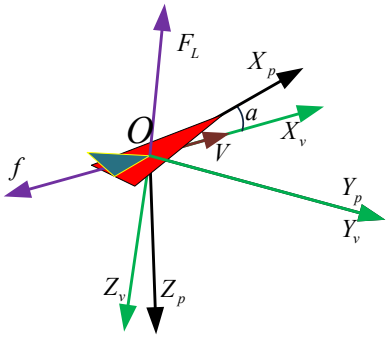

Figure 3. Aircraft coordinate system and velocity coordinate system and their geometric relationship.

2.3.3. Dynamic Model

Reference [23] establishes the dynamic model. If the velocity roll angle is γ^v, the track deflection angle is ψ, and the track inclination angle is θ, the conversion matrix from $OXYZ$ to $(OXYZ)_v$ is

$$T_{v,e} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\gamma\sin\theta\cos\psi - \cos\gamma\sin\psi & \sin\gamma\sin\theta\sin\psi + \cos\gamma\cos\psi & \sin\gamma\cos\theta \\ \cos\gamma\sin\theta\cos\psi + \sin\gamma\sin\psi & \cos\gamma\sin\theta\sin\psi - \sin\gamma\cos\psi & \cos\gamma\cos\theta \end{bmatrix} \tag{5}$$

If the expressions of F_L and f in $OXYZ$ are, respectively, $F' = \begin{bmatrix} F'_x & F'_y & F'_z \end{bmatrix}^T$ and $f' = \begin{bmatrix} f'_x & f'_y & f'_z \end{bmatrix}^T$, then

$$F' = \begin{bmatrix} F'_x & F'_y & F'_z \end{bmatrix}^T = T_{v,e}^{-1} \cdot [0 \quad 0 \quad -F_L]^T. \tag{6}$$

$$f' = \begin{bmatrix} f'_x & f'_y & f'_z \end{bmatrix}^T = T_{v,e}^{-1} \cdot [-f \quad 0 \quad 0]^T. \tag{7}$$

Assuming that the engine installation angle is 0, engine thrust F_T in $OXYZ$ can be expressed as $F'_T = \begin{bmatrix} F'_{Tx} & F'_{Ty} & F'_{Tz} \end{bmatrix}$; then, there is

$$F'_T = \begin{bmatrix} F'_{Tx} & F'_{Ty} & F'_{Tz} \end{bmatrix} = T_{v,e}^{-1} \cdot T_{v,p} \cdot [F_T \quad 0 \quad 0]^T. \tag{8}$$

The three-degrees-of-freedom dynamic model can be obtained as follows:

$$\begin{cases} \dot{v}_x = (F'_x + f'_x + F'_{Tx})/m \\ \dot{v}_y = (F'_y + f'_y + F'_{Ty})/m \\ \dot{v}_z = (F'_z + f'_z + F'_{Tz})/m + g \end{cases} \tag{9}$$

where m is the mass and g is the gravitational acceleration.

2.4. Action Representation

Assuming that the UAV adopts the BTT control mode, the required overload can be generated by controlling the change of attack angle α, speed roll angle γ^v, and engine thrust F_T to achieve the maneuvering purpose. Then, the action in reinforcement learning can be expressed as

$$Action = [\alpha, \gamma^v, F_T] \tag{10}$$

3. Design of Air Close-Combat Maneuvering Decision-Making Algorithm Based on the Node Clustering Method and DDPG

The DDPG algorithm has a natural applicability to air combat maneuvering decision-making problems for several reasons. Firstly, it retains the characteristics of reinforcement learning (RL), exploring and learning based on a "trial-and-error" mechanism, which can address the issue of lacking transitions in air combat. Secondly, DDPG employs both a value network and a policy network, enabling it to search within continuous state and action spaces, which is essential for the typical continuous decision-making nature of air combat maneuvering. Thirdly, DDPG utilizes neural networks to implement value and policy functions, offering excellent nonlinear mapping capabilities for complex state and action spaces.

Despite its suitability for air combat maneuvering decisions, DDPG has encountered some difficulties, particularly the explosion of transition numbers. Each "trial-and-error" process in air combat generates a relatively large number of transitions, and when the maneuvering strategy is complex, a significant amount of "trial-and-error" processes are required, leading to an enormous volume of air combat transitions. While more transitions theoretically lead to better network learning, storing all these air combat transitions in the replay database can be disastrous for the algorithm's operation, greatly increasing computational load and time. Moreover, many of these transitions are similar and can be discarded. Only representative air combat transitions need to be retained. This necessitates a method to classify similar transitions into one category and then use a typical transition as the representative of this category. Therefore, a node clustering method is proposed in this paper.

3.1. Node Clustering Algorithm

Clustering method is an effective way to solve classification problems. Common clustering methods include K-means clustering [37,38], grid-based clustering [39,40], and so on. However, since the main purpose of this paper is to exclude similar transitions, we only refer to the ideas of grid division and density calculation, and do not classify, but only exclude. Based on this purpose, using the ideas of grid clustering for reference, the following exclusion algorithm is designed:

1. Determine grid scales and set nodes based on these scales to grid the data space. Fuzzy each dimension of each set of data in the data space towards its nearest node. A set of data in the data space represents a transition. Suppose that a standardized set of data contains u dimensions, that is $x_i = [x_{i1}, x_{i2}, \ldots, x_{iu}]$. For any dimension x_{iu} of x_i, the corresponding fuzzification scale is represented by d_u; then, u scales form the scale vector $d = [d_1, d_2, \ldots, d_u]$. The scale vector d, once set, does not change.

2. For the uth dimension x_{iu} of a set of data x_i, using the x_i integer divided by d_u equals p_u, and the remaining remainder is e_u. Then,

$$x_{iu} = p_u \cdot d_u + e_u \tag{11}$$

Node fuzzy x_{iu} as follows:

$$\begin{cases} x_{iu} \leftarrow x'_{iu} = p_u \cdot d_u, e_u < \frac{1}{2} d_u \\ x_{iu} \leftarrow x'_{iu} = (p_u + 1) \cdot d_u, e_u \geq \frac{1}{2} d_u \end{cases} \tag{12}$$

Then, the node fuzzy clustering result $x'_i = \left[x'_{i1}, x'_{i2}, \ldots, x'_{iu}\right]$ of x_i is obtained. Calculate the Euclidean distance between x_i and x'_i, and express it in l_i.

$$l_i = \left\| x_i - x'_i \right\| \tag{13}$$

l_i is the membership degree of x_i with respect to node x'_i.

3. x'_j represents the results of another set of data x_j after node fuzzy clustering. After the node fuzzy clustering in step 2, if the node fuzzy clustering results of x_i and x_j are the same, that is,

$$x'_i = x'_j \tag{14}$$

it can be considered that the two sets (transition) x_i and x_j are similar, and one of them can be excluded.

4. If the two sets of x_1 and x_2 data are similar, one of them is excluded according to the size of their node membership l. In general, the smaller the Euclidean distance between a set of data x_i and the node data set x'_i, it can be considered that the closer x_i is to x'_i, the more x_i can reflect the characteristics of the node data set x'_i. Therefore, it is possible to determine which set of data is retained based on the Euclidean distance l.

The advantage of using this approach is that it can reduce the memory requirements of the algorithm, and only the typical transitions involved in the database need to be stored (i.e., the other useless parts of the grid data space do not need to be stored).

In RL, each transition usually includes the following elements:

$$\{S_t, A_t, r_{t+1}, S_{t+1}\} \tag{15}$$

Since the values of r_{t+1} and S_{t+1} are determined by S_t and A_t, node fuzzy clustering is only performed on S_t and A_t. If S_t and A_t in two transitions tend to the same node after being clustered, that is,

$$\begin{array}{l} \{S_{1t}, A_{1t}, r_{1,t+1}, S_{1,t+1}\} -- > \{S'_{1t}, A'_{1t}, r_{1,t+1}, S_{1,t+1}\} \\ \{S_{2t}, A_{2t}, r_{2,t+1}, S_{2,t+1}\} -- > \{S'_{1t}, A'_{1t}, r_{2,t+1}, S_{2,t+1}\} \end{array} \tag{16}$$

then one set of transitions can be excluded according to the Euclidean distance minimization method described above.

3.2. Autonomous Short-Range Air Combat Maneuvering Decision-Making Algorithm Based on Fuzzy Node Clustering and DDPG

As shown in Figure 4, following the algorithm framework of DDPG, four neural networks are also adopted in this algorithm: two actor policy networks with the same structure, namely Actor_online network $NN(\theta^{\pi})$ and Actor_target network $NN(\theta^{\pi-})$, and two critic evaluation networks with the same structure, namely Critic_online network $NN(\theta^{Q})$ and Critic_target network $NN(\theta^{Q-})$.

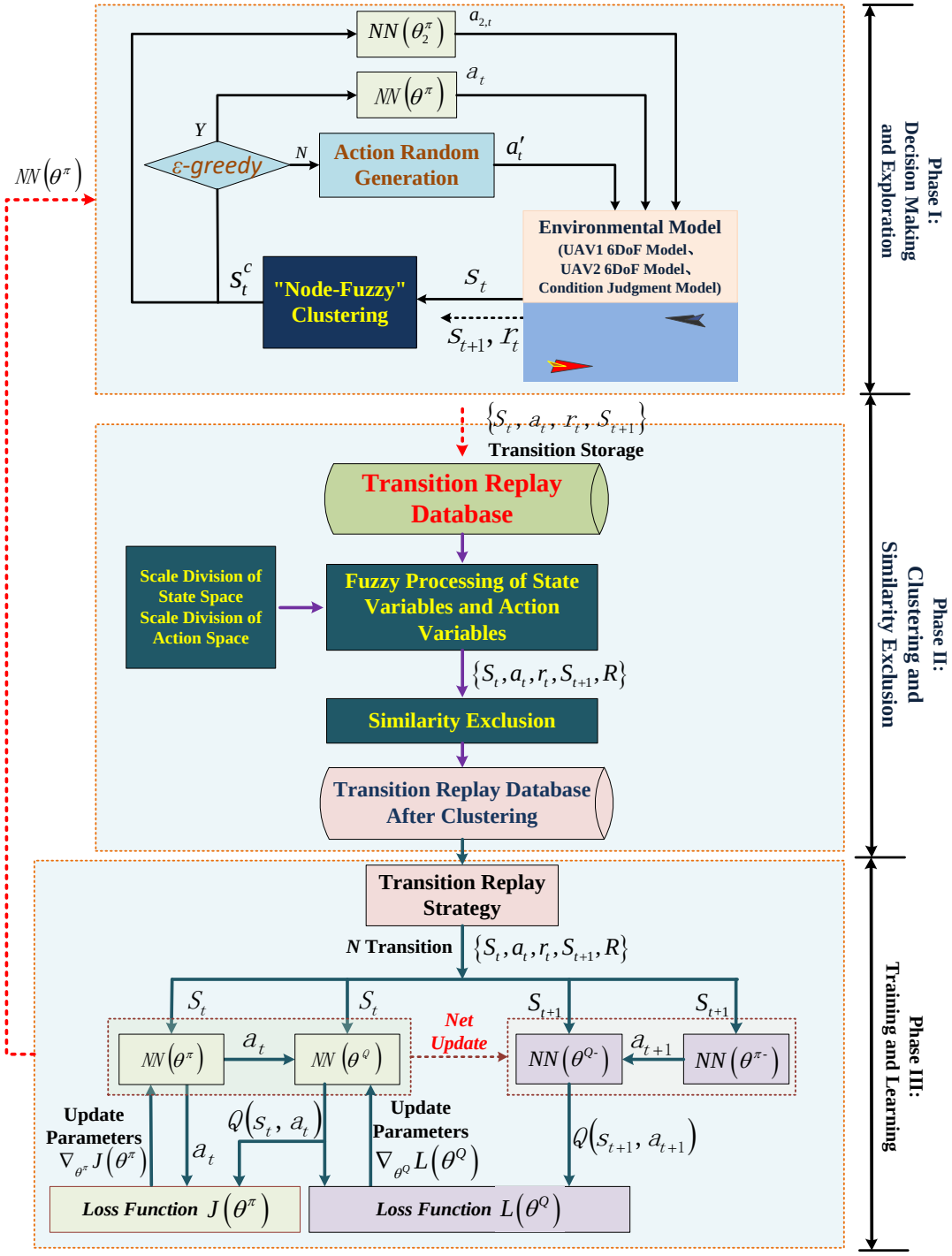

Figure 4. Flow chart of autonomous decision-making algorithm for short-range air combat maneuvering based on DDPG and node clustering.

