




Article

# UAV Air Game Maneuver Decision-Making Using Dueling Double Deep Q Network with Expert Experience Storage Mechanism

Jiahui Zhang , Zhijun Meng , Jiazheng He, Zichen Wang and Lulu Liu 

School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China; zhangjiahui@buaa.edu.cn (J.Z.); h19374301@buaa.edu.cn (J.H.); wangzichen@buaa.edu.cn (Z.W.); liu\_lulu@buaa.edu.cn (L.L.)

\* Correspondence: mengzhijun@buaa.edu.cn

**Abstract:** Deep reinforcement learning technology applied to three-dimensional Unmanned Aerial Vehicle (UAV) air game maneuver decision-making often results in low utilization efficiency of training data and algorithm convergence difficulties. To address these issues, this study proposes an expert experience storage mechanism that improves the algorithm's performance with less experience replay time. Based on this mechanism, a maneuver decision algorithm using the Dueling Double Deep Q Network is introduced. Simulation experiments demonstrate that the proposed mechanism significantly enhances the algorithm's performance by reducing the experience by 81.3% compared to the prioritized experience replay mechanism, enabling the UAV agent to achieve a higher maximum average reward value. The experimental results suggest that the proposed expert experience storage mechanism improves the algorithm's performance with less experience replay time. Additionally, the proposed maneuver decision algorithm identifies the optimal policy for attacking target UAVs using different fixed strategies.

**Keywords:** UAV; maneuver decision-making; deep reinforcement learning; air game



**Citation:** Zhang, J.; Meng, Z.; He, J.; Wang, Z.; Liu, L. UAV Air Game Maneuver Decision-Making Using Dueling Double Deep Q Network with Expert Experience Storage Mechanism. *Drones* **2023**, *7*, 385. <https://doi.org/10.3390/drones7060385>

Academic Editor: Diego González-Aguilera

Received: 27 April 2023

Revised: 3 June 2023

Accepted: 5 June 2023

Published: 8 June 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The advancement of Unmanned Aerial Vehicle (UAV) technology has led to its emergence as a novel weapon system in air games. In comparison to manned aircraft, UAVs offer benefits such as no risk of human casualties, extended flight duration, and low cost [1]. Nevertheless, UAVs do not possess the same degree of flexibility and adaptability as manned aircraft and suffer from limited autonomous gaming capabilities. Consequently, enhancing the autonomous decision-making capacity of UAVs is a primary area of interest in contemporary UAV air game research. UAV air game maneuver decision-making algorithms utilize sensor data as input to generate the optimal strategy for successfully executing target UAV strike missions. Broadly, maneuver decision algorithms employed by UAVs can be classified into three categories: game theory methods, optimization methods and artificial intelligence methods [2].

Game theory is a discipline that studies mathematical models of conflict and cooperation between intelligent decision makers [3]. Given the inherent game-like nature of interactions between two UAVs in air game scenarios, employing game theory to address air game challenges has emerged as a principal research focus. Currently, prominent approaches to maneuver decision algorithms grounded in game theory encompass those based on influence diagrams and differential games.

Influence diagrams are graphical models that represent uncertain variables and decision problems [4]. The maneuver decision approach based on influence diagrams integrates prior knowledge of pilot's habits and preferences for behavioral reasoning [5].

Virtanen et al. [6] introduced a multi-stage influence diagram game to simulate a pilot's maneuver decision-making in one-on-one air games, graphically illustrating aircraft dynamics and pilot preferences under uncertain conditions. A moving-horizon control method was employed to solve the influence diagram game. To solve multi-level influence diagrams, Zhong et al. [7] transformed the multi-stage influence diagrams into a two-level optimization problem based on multi-stage influence-diagram decision-process modeling. A study by Pan et al. [8] utilized influence diagrams in the context of multi-aircraft air games and suggested a maneuver model based on state prediction influence diagrams. This model incorporated a state acquisition model using an unscented Kalman filter based on belief states. To resolve the state prediction influence diagrams, the moving horizon approach was applied, and the model was substantiated through a simulation of a two-on-two cooperative air game. Xie et al. [9] adopted a Bayesian theorem for real-time situation assessment, transforming the multi-aircraft air game model into a one-on-one air game model, and they employed influence diagrams to represent the multi-aircraft air game process. A maneuver decision algorithm based on influence diagrams integrates pilots' prior knowledge and offers strong simulation outcomes and interpretability. However, a notable drawback of this approach is its high computational complexity, which poses challenges when applying it to intricate air game environments.

Differential game theory [10] is commonly employed to address maneuver decision-making challenges in three-dimensional air games. Mukai et al. [11] devised a sequential linear quadratic method to identify local Nash solutions of general differential games and applied this technique to air game simulations. Simulation experiments demonstrated that solution estimation could converge to the local Nash (saddle) solution of the original game. Fu et al. [12] established a fixed-duration difference game model with speed, elevation angle, and yaw angle as control variables. In this model, the integral of the deviation angle and the departure angle was utilized as the cost function, while a penalty function was used to deal with constraints. Park et al.'s [13] maneuver decision algorithm features a hierarchical decision structure and leverages a differential game theory to compute the scoring function matrix, which reflects the degree of UAV air superiority. Başpınar et al. [14] employed the flatness property of the dynamical system to generate control and state sequences using a B-spline curve for the flat output. Then, by employing the security policy approach in game theory, the air game problem was transformed into an optimization problem of parameterized curves. He et al. [15] established a target-defender-attacker model and designed a nonlinear guiding rule based on differential game theory, verifying the algorithm's effectiveness through simulation experiments. Although the application of a differential games model to solve air game problems has yielded numerous theoretical results, the expansion of decision spaces can lead to a dimension explosion issue, resulting in reduced solution efficiency.