The role of $NN(\theta^{\pi})$ is equivalent to the strategy function $\pi(\cdot)$. θ^{π} represents the network parameters. Then,

$$Action = \pi(S|\theta^{\pi}) \tag{17}$$

The role of $NN(\theta^{Q})$ is equivalent to the value function $Q(S, Action|\theta^{Q})$, where θ^{Q} represents the network parameter. Setting $NN(\theta^{\pi-})$ and $NN(\theta^{Q-})$ can enhance the stability of the algorithm learning process.

The overall framework of the algorithm is shown in Figure 4, and its running process can be divided into three phases:

- Phase 1—Decision-making and exploration;
- Phase 2—Node fuzzy clustering;
- Phase 3—Learning and training.

The main steps and content of each phase are described below.

3.2.1. Decision and Exploration Phase

In this phase, the main steps of the algorithm are as follows:

1. Obtain the current state S_t and take it as the input of the U_1 policy network $NN(\theta^{\pi})$. The output is the action $a_t = \begin{bmatrix} n_y & \gamma & F^T \end{bmatrix}$ for U_1. Take S_t as the input of the U_2 policy network, and the action a_{2t} to be performed by U_2 can also be obtained.
2. U_1 performs action a_t, U_2 performs action a_{2t}, and the current state becomes S_{t+1}. According to the changes in state, U_1 receives a return r_t.
3. Based on this process of state change, a transition $\{S_t, a_t, r_t, S_{t+1}\}$ can be obtained and stored in the temporary transition database D1.
4. End condition judgment.

The pseudo code of the main program and phase I is as Algorithm 1 shows.

Algorithm 1: Pseudo code of the main program

1: Initialize transition replay database D to capacity $N_{\mathrm{D},}$ initialize temp transition database D_1 to capacity N_{D1};
2: Initialize the critic network $NN(\theta^{Q})$ and the actor network $NN(\theta^{\pi})$ randomly. Initialize the target network $NN(\theta^{Q-}) = NN(\theta^{Q})$ and $NN(\theta^{\pi-}) = NN(\theta^{\pi})$.
3: Initialize Parameters: random probability ε, learning rate τ, maximum episode Ep, target networks update interval f_u, training frequency f_t, node clustering frequency f_{nc}, discount ratio γ, maximum step T, mini-batch size N.
4: For episode e = 1 to Ep do
5: Observe initial states of UAVs to obtain the current situation: speed, distance and azimuth of both sides. Height difference. $S = [d, v, v, \mu, v_1, v_2, \theta_1, \theta_2, H_1, H_2]$.
6: If t <= T do
7: Maneuver actions decision of both sides based on s_t with $\varepsilon - greedy$. Action: $a_t = [\alpha, \gamma, F_T]$.
8: Perform action a_t, get reward r_t and new state s_{t+1}.
9: Save transition $\{s_t, a_t, r_{t+1}, s_{t+1}\}$ into temp database D_1.
10: End if
11: If the remainder of e/f_{nc} is equal to 0
12: Clustering and similarity exclusion on D and D_1 based on algo. 2.
13: End if
14: If the remainder of e/f_t is equal to 0
15: Training $NN(\theta^{Q})$ and $NN(\theta^{\pi})$ based on algo. 3.
16: End if
17: If the remainder of e/f_u is equal to 0
18: Soft updating. $\theta^{\pi^-} \leftarrow (1-\tau)\theta^{\pi^-} + \tau\theta^{\pi}$; $\theta^{Q^-} \leftarrow (1-\tau)\theta^{Q^-} + \tau\theta^{Q}$.
19: End if
20: End for

3.2.2. Node Clustering Phase

In order to improve the algorithmic efficiency, a temporary database D1 is set in the algorithm in this paper. The empirical data generated in the exploration of stage 1 is first stored in D1; then, node clustering and similar exclusion are performed on D1. Finally, the empirical data in D1 and D are compared for similar exclusion. The main steps of phase II are as follows:

1. Set the node division scale of each state variable in the state vector S and obtain the division scale vector $d_S = [d_{S1}, d_{S2}, \ldots, d_{Sm}]$ of the state node. The scale of each action variable in the action vector is also set, and the scale vector $d_A = [d_{A1}, d_{A2}, \ldots, d_{Ak}]$ of the action node is obtained.

2. For transition $\{S_t, A_t, r_{t+1}, S_{t+1}\}$, node clustering is performed on S_t and S_{t+1} based on d_S, and node clustering is performed on A_t based on d_A. The transition $\{S_{t.node}, A_{t,node}, r_{t+1}, S_{t+1,node}\}$ after node clustering is obtained. Suppose that the transition obtained by another transition $\{S'_t, A'_t, r'_{t+1}, S'_{t+1}\}$ after node clustering is $\{S'_{t.node}, A'_{t,node}, r'_{t+1}, S'_{t+1,node}\}$, if and only if

$$\left[S'_{t.node}, r'_{t+1}, S'_{t+1,node}\right] = [S_{t.node}, r_{t+1}, S_{t+1,node}] \tag{18}$$

$\{S_t, A_t, r_{t+1}, S_{t+1}\}$ and $\{S'_t, A'_t, r'_{t+1}, S'_{t+1}\}$ are considered similar. If they are not similar, both transitions need to be preserved.
3. Calculate the similarity R between each experience $\{S_t, A_t, r_{t+1}, S_{t+1}\}$ in D1 and its corresponding node experience $\{S_{t.node}, A_{t,node}, r_{t+1}, S_{t+1,node}\}$. See formula (18). The similarity R and the labels of the node experience are retained in the transition as $\{S_t, A_t, r_{t+1}, S_{t+1}, R, n\_lable\}$;
4. If two transitions are similar, the similarity R of the two transitions is compared. The smaller R is, the higher the similarity is. Retain one transition and exclude the other transition.
5. Compare the transitions in D1 and D pairwise to exclude similar transitions with low similarity.
6. Merge D1 and D.

The pseudo code of fuzzy node clustering and similar exclusion is as Algorithm 2 shows.

Algorithm 2: Pseudo code of node clustering and similarity exclusion

1: Initialize state node division scale Id_S, initialize action node division scale Id_A.
2: Obtain the number of experiences in D1 and D, expressed in N_{D1} and N_D respectively.
3: For n = 1 to N_{D1} do
4: Node clustering of state variables in S_t and S_{t+1}, node clustering of action variables in A_t, then experience after node clustering can be obtained as $\{S_{t.node}, A_{t,node}, r_{t+1}, S_{t+1,node}\}$.
5: Calculate the similarity R between $\{S_t, A_t, r_{t+1}, S_{t+1}\}$ and $\{S_{t.node}, A_{t,node}, r_{t+1}, S_{t+1,node}\}$.
6: Judge whether $\{S_t, A_t, r_{t+1}, S_{t+1}\}$ is similar to the other experiences that have been node clustered. If there is similar experience, it shall be excluded according to the similarity R.
7: End for
8: Obtain the number of experiences in D_1, expressed in N'_{D1}
9: For n = 1 to N'_{D1} do
10: For m = 1 to N_D do
11: Judge whether $\{S_{t,n}, A_{t,n}, r_{t+1,n}, S_{t+1,n}\}$ in D_1 is similar to $\{S_{t,m}, A_{t,m}, r_{t+1,m}, S_{t+1,m}\}$ in D. If they are similar experience, one of them shall be excluded according to their similarity R.
12: End for
13: End for

3.2.3. Network Learning Phase

In the learning phase of phase III, the methods and steps of network learning follow the principles of DDPG. The main steps are as follows:

1. N pieces of transitions are obtained according to the transition replay strategy to form a mini-batch set. The replay strategy is as follows:
2. For any transition $\{S_t, a_t, r_t, S_{t+1}\}$, the current state action value $Q(S_t, a_t|\theta^Q)$ is obtained according to the value network $NN(\theta^Q)$ and $[S_t, a_t]$.
3. The best action a^-_{t+1} in this state is obtained according to the target strategy network $NN(\theta^{\pi-})$ and S_{t+1} in the transition.

4. Taking $[S_{t+1}, a^-_{t+1}]$ as input, the state action value $Q\left(S_{t+1}, a^-_{t+1}\middle|\theta^{Q-}\right)$ is obtained according to the target value network $NN\left(\theta^{Q-}\right)$, and the expected value $r_t + \gamma \cdot Q\left(S_{t+1}, a^-_{t+1}\middle|\theta^{Q-}\right)$ of the evaluation network of this transition can be further obtained.
5. Construct the error function $L\left(\theta^Q\right)$ of the current value network $NN\left(\theta^Q\right)$:

$$L\left(\theta^Q\right) = E\left(\left(r_{t+}\gamma Q\left(S_{t+1},\ a^-_{t+1}\middle|\theta^{Q-}\right) - Q\left(S_t, a_t\middle|\theta^Q\right)\right)^2\right) \tag{19}$$

6. Update $NN\left(\theta^Q\right)$ based on $L\left(\theta^Q\right)$ and the error gradient backward propagation algorithm, where the gradient is expressed as

$$\nabla_{\theta_Q} L(\theta_Q) = \frac{\partial L\left(\theta^Q\right)}{\partial \theta_Q} = \left(r_{t+}\gamma Q\left(S_{t+1},\ a^-_{t+1}\middle|\theta^{Q-}\right) - Q\left(S_t, a_t\middle|\theta^Q\right)\right)\frac{\partial Q\left(S_t, a_t\middle|\theta^Q\right)}{\theta^Q} \tag{20}$$

7. Update the current strategy network $NN(\theta^\pi)$ with the error gradient back propagation algorithm. The loss function of $NN(\theta^\pi)$ is represented by $L(\theta^\pi)$. The gradient of $L(\theta^\pi)$ with respect to θ^π is equivalent to the expected gradient of $Q\left(s, a\middle|\theta^Q\right)$ to θ^π, and its expected value is generally estimated without bias, according to Monte Carlo methods using the mini-batch training set.

$$\nabla_{\theta^\pi} L(\theta^\pi) = \frac{\partial L(\theta^\pi)}{\partial \theta^\pi} = E\left(\frac{\partial Q(s, a\middle|\theta^Q)}{\theta^\pi}\right) \tag{21}$$

Since $a = \pi(s|\theta^\pi)$, substituting it into the above formula, we can obtain

$$\left[S'_{t.node}, r'_{t+1}, S'_{t+1,node}\right] = \left[S_{t.node}, r_{t+1}, S_{t+1,node}\right] \tag{22}$$

8. Update the target networks $NN(\theta^{\pi-})$ and $NN\left(\theta^{Q-}\right)$ with a soft strategy at regular intervals, as shown in formula (23).

$$\begin{array}{l} \theta^{\pi^-} \leftarrow (1-\tau)\theta^{\pi^-} + \tau\theta^\pi \\ \theta^{Q^-} \leftarrow (1-\tau)\theta^{Q^-} + \tau\theta^Q \end{array} \tag{23}$$

τ is the learning rate, and $0 < \tau < 1$.
The pseudo-code of the network learning algorithm is as Algorithm 3 shows.

Algorithm 3: Network training algorithm based on DDPG

1: Set the target error of the networks training as $E_{Q,Tar}$.
2: Sample N transitions from D based on replay strategy.
3: If $E_Q > E_{Q,Tar}$
4: Set $y' = \begin{cases} 0 & S_{t+1} \in terminal \quad status \\ Q\left(S_{t+1}, a^-_{t+1}\middle|\theta^{Q-}\right) & S_{t+1} \notin terminal \quad status \end{cases}$
5: Set $y_t = r_t + \gamma y\prime$ as the target value of the Critic network $NN\left(\theta^Q\right)$
6: Update network $NN\left(\theta^Q\right)$ by minimizing loss function $L\left(\theta^Q\right) = E\left(\left(y_t - Q\left(s_t, a_t\middle|\theta^Q\right)\right)^2\right)$.
7: Update network $NN(\theta^\pi)$ by error gradient
$\nabla_{\theta^\pi} J(\theta^\pi) = \frac{\partial L(\theta^\pi)}{\partial \theta^\pi} = E_S\left(\frac{\partial Q\left(s_t, a_t\middle|\theta^Q\right)}{\partial \theta^\pi}\right) = E_S\left(\frac{\partial Q\left(s_t, a_t\middle|\theta^Q\right)}{\partial a_t} \cdot \frac{\partial \pi(s_t|\theta^\pi)}{\partial \theta^\pi}\right)$
8: Calculate the current training error of the critic network $NN\left(\theta^Q\right)$. $E_Q = \sum_n \frac{1}{n}\left(\left(y_t - Q\left(s_t, a_t\middle|\theta^Q\right)\right)^2\right)$
9: End If

3.3. Improved Reward Function Design

Designing a perfect and accurate reward function based on professional knowledge is a key step to obtaining high-performance reinforcement learning algorithms, and has a great impact on improving algorithm efficiency and obtaining excellent strategies. However, air combat decision-making is a complex nonlinear problem that is constrained by many factors, such as aircraft performance, weapon attack conditions, environment, etc. Since the action mechanism of many factors is not conclusive, professional knowledge is adopted to a limited extent to improve the reward function in this study.

The air combat situation is divided into three undisputed situations: attack conditions met, attack conditions not met and not attacked, and attacked. The impacts of flight control and environment are also reflected in the form of reward functions. The decision action is not feasible and exceeds the limited range. The reward functions for five scenarios are as follows.

1. When the attack conditions are met, the agent should receive a larger positive reward, and according to the energy air combat theory, the faster the UAV is retained, the better, so the reward function is defined as follows:

$$r_t = 100 \cdot \left(\frac{v_1 + 50}{v_2 + 50} \right) \tag{24}$$

   where v_1 and v_2 are, respectively, the speed of the UAV and the target, in m/s.
2. When the attack conditions are not met and not attacked, it means that the action does not achieve an effective result and should receive a negative reward value. The reward function is defined as follows:

$$r_t = -1 \tag{25}$$

3. When the attack conditions are not met and are attacked, the UAV should receive a larger negative reward (punishment). The faster the UAV is retained, the stronger the ability to avoid missile attack, and the negative reward should be reduced accordingly. The greater the retention speed of the target machine, the more difficult it is to avoid, so the negative reward increases accordingly. Therefore, the reward function is defined as follows:

$$r_t = -50 \cdot \left(\frac{v_2 + 50}{v_1 + 50} \right) \tag{26}$$

4. If the decision result exceeds the current capability of the UAV, the decision is wrong and a larger negative reward should be given:

$$r_t = -20 \tag{27}$$

5. If the current state exceeds the limit, a negative reward value should be given to avoid this state:

$$r_t = -10 \tag{28}$$

4. Simulation and Verification

Due to the limitations of computing, the simulation difficulty is reduced in this paper. Both the target and the UAV are restricted to maneuvering in a plane, without vertical maneuvering, and the flight altitude is 2000 m. At the same time, assuming that the speed of the target does not change and presents circular motion, the target speed is 220 m/s, and the initial speed of the UAV is also 220 m/s.

Considering the complexity of missile attack zone and target radiation characteristics, the missile launch zone is simplified. Set the cone apex angle of D to 160° and D′ to 100°. Combined with Figure 2, the emission condition 1 can be expressed as follows: the angle between the target line U_1U_2 and the velocity vector V_2 is greater than 120°. Emission condition 2 can be expressed as follows: the angle between the target line U_1U_2 and the

velocity vector V_1 is less than or equal to 50°. Set the values of $d_{\min}$ and $d_{\max}$ to be fixed and unchanged at 500 m and 6 km, respectively.

4.1. Validity Verification of the Node Clustering Algorithm

To assess the effectiveness of the node clustering algorithm, the three variable values in state S_t are used as coordinates to generate a point in three-dimensional space. This method allows for a visual representation of the distribution of transitions within the transition database. For ease of comparison, the visual distributions of the transition database before and after clustering are provided. Additionally, the performance is compared with the vector distance method presented in [23].

Firstly, $[d, v_1, v_2]$ are selected as coordinate values for three-dimensional presentation. TS is the temporary transition database generated by the algorithm after 500 rounds of air battle, and the number of transitions is 40,189. The transition database obtained after the node clustering of TS is represented by TS1, the total number is reduced to 14,081, and the time cost is 3.502 s. The transition base obtained after the vector distance similarity exclusion for TS is represented by TS2, the total number is reduced to 12,744, and the time cost is 40.716 s. The time of the vector distance method is much longer than the method in this article. This is because the vector distance between each two pairs needs to be calculated to eliminate similar transitions, and the calculation amount is large. However, in the method in this paper, for each transition, the Euclidean distance from the corresponding node only needs to be calculated once, and then the distances are compared to exclude similar transitions, resulting in a significant reduction in calculation amount.

After normalizing the $[d, v_1, v_2]$ of each transition according to a uniform law, the three-dimensional distributions of TS, TS1, and TS2 are shown in (a), (b), and (c) in Figure 5, respectively.

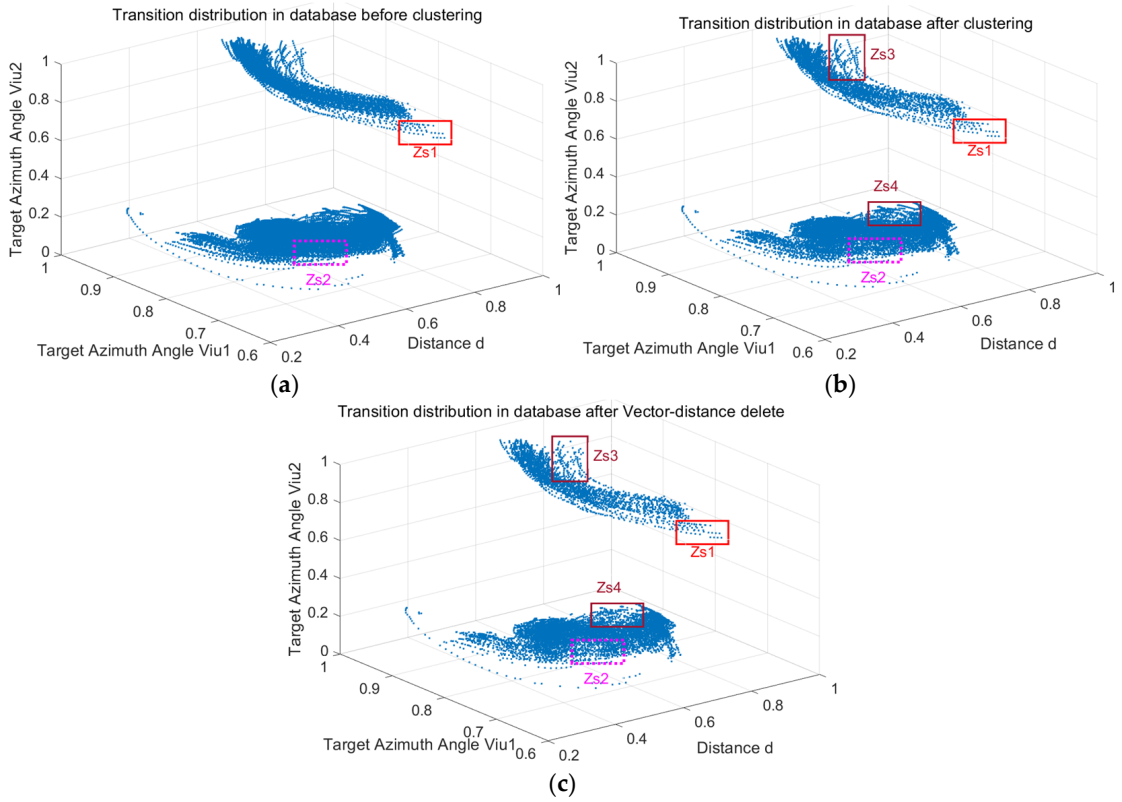

Figure 5. Using $[d, v_1, v_2]$ as coordinates, the transition distribution before clustering, after clustering, and after vector distance exclusion are visually displayed. (**a**) Visual display of the distribution of transitions before clustering; (**b**) visual display of the distribution of transitions after node clustering; (**c**) visual display of the transition distribution after vector distance exclusion.

In Figure 5, each point represents a transition, and Figure 5a is a visual representation of the transition distribution of the database before clustering; Figure 5b shows the visual display of the transition distribution of the database after node clustering in this paper, and Figure 5c shows the visual display of the transition distribution after similar exclusion using the vector distance method. From the visual comparison of transition distribution, it can be seen that after node clustering or vector distance similarity exclusion, the transition in transition-intensive regions is obviously reduced, such as in the Zs2 region, and the color of the same region in Figure 5b,c is obviously lighter, indicating that the density is reduced. At the same time, the number of transitions in transition-sparse regions is not significantly reduced, such as in the Zs1 region, indicating that the algorithm retains the transition of sparse transition regions. This proves that both methods are effective in excluding similar transitions.

By comparing the Zs3 and Zs4 regions in Figure 5b,c, it can be seen that the graph in Figure 5b is closer to the distribution shape of the original database, and the distribution is more uniform, which has a better retention effect on sparse samples. Therefore, it can be considered that the node clustering method in this paper can better maintain the diversity of transition, is beneficial for subsequent network learning, and helps to improve the robustness of the network trained.

In order to confirm the universality of the conclusions, $[v_1, v_2, \alpha]$ is selected as the coordinate value for three-dimensional presentation, and after normalizing the $[v_1, v_2, \alpha]$ of each transition according to a unified law, the three-dimensional distributions of TS, TS1, and TS2 are shown in (a), (b), and (c) in Figure 6, respectively.

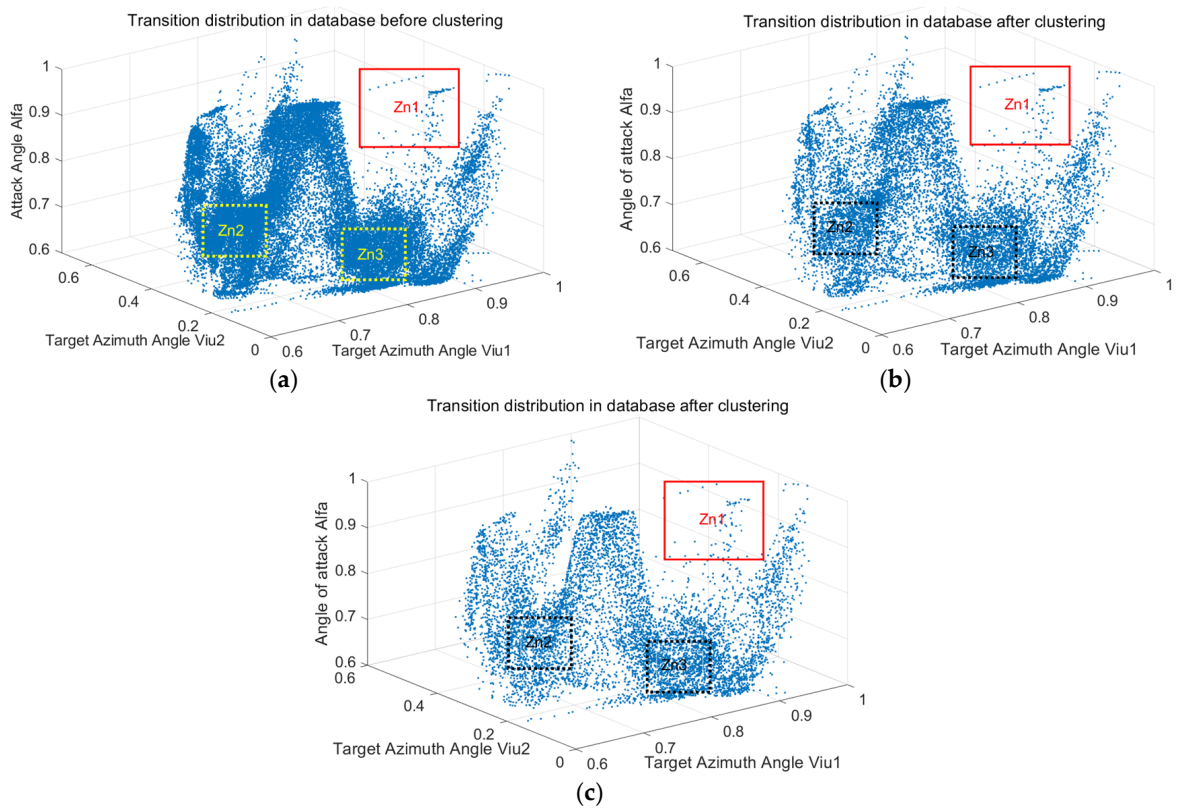

Figure 6. Using $[v_1, v_2, \alpha]$ as coordinates, the transition distribution before clustering, after clustering, and after vector distance exclusion are visually displayed. (**a**) Visual display of the distribution of transitions before clustering; (**b**) visual display of the distribution of transitions after node clustering; (**c**) visual display of the transitions' distribution after vector distance exclusion.

In Figure 6, using $[v_1, v_2, \alpha]$ as coordinates, the transition distributions before clustering, after clustering, and after vector distance exclusion are visually displayed.

It can be seen that, on the whole, Figure 6b is closer to the transition distribution of the original data, and can be considered to better retain the distribution characteristics of the original database. Comparing the Zn1 region of the three figures, it can be seen that Figure 6b is almost identical to Figure 6a, while Figure 6c is much sparser compared with Figure 6b. Obviously, some transitions are excluded from the region in Figure 6c, so the proposed method in this paper has better sparse transition retention ability and can better maintain the diversity of the transition database.

4.2. Verification of Algorithm Validity

In order to verify the autonomous learning and autonomous decision-making ability of the algorithm, different initial conditions are set for verification, including the advantageous situation, the disadvantageous situation, and the equilibrium situation.

4.2.1. Verification of Algorithm Validity Under the Advantageous Situation

In the $OXYZ$ coordinate system, the heading angle of UAV1 (replaced by U1 behind) is 60°, the heading angle of UAV2 (replaced by U2 behind) is −20°, the initial coordinate of U1 is (0, 2000, 0), the initial coordinate of U2 is (6000, 2000, 0), and the initial distance between the two drones is 6 km. At this time, the azimuth angle of U2 relative to U1 v_1= 60° and the azimuth of U1 relative to the target U2 v_2= 160°. The maximum number of air combat episodes EP = 5000, the node cluster exclusion frequency $f_c = 100$ episodes, the network learning frequency $f_T = 200$, and the network soft update frequency $f_u = 400$.

In a certain period, N_{Cy} rounds of air combat are conducted by the algorithm, the number of winning rounds is N_{Su}, and the winning ratio is defined as

$$p_{Su} = N_{Su}/N_{Cy} \tag{29}$$

Similarly, if the number of failed rounds is N_{Fl} and the number of rounds in other situations (such as out of control, out of boundary, etc.) is N_{Oth}, the failure ratio and other case ratio can be defined as

$$p_{Fl} = N_{Fl}/N_{Cy} \tag{30}$$

$$p_{Oth} = N_{Oth}/N_{Cy} \tag{31}$$

In the process of 5000 rounds of exploration and learning, the algorithm takes about 37 min, and the change curves of p_{Su}, p_{Fl}, and p_{Oth} are shown in Figure 7.

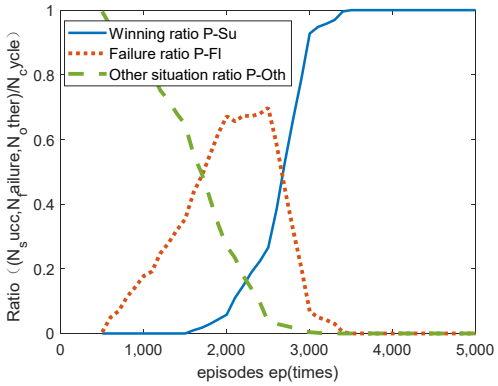

Figure 7. The change curves of p_{Su}, p_{Fl}, and p_{Oth} in the advantageous situation.

It can be seen from the figure that the occurrence of other situations was first avoided through exploration in the early stage, but because the correct strategy was not found, the probability of failure increased, and as the learning process progressed, when the algorithm found the optimal strategy, the success rate p_{Su} gradually increased, and the failure rate p_{Fl} and other ratios p_{Oth} decreased rapidly.

In the learning process, the algorithm carries out continuous strategy exploration, and the typical emerging processes are shown in Figure 8. From the changes in the maneuvering process, it can be seen that, on the whole, U1 carried out more exploration attempts in the early stage of the algorithm operation, found more typical processes, and gradually found an excellent winning strategy in the later stage.

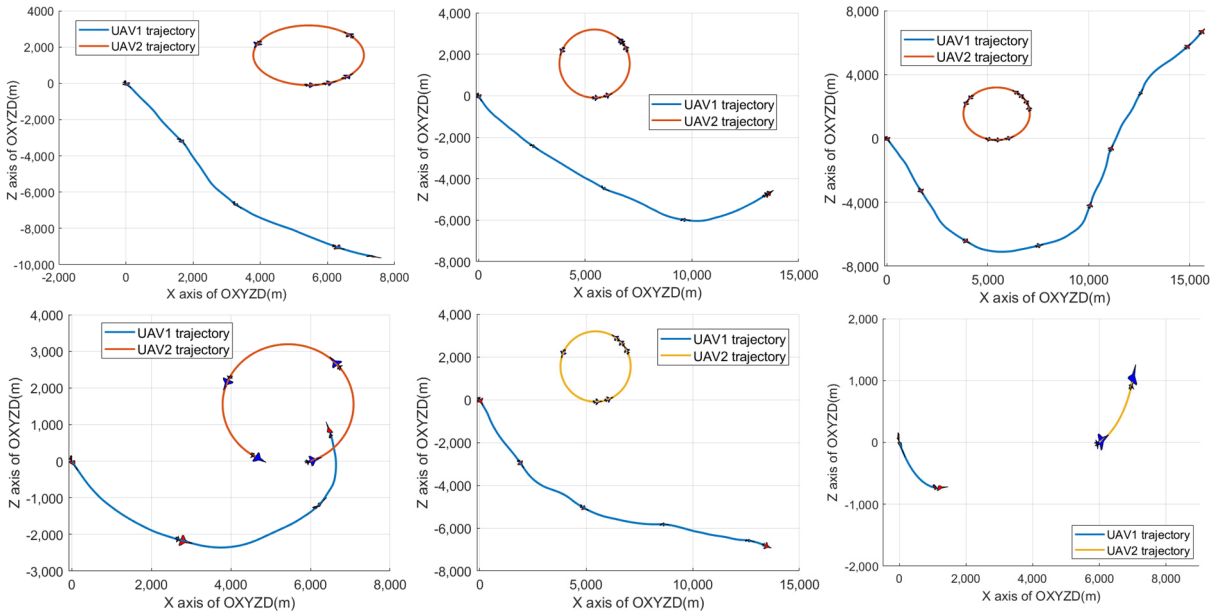

Figure 8. Some typical processes emerged with algorithm in the advantageous situation.

Based on the trained strategy network $NN(\theta^{\pi})$, a maneuvering decision under the same initial conditions is made. The maneuvering processes of U1 and U2 are shown in Figure 9. It can be seen that U1 achieved missile attack conditions against U2 through left-turn maneuvering with a large overload.

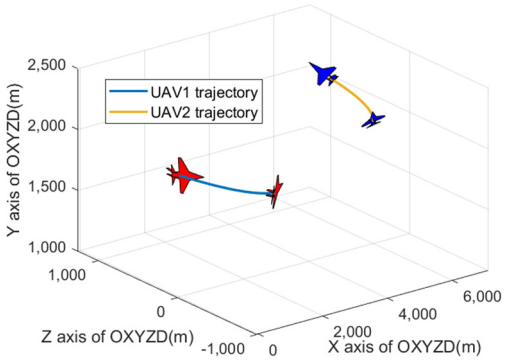

Figure 9. The maneuvering decision process based on the trained strategy network in the advantage situation.

Figure 10a to Figure 10c, respectively, show the change curves of engine thrust, angle of attack, and speed roll angle determined by $NN(\theta^{\pi})$ in the maneuvering process of U1, and Figure 10d shows the change curves of speed during the maneuvering process.

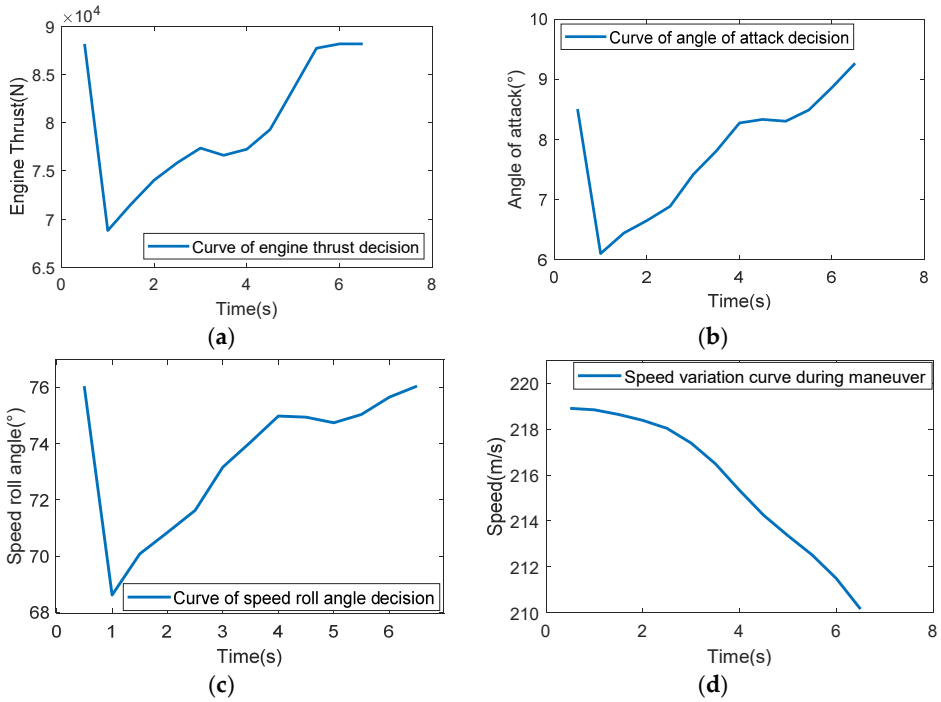

Figure 10. Results of maneuvering command decision-making based on $NN(\theta^{\pi})$ for U1 in the advantageous situation. (**a**) Change curve of engine thrust decision; (**b**) change curve of angle of attack decision; (**c**) change curve of speed roll angle decision; (**d**) change curve of speed during maneuvering.

It can be seen from the simulation results that, based on $NN(\theta^{\pi})$, U1 presented correct decision-making in relation to its actions, which not only ensures the flight control of the UAV, but also meets the attack conditions of its own missiles.

It can be observed that in order to meet the launch conditions, U1 continuously performs high-overload turns to quickly bring the opponent into the missile's attack angle range. Accompanying these high-overload turns, U1's speed continuously decreases. To compensate for the loss of speed, the engine thrust is continuously increased. These changes in parameters all meet the basic requirements of high-overload maneuvering.

4.2.2. Analysis of the Influence of Clustering Scale on Algorithm Performance

In order to analyze the characteristics of the algorithm when the clustering scale is different, different clustering scales are set, respectively, and the simulation analysis of self-learning is carried out under the same condition.

Firstly, multiple clustering scales are set by adjusting the number of variable nodes, with the number of nodes for each variable set as shown in Table 1. It should be noted that due to the limited capabilities of the computing equipment, this simulation is limited to a horizontal plane at an altitude of 2000 m, and it is assumed that the target performs a uniform turn. At this time, $\mu, v_2, \theta_1, \theta_2, H_1, H_2$ are constant values, so these variables do not require clustering nodes to be set.

Table 1. The number of nodes for each variable set for different clustering scales.

	d	v_1	v_2	v_1	F_T	α	γ^v
Scale 1	40	50	50	40	40	40	100
Scale 2	20	50	100	40	40	40	50
Scale 3	20	50	50	20	20	20	50
Scale 4	20	25	25	20	20	20	25
Scale 5	20	20	25	20	20	20	20

As can be seen from Table 1, under scale 1, the number of nodes set for each variable is the largest, the nodes are the most dense, and the clustering scale is the smallest; under scale 5, the number of nodes set for each variable is the smallest, the nodes are the most sparse, and the clustering scale is the largest. In terms of clustering scales, the following relationship exists:

$$\text{Scale 1} < \text{Scale 2} < \text{Scale 3} < \text{Scale 4} < \text{Scale 5}$$

Simulation tests are conducted based on different clustering scales, with other conditions remaining unchanged. Each scale underwent four rounds of simulation. The curves showing the change in winning rate across different rounds of simulation are illustrated in Figure 11a–e. Table 2 records the evolutionary generation when the winning rate p_{succ} reaches 100% in each simulation, denoted as n_{p_succ}, as well as the time consumed by the algorithm to complete all 5000 rounds of simulation, denoted as T_{Cost}.

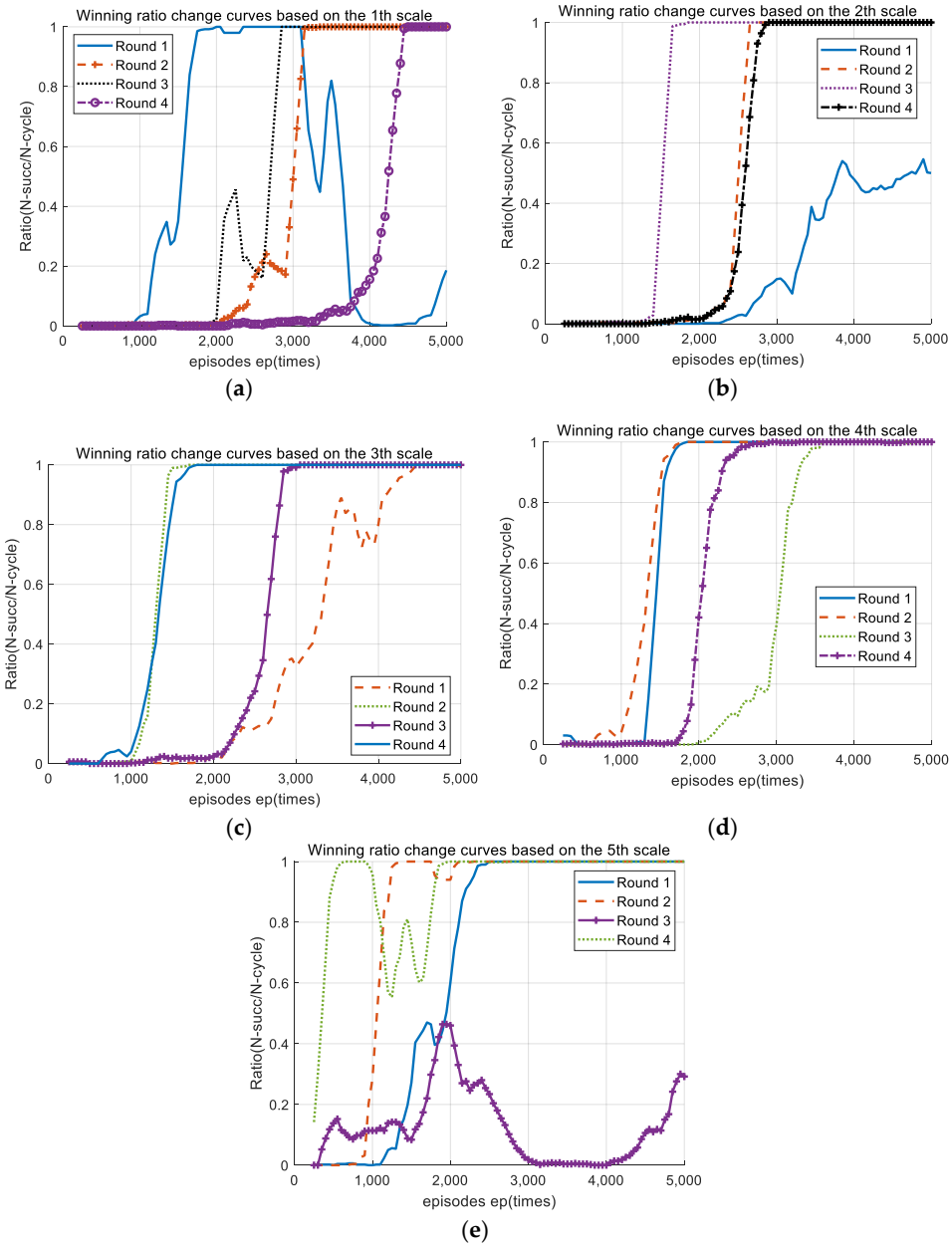

Figure 11. Based on different clustering scales, the changing curves of p_{succ} in the advantageous situation are shown.

Table 2. p_{succ} = 100% algebra and time T_{Cost} to complete EP = 5000 full learning.

		Round1	Round2	Round3	Round4	Average1	Average2	Number of Non-Convergence Rounds
Scale1	n_{p_succ}	--	2850	3350	4500	5175	3566.67	1
	$T_{Cost}(h)$	1.39	0.83	0.98	1.22	1.11	1.01	
Scale2	n_{p_succ}	--	2750	1850	2900	4375	2500	1
	$T_{Cost}(h)$	1.47	0.87	0.73	0.94	1.19	0.82	
Scale3	n_{p_succ}	4550	1700	2900	1700	2712.5	2712.5	0
	$T_{Cost}(h)$	1.09	0.64	0.86	0.72	0.83	0.83	
Scale4	n_{p_succ}	1850	1800	3650	2950	2562.5	2562.5	0
	$T_{Cost}(h)$	0.70	0.71	1.13	0.84	0.85	0.85	
Scale5	n_{p_succ}	2500	1400	--	650	3637.5	1516.67	1
	$T_{Cost}(h)$	0.83	0.67	1.02	0.97	0.85	0.79	

"--" indicates that the algorithm is not converged, and the mean value of T_{Cost} is calculated according to "T_{Cost} = 10,000".

In Figure 11, (a) shows the simulation results at scale 1, (b) shows the simulation results at scale 2, and so on. From the simulation results, it can be observed that at scales 1, 2, and 5, there was one round of simulation that did not converge on three occasions, accounting for 15% of the total 20 rounds of simulations. The simulations based on scales 3 and 4 were all successful. Judging from the simulation results, the algorithm exhibits significant randomness. Even the learning processes conducted at the same scale vary greatly. For ease of observation, the average values of the four rounds of simulation results at each scale are compared, and the statistical results are shown as "Average1" in Table 2. At the same time, for a more intuitive comparison, histograms of the average values of n_{p_succ} and T_{Cost} at each scale are provided in Figure 12a. If the rounds that did not converge are excluded and the statistics are re-calculated, different results are achieved, shown as "Average2" in Table 2 and on the histogram in Figure 12b.

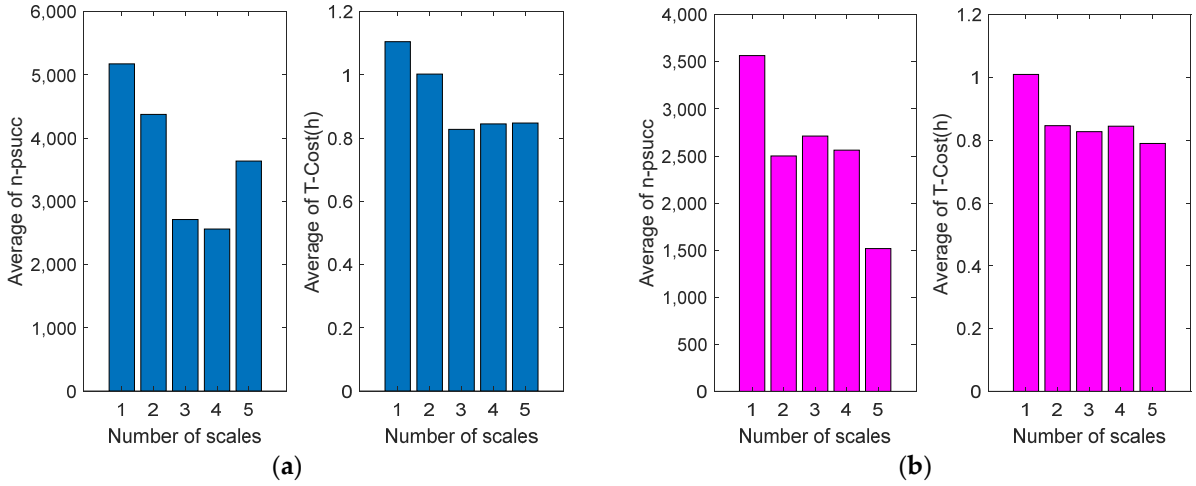

Figure 12. Comparison of histograms of the mean values of n_{p_succ} and T_{Cost} at various scales.

It can be conveniently observed that although the algorithm exhibits a certain degree of randomness, different clustering scales have a significant impact on the algorithm. Basically, under scale 1, the learning process is more time-consuming and takes longer, with the mean values of n_{p_succ} and T_{Cost} being the largest. Scale 2 follows closely. If we consider that

there is an unsuccessful learning process under both scale 1 and scale 2, it can be inferred that when the clustering scale becomes smaller and the number of nodes is larger, this will increase the learning difficulty of the algorithm. However, larger is not necessarily better for the learning scale. Scale 5 has the largest clustering scale and the least number of nodes, but it does not have an advantage in terms of time consumption and even has an unsuccessful learning process. The learning processes under scales 3 and 4 are the most stable.

Overall, when there are a large number of variable clustering nodes, the description of states and actions by the clustering nodes is more refined, but this obviously puts higher demands on the learning algorithm, which will also lead to a longer learning process and more time consumption. However, the clustering scale should not be as large as possible. When there are fewer nodes, this may lead to overly coarse descriptions of states and actions by the nodes, resulting in oscillations in the learning process, which is also not conducive to algorithm convergence. Therefore, when applying this method, the node scale should be determined comprehensively based on the computing power of the equipment, accuracy requirements, and algorithm convergence.

4.2.3. Comparative Analysis with TD3 Algorithm

The TD3 (Twin Delayed DDPG) algorithm is a novel deep reinforcement learning algorithm that primarily integrates the concept of Double Q-Learning into the DDPG algorithm. By incorporating a set of critic networks, it addresses the issue of overestimating values. Additionally, it introduces noise into the exploration process, enhancing the exploration process and effectively avoiding local optima. Based on the standard TD3 algorithm, a maneuver decision-making autonomous learning algorithm is constructed. Furthermore, by integrating the NC method proposed in this paper with the TD3 method, an NC-TD3-based maneuver decision-making autonomous learning algorithm is developed. Simulation validations are conducted over multiple rounds based on TD3 and NC-TD3, respectively, and the node clustering scale selected is scale 3 from Table 1.

In the verification based on the standard TD3 algorithm, the curve of winning rate p_{succ} changes during the process of Ep = 10,000, as shown in Figure 13. It can be seen that the self-learning of the maneuvering strategy can also be achieved based on the standard TD3 algorithm.

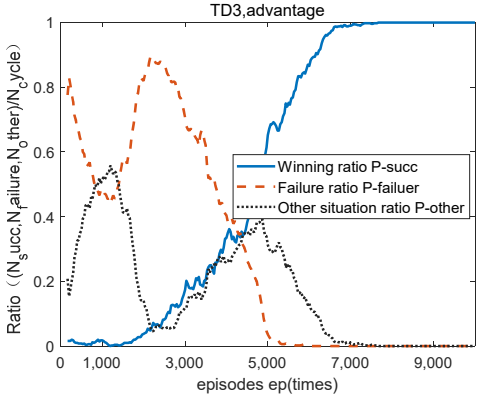

Figure 13. The change curves of p_{Su}, p_{Fl}, and p_{Oth} based on TD3 in the advantageous situation.

However, due to the lack of similarity elimination in the transitions library of the standard TD3 algorithm, the number of transitions in D1 rapidly increases, which also leads to a rapid increase in the amount of transitions, as well as replay and network learning computation, resulting in an abnormally slow completion of the training process. Table 3 presents a comparison of the learning process results between the TD3-based algorithm and the NC-DDPG-based algorithm, focusing primarily on the comparison of the

number of transitions in D1, algorithm running time (Time Cost), and the moment when p_{succ} = 100%.

Table 3. Comparison of the standard TD3 algorithm with typical NC-DDPG simulation results.

	Number of Episodes	Number of Transitions in D1	Time Cost (*h*)	p_{succ} = 100% Algebra
TD3	10,000	2,147,962	147.53	7450
NC-DDPG	5000	164,766	0.86	2900

In terms of the most important time spent, it can be seen that the standard TD3 algorithm took 147.53 h to complete a round of training without similar exclusion, while the typical learning process of NC-DDPG under the same conditions took only 0.86 h to complete, which shows that it is necessary to use a certain method to exclude similar transitions. Additionally, considering that the time consumption of the standard TD3 algorithm significantly exceeds that of the algorithm presented in this paper, further experiments will not be conducted.

Based on NC-TD3, multiple rounds of simulations were conducted with Ep set to 10,000. The curves depicting the change in winning rate p_{succ} across eight different rounds of simulations are shown in Figure 14. The main simulation results are presented in Table 4. Here, p_{su_final} represents the final value of p_{succ}, T_{Cost} represents the time consumed for 10,000 episodes of learning, and N_{D1} represents the number of transitions in D1.

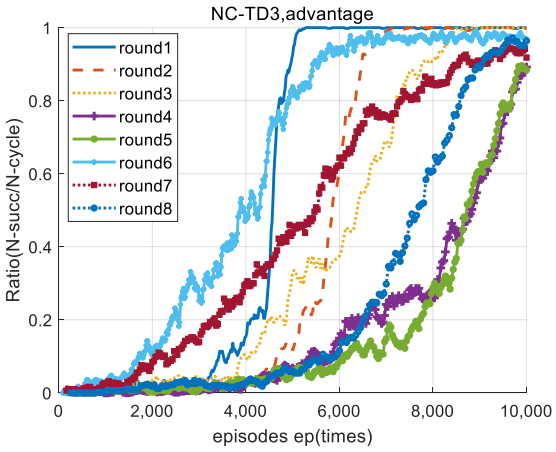

Figure 14. The change curves of p_{Succ} over 8 rounds based on NC-TD3 in the advantageous situation.

Table 4. The main simulation results of NC-TD3.

	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8
p_{su_final}	100%	100%	100%	89.2%	88.4%	96.8%	91.8%	96.4%
$T_{\mathrm{Cost}}(h)$	2.19	3.43	1.88	6.47	4.32	4.19	2.76	3.81
N_{D1}	367,211	439,802	243,338	891,622	499,679	450,977	315,151	317,115

From the simulation results, it can be observed that the learning process based on the NC-TD3 algorithm demonstrates good algorithmic effectiveness. Compared to the NC-DDPG algorithm, it exhibits distinct characteristics, including the following:

(1) The algorithm operation process is relatively stable, and the final results of p_{Succ} are above 88% in all eight rounds of simulation.

(2) There is a notable difference between the p_{Succ} rising curve of NC-TD3 and that of NC-DDPG. NC-TD3's p_{Succ} tends to show a steady increase, with fewer occurrences of "step-like" rises, while NC-DDPG often exhibits "step-like" rising characteristics. This indicates the stability of the NC-TD3 algorithm.

(3) The algorithm does not have any advantage in convergence speed; it takes a longer time and requires more computational resources compared to NC-DDPG.

4.2.4. Verification of Algorithm Effectiveness Under Equilibrium Situation

In the $OXYZ$ coordinate system, the heading angle of U1 is set to 90° and the heading angle of U2 is −90°, which can be considered an equilibrium situation in short-range air combat. The change curves of p_{Su}, p_{Fl}, and p_{Oth} during the exploration and learning process of the algorithm are shown in Figure 15.

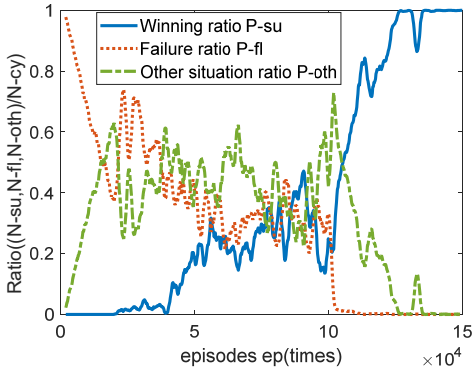

Figure 15. Change curves of p_{Su}, p_{Fl}, and p_{Oth} in the equilibrium situation.

After 150,000 rounds, the algorithm learned the correct maneuvering strategy, and the winning probability of the algorithm increased to 100%. The time cost was about 131 h. It can be seen that the autonomous exploration and learning of the air combat strategy in the equilibrium situation was a relatively long process, and the computational time and amount of data consumed by the algorithm far exceed the process in the advantageous situation, which reflects the complexity of the maneuvering strategy in the equilibrium situation.

As can be seen intuitively from Figure 16, U1 first performed a right turn to avoid the opponent's missile attack area. Then, it turned around on a large slope after the target was about to expose its tail, making the target enter the launchable range of its own missile while ensuring that it did not lose control, achieving the launch conditions for its own missile. This reflects the autonomous learning ability of the algorithm for complex strategies under conditions of multiple limitations.

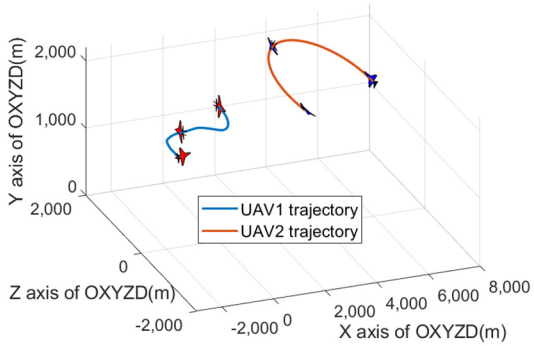

Figure 16. The maneuvering decision process based on the trained strategy network in the equilibrium situation.

Figure 17a–c show the change curves of attack angle, engine thrust, and speed roll angle decided by $NN(\theta^{\pi})$ in the maneuvering process of U1, and Figure 17d shows the change curve of speed in the process of U1 maneuvering. It can be observed that in the equilibrium situation, U1 completed a complex maneuvering process. To achieve this process, the change in the angle of attack can be divided into three stages: high angle of

attack with right turn, low angle of attack with level flight, and high angle of attack with left turn. Continuous maneuvering results in an overall decrease in speed, but during the intermediate low angle of attack with a level flight stage, the speed is slightly increased. The engine thrust is always operated at high thrust to compensate for speed loss. The parameter decision-making results are basically in line with reality, indicating that the algorithm is effective.

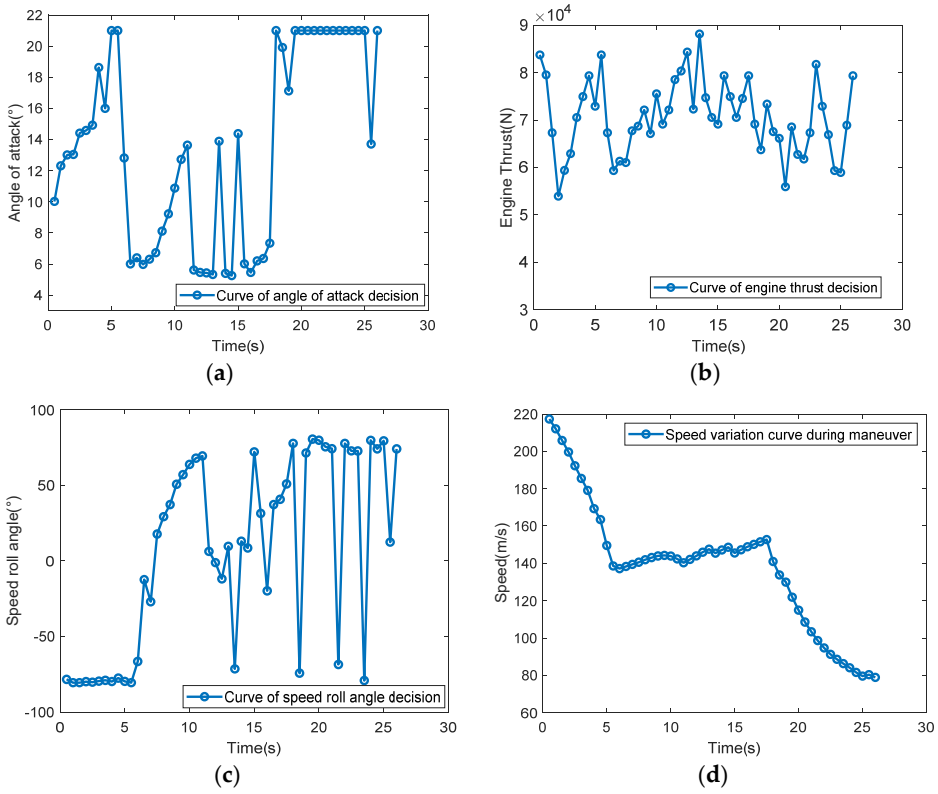

Figure 17. Results of maneuvering command decision based on $NN(\theta^{\pi})$ for U1 in the equilibrium situation. (**a**) Change curve of angle of attack decision; (**b**) change curve of engine thrust decision; (**c**) change curve of speed roll angle decision; (**d**) change curve of speed during maneuvering.

4.2.5. Verification of Algorithm Effectiveness in Disadvantageous Situation

In the $OXYZ$ coordinate system, the heading angle of U1 is set to 90° and the heading angle of U2 is −130°, which can be considered a disadvantageous situation in short-range air combat. During the exploration and learning process of the algorithm, the change curves of p_{Su}, p_{Fl}, and p_{Oth} are shown in Figure 18.

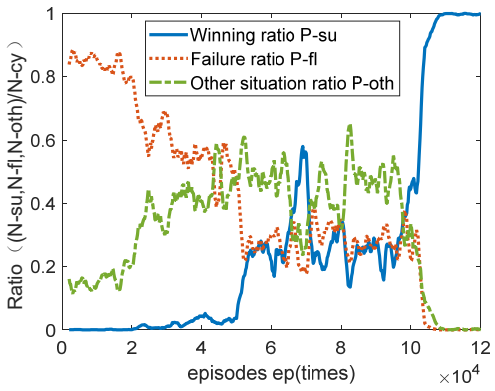

Figure 18. The change curves of p_{Su}, p_{Fl}, and p_{Oth} in the disadvantageous situation.

After about 110,000 rounds, the algorithm learned the correct maneuvering strategy, and the winning probability of the algorithm increased to 100%. The time spent on 120,000 rounds was approximately 116 h. It can be seen that the exploration and learning of air combat strategies in the disadvantageous situation was also a relatively long process, which reflects the complexity of maneuvering decision-making in the disadvantageous situation.

However, compared with the original algorithm using vector moments to carry out similar exclusion (referred to as V_DDPG), the proposed algorithm (referred to as NC_DDPG) shows advantages both under the equilibrium condition and under the disadvantageous condition, as shown in Table 5.

Table 5. Comparison of the main indicators of NC_DDPG and V_DDPG in situations of advantage and disadvantage.

		Time Cost	Number of Episodes	The Final Number of Transitions in the Replay Database
equilibrium situation	V_DDPG	≈170 h	260,000	1,479,665
	NC_DDPG	≈131 h	150,000	1,114,721
disadvantageous situation	V_DDPG	≈150 h	220,000	1,293,425
	NC_DDPG	≈116 h	120,000	836,997

The maneuver decision in this situation is made based on the trained $NN(\theta^{\pi})$, and Figure 19 shows a visual display of the maneuver process.

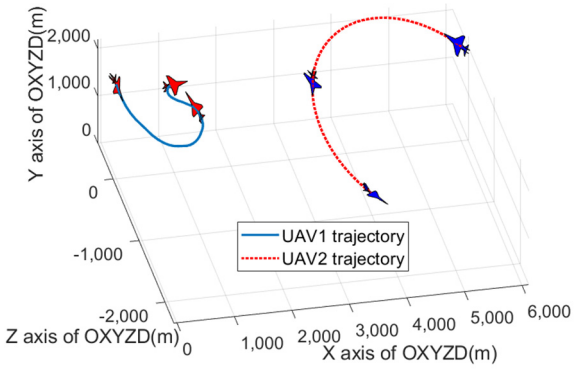

Figure 19. Based on the trained strategy network $NN(\theta^{\pi})$, the maneuvering decision process in the disadvantageous situation is shown.

It can be intuitively seen from the figure that U1 continues to turn left, always keeping itself in the front hemisphere of the target, not creating launch conditions for the opponent, and at the same time, maintaining distance from the target to avoid a distance less than 2000 m (program setting: when the distance is less than 2000 m, the simulation is terminated). When U1 enters the rear hemisphere of the target, it makes a large overload right turn, so that the nose is pointed at the target. Because the two sides are close at this time, so when the target enters the angle range of its own missile, all of the launch conditions of its own missile are constituted. Over the whole process, it is ensured that the UAV does not lose control, and all parameters are within the allowable range, which once again confirms the autonomous learning ability of the algorithm for complex strategies under the condition of multiple constraints.

Figure 20a–c show the change curves of engine thrust, angle of attack, and speed roll angle decided by $NN(\theta^{\pi})$ in the maneuvering process of U1, and Figure 20d shows the change curve of speed during maneuvering. It can be observed that in order to meet the launch conditions, U1 continuously performs high-overload turns with a consistently high angle of attack and overload, resulting in a continuous decrease in speed. To complete the sustained turns, the engine thrust is always operated at a high thrust state. The

decision-making results of these parameters are in line with the actual situation, proving the effectiveness of the decision-making algorithm.

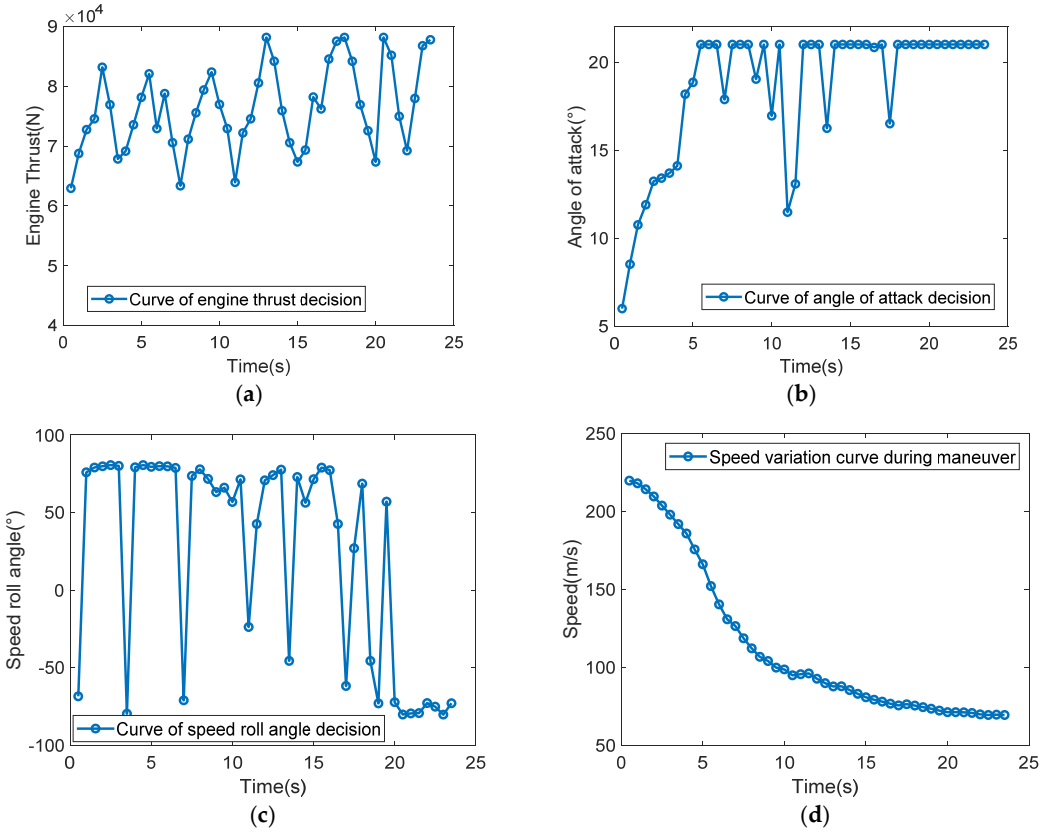

Figure 20. Results of maneuvering command decision based on $NN(\theta^{\pi})$ for U1 in the disadvantageous situation. (**a**) Change curve of engine thrust decision; (**b**) change curve of angle of attack decision; (**c**) change curve of speed roll angle decision; (**d**) change curve of speed during maneuvering.

4.3. Simulation Analysis of the Influence of Altitude on Strategy

In order to observe the influence of altitude on air combat strategy, five altitudes, including 1000 m, 2000 m, 4000 m, 6000 m, and 8000 m, were set, respectively, and the obtained maneuvering strategies were compared. The other initial conditions were the same as in the advantageous situation in Section 4.2.1. The comparison of the strategies obtained for the five conditions is shown in Figure 21, and the change curves of engine thrust, normal overload, angle of attack, speed roll angle, and speed under different strategies are shown in Figure 22a–d, respectively.

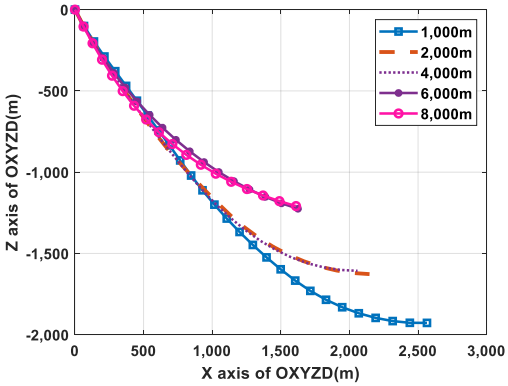

Figure 21. Comparison of maneuvering processes at different altitudes.

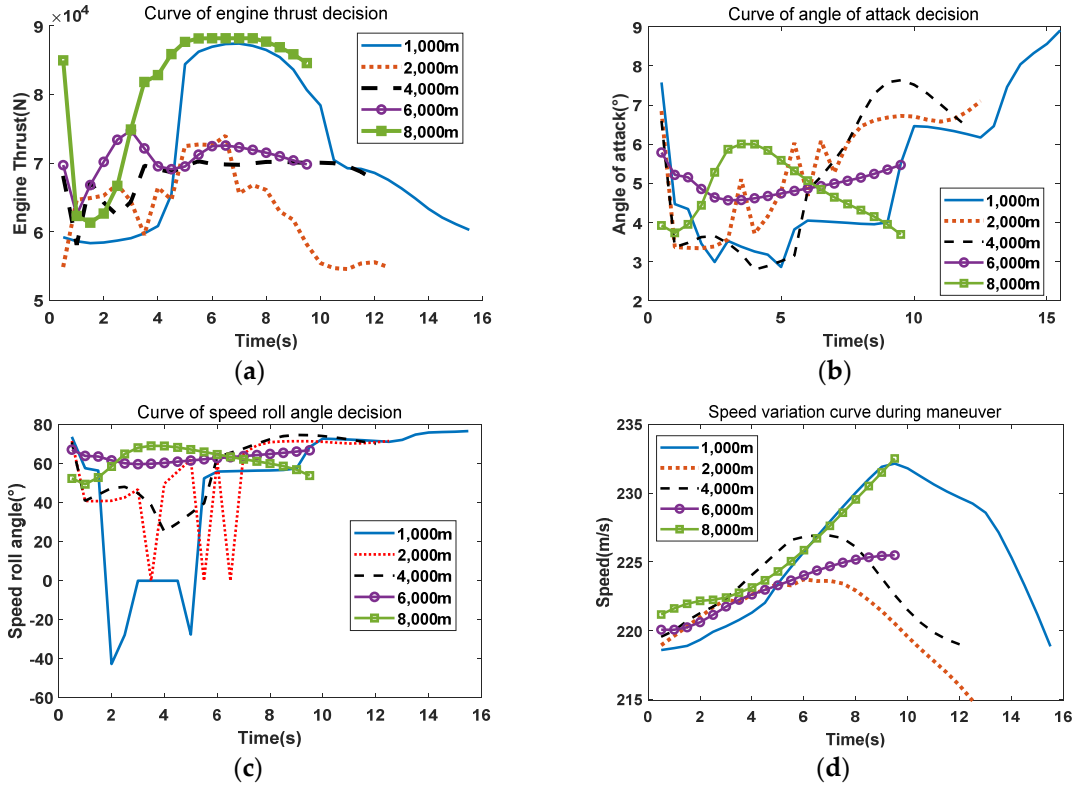

Figure 22. Comparison of the change curves of action commands during maneuvering at different heights. (**a**) Comparison of engine thrust curves; (**b**) comparison of normal overload curves during maneuvering; (**c**) comparison of angle of attack change curves; (**d**) comparison of speed roll angle change curves.

As can be seen from Figure 21, with the change in altitude, the maneuvering strategy of U1 underwent a great change. In Figure 22, during maneuvering at different heights, the change curves of the action commands vary greatly from each other. These differences all prove that the influence of altitude on strategy learning is huge, and it is necessary to introduce altitude parameters into model state representation.

As can be seen from Figures 21 and 22, although maneuvering processes at different altitudes show great differences, they do not show an obvious trend. For example, in Figure 22, the maneuvering processes at 6000 m and 8000 m are approximately coincident, and the maneuvering process at 2000 m and 4000 m approximately coincide. Theoretically, the maneuvering processes at 2000 m, 4000 m, 6000 m, and 8000 m should exhibit a consistent trend. The changes accompanying the different maneuvering commands in Figure 22 also show greater randomness and no obvious trend of change. This phenomenon is mainly due to the randomness in the operation of the algorithm, especially in the initial stage, which leads to a large difference in the air combat transitions obtained. Thus, decision networks with large difference were trained. In addition, there is also a certain randomness in the decision-making process, which leads to the randomness of the decision-making results to a certain extent.

5. Conclusions

To address the difficulty of strategy learning in complex situations with the DDPG algorithm, a nodal clustering method is initially proposed. Utilizing this method, similar transitions are filtered out from the transition pool, retaining only the most representative ones. By integrating node clustering with DDPG, a self-learning and autonomous decision-making algorithm for short-range air combat maneuvering is designed. This includes the algorithm framework and steps, with a pseudocode provided for the main steps.

Additionally, the state representation is enhanced by incorporating flight altitude into the state variable vector.

The effectiveness of both the node clustering method and the air combat maneuvering decision-making method were tested. The results indicate that, compared to the DDPG algorithm integrated with vector distance exclusion, the proposed algorithm requires only 77.06% of V_DDPG's computational load under equilibrium conditions and just 77.33% under disadvantageous conditions. This demonstrates that the proposed method offers higher efficiency. The clustering scale can influence the algorithm's learning process, with a finer clustering scale demanding greater computational resources. Therefore, the clustering scale should be determined by considering various factors comprehensively. The differences in maneuvering strategies at various altitudes were tested, highlighting the importance of enhancing state variables.

Author Contributions: This article was jointly produced by all the authors. X.J. and J.H. identified the problem and suggested the idea. X.J. and F.C. designed the research methodology and F.C. provided valuable advice. X.J., J.H., and F.C. improved the problem model. X.J. and C.T. reviewed the literature. X.J., F.C., and Z.S. designed the algorithm flow. X.J. and C.T. conducted the test experiments. X.J. and C.T. edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available in paper "Autonomous Decision-making Algorithm for UAV Maneuvering Based on Node Clustering and DDPG".

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

References

1. Mostafa, S.A.; Mustapha, A.; Gunasekaran, S.S.; Ahmad, M.S.; Mohammed, M.A.; Parwekar, P.; Kadry, S. An agent architecture for autonomous uav flight control in object classification and recognition missions. *Soft Comput.* **2023**, *27*, 391–404. [CrossRef]
2. Huang, H.; Weng, W.; Zhou, H.; Jiang, Z.; Dong, Y. Maneuvering Decision Making Based on Cloud Modeling Algorithm for UAV Evasion–Pursuit Game. *Aerospace* **2024**, *11*, 190. [CrossRef]
3. Yang, Z.; Zhou, D.; Kong, W.; Piao, H.; Zhang, K.; Zhao, Y. Nondominated maneuver strategy set with tactical requirements for a fighter against missiles in a dogfight. *IEEE Access* **2020**, *8*, 117298–117312. [CrossRef]
4. Xi, A.; Cai, Y. Deep Reinforcement Learning-Based Differential Game Guidance Law against Maneuvering Evaders. *Aerospace* **2024**, *11*, 558. [CrossRef]
5. Virtanen, K.; Karelahti, J.; Raivio, T. Modeling air combat by a moving horizon influence diagram game. *J. Guid. Control Dyn.* **2006**, *29*, 1080–1091. [CrossRef]
6. McGrew, J.S.; How, J.P.; Williams, B.; Roy, N. Air-combat strategy using approximate dynamic programming. *J. Guid. Control Dyn.* **2010**, *33*, 1641–1654. [CrossRef]
7. Zhou, Y.; Ma, Y.; Song, X.; Gong, G. Hierarchical fuzzy art for q-learning and its application in air combat simulation. *Int. J. Model. Simul. Sci. Comput.* **2017**, *8*, 1750052. [CrossRef]
8. Liu, P.; Ma, Y. A deep reinforcement learning based intelligent decision method for ucav air combat. In *Modeling, Design and Simulation of Systems*; Mohamed Ali, M.S., Ed.; Springer: Singapore, 2017; pp. 274–286.
9. Schvaneveldt, R.; Goldsmith, T.; Benson, A.; Waag, W. *Neural Network Models of Air Combat*; Maneuvering New Mexico State University: Las Cruces, NM, USA, 1992.
10. Kaneshige, J.T.; Krishnakumar, K.S. Artificial immune system approach for air combat maneuvering. In Proceedings of the SPIE—The International Society for Optical Engineering, Orlando, FL, USA, 30 April 2007. [CrossRef]
11. Burgin, G.H.; Sidor, L. Rule-Based Air Combat Simulation. 1988. Available online: https://ntrs.nasa.gov/citations/19890018022 (accessed on 10 November 2024).
12. Ernest, N.; Carroll, D.; Schumacher, C.J.; Clark, M.A.; Cohen, K.; Lee, G. Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions. *J. Def. Manag.* **2016**, *6*. [CrossRef]
13. Wu, A.; Yang, R.; Liang, X.; Zhang, J.; Qi, D.; Wang, N. Visual range maneuver decision of unmanned combat aerial vehicle based on fuzzy reasoning. *Int. J. Fuzzy Syst.* **2022**, *24*, 519–536. [CrossRef]
14. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.A. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602. [CrossRef]
15. Botvinick, M.; Wang, J.X.; Dabney, W.; Miller, K.J.; Kurth-Nelson, Z. Deep reinforcement learning and its neuroscientific implications. *Neuron* **2020**, *107*, 603–616. [CrossRef] [PubMed]

16. Agency, D.A.R.P. Alphadogfight Trials Go Virtual for Final Event. Defense Advanced Research Projects Agency: 2020. Available online: https://www.pressreleasepoint.com/alphadogfight-trials-go-virtual-final-event (accessed on 10 November 2024).
17. Theresa, H. DARPA's AlphaDogfight tests AI Pilot's Combat Chops. Available online: https://breakingdefense.com/2020/08/darpas-alphadogfight-tests-ai-pilots-combat-chops/ (accessed on 17 March 2023).
18. Yang, Q.; Zhang, J.; Shi, G.; Hu, J.; Wu, Y. Maneuver decision of uav in short-range air combat based on deep reinforcement learning. *IEEE Access* **2020**, *8*, 363–378. [CrossRef]
19. Li, Y.-F.; Shi, J.-P.; Jiang, W.; Zhang, W.-G.; Lyu, Y.-X. Autonomous maneuver decision-making for a ucav in short-range aerial combat based on an ms-ddqn algorithm. *Def. Technol.* **2022**, *18*, 1697–1714. [CrossRef]
20. Gunning, D.; Vorm, E.; Wang, Y.; Turek, M. DARPA's explainable AI (XAI) program: A retrospective. *Appl. Lett.* **2021**, *2*, e61. [CrossRef]
21. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.M.O.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
22. Yang, Q.; Zhu, Y.; Zhang, J.; Qiao, S.; Liu, J. Uav air combat autonomous maneuver decision based on ddpg algorithm. In Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA), Edinburgh, UK, 16–19 July 2019; pp. 37–42.
23. Jing, X.; Hou, M.; Wu, G.; Ma, Z.; Tao, Z. Research on maneuvering decision algorithm based on improved deep deterministic policy gradient. *IEEE Access* **2022**, *10*, 92426–92445. [CrossRef]
24. Li, B.; Bai, S.; Liang, S.; Ma, R.; Neretin, E.; Huang, J. Manoeuvre decision-making of unmanned aerial vehicles in air combat based on an expert actor-based soft actor critic algorithm. *CAAI Trans. Intell. Technol.* **2023**, *8*, 1608–1619. [CrossRef]
25. Zhang, S.; Li, Y.; Dong, Q. Autonomous navigation of uav in multi-obstacle environments based on a deep reinforcement learning approach. *Appl. Soft Comput.* **2022**, *115*, 108194. [CrossRef]
26. Li, B.; Yang, Z.-P.; Chen, D.-Q.; Liang, S.-Y.; Ma, H. Maneuvering target tracking of uav based on mn-ddpg and transfer learning. *Def. Technol.* **2021**, *17*, 457–466. [CrossRef]
27. Li, B.; Gan, Z.; Chen, D.; Sergey Aleksandrovich, D. Uav maneuvering target tracking in uncertain environments based on deep reinforcement learning and meta-learning. *Remote Sens.* **2020**, *12*, 3789. [CrossRef]
28. Xie, J.; Peng, X.; Wang, H.; Niu, W.; Zheng, X. Uav autonomous tracking and landing based on deep reinforcement learning strategy. *Sensors* **2020**, *20*, 5630. [CrossRef] [PubMed]
29. 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]
30. Mei, J.; Li, G.; Huang, H. Deep reinforcement-learning-based air-combat-maneuver generation framework. *Mathematics* **2024**, *12*, 3020. [CrossRef]
31. Liu, X.; Yin, Y.; Su, Y.; Ming, R. A Multi-UCAV Cooperative Decision-Making Method Based on an MAPPO Algorithm for Beyond-Visual-Range Air Combat. *Aerospace* **2022**, *9*, 563. [CrossRef]
32. Wang, L.; Wang, J.; Liu, H.; Yue, T. Decision-Making Strategies for Close-Range Air Combat Based on Reinforcement Learning with Variable-Scale Actions. *Aerospace* **2023**, *10*, 401. [CrossRef]
33. Qi, G.; Li, Y. Reinforcement learning control for robot arm grasping based on improved ddpg. In Proceedings of the 2021 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021; pp. 4132–4137.
34. Zhang, M.; Zhang, Y.; Gao, Z.; He, X. An improved ddpg and its application based on the double-layer bp neural network. *IEEE Access* **2020**, *8*, 177734–177744. [CrossRef]
35. Xinlei, Y.; Zhipeng, G.; Zijian, X.; Chen, Z.; Yang, Y. Ddpg-adaptconfig: A deep reinforcement learning framework for adaptive device selection and training configuration in heterogeneity federated learning. *Future Gener. Comput. Syst.* **2025**, *163*, 107528.
36. Hu, W.; Zhou, Y.; Ho, H.W.; Zhang, C. Double critics and double actors deep deterministic policy gradient for mobile robot navigation using adaptive parameter space noise and parallel experience replay. *IEEE Access* **2024**, *12*, 173192–173208. [CrossRef]
37. Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 881–892. [CrossRef]
38. Hämäläinen, J.; Kärkkäinen, T.; Rosi, T. Improving scalable k-means++. *Algorithms* **2021**, *14*, 6. [CrossRef]
39. Du, M.; Wu, F. Grid-based clustering using boundary detection. *Entropy* **2022**, *24*, 1606. [CrossRef] [PubMed]
40. Gan, G.; Ma, C.; Wu, J. *Data Clustering: Theory, Algorithms, and Applications*, 2nd ed.; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 2020.