Optimization methods treat air game problems as multi-objective optimization problems subject to aerodynamic constraints. Smith et al. [16] developed a genetics-based machine learning system that employs digital simulation models of one-on-one air games and genetic algorithms to generate maneuvers. Experimental results demonstrated that the proposed system enabled the X-31 to successfully counter traditional fighter opponents. Sprinkle et al. [17] framed the control problem as a cost-minimization problem and proposed a non-linear model predictive tracking controller algorithm for evasive maneuvers in fixed-wing UAVs. McGrew et al. [18] utilized approximate dynamic programming (ADP) for UAVs to learn the optimal strategies for air games, facilitating rapid response to quickly changing tactical situations with long planning horizons and good performance. Duan et al. [19] introduced a predator-based particle swarm optimization method that effectively addresses the dynamic task assignment of UCAVs as validated by air game simulations. To enhance the efficiency and accuracy of air game confrontation evolution, Ji et al. [20] applied biological immunity to air game decision-making and developed an improved Tactical Immune Maneuver System (TIMS) model that combines sequential-association data mining with the TIMS model. Han et al. [21], considering the impact of

uncertainty factors and decision-making lag in UAV air games, proposed a robust maneuver decision-making method based on target-accessibility domain-state prediction. This method designed a robust multi-objective optimization function grounded in state function and statistical principles and utilized the grey wolf optimizer to solve real-time optimization issues. Tan et al. [22] introduced the Model Predictive Control (MPC) theory and introduced the LSHADE-TSO-MPC air game maneuver decision-making method, which predicted target UAV trajectories and accelerated air game victories. Ruan et al. [23] combined the bionic intelligent optimization algorithm with air game decision-making problems and designed a transfer-learning pigeon-inspired optimization model that demonstrated good convergence, high precision, and reasonable maneuver decisions in one-on-one dogfights. Machmudah et al. [24] solved the optimization problem of flight trajectories by using the bank-turn mechanism of fixed-wing UAVs at a constant altitude. The experimental results showed that the particle swarm optimization algorithm had good performance in path planning and flight trajectory planning. Despite the numerous methods for solving optimization problems, air game decision-making that employs optimization theory faces challenges such as difficulty in convergence and poor real-time performance due to the complexity of the air game state space.

The advancement of UAV technology and artificial intelligence has spurred research of maneuver decision algorithms. These algorithms can be categorized into reinforcement learning methods and other machine learning methods. In the latter category, machine learning algorithms are combined with autonomous maneuver decision-making to address the problem of UAV autonomous decisions in air games. Some researchers have employed neural networks to store air game rules, enabling the replication of air game behavior [25–27]. This data-driven approach does not necessitate a detailed aircraft dynamics model. However, the trained agent's decision-making ability is influenced by the training data. Machine learning algorithms, including Bayesian networks and fuzzy theory, have also been utilized in generating air game strategies. Considering the uncertainty and real-time nature of the UAV air game decision-making process, Geng et al. [28] proposed a hybrid tactical decision-making method based on rule sets and fuzzy Bayesian networks (FBN), preserving the advantages of expert systems. Similarly, Wu et al. [29] developed a situation assessment model utilizing a fuzzy reasoning machine, enabling the UAVs to conduct thorough assessment of air game benefits. Çintaş et al. [30] used deep learning to automatically detect and track another moving UAV in the air. The experimental results showed that the proposed method provided the highest accuracy rate of 82.7% and a mean fps speed of 29.6.

However, approaches relying on training data or expert systems are constrained by the training data of human experts or pilots. Reinforcement-learning-based algorithms simulate the air game environment as a Markov Decision Process (MDP) and derive optimal strategies through continuous trial and error in the simulation environment, making them suitable for various complex environments without depending on pilot training data. Recently, reinforcement learning technology has rapidly advanced, and its integration with air games has become a prominent research area.

Acknowledging sensor errors in air game environments, Kong et al. [1] represented air games as a state-adversarial Markov Decision Process (SA-MDP) and introduced an autonomous maneuver strategy generation algorithm for aircraft based on the state-adversarial deep deterministic policy gradient algorithm (SA-DDPG). The shaping reward obtained by the inverse reinforcement learning algorithm for addressing reward sparsity effectively enhances air game strategy learning speed. To solve reinforcement learning algorithm convergence speed issues, Li et al. [31] improved the deep deterministic policy gradient algorithm [32] (DDPG) convergence speed by proportionally returning the final reward value to other reward values in the same air game process. Targeting the DDPG exploration strategy's insufficient exploration and low data utilization, Wan et al. [33] incorporated a heuristic exploration strategy to enhance the algorithm's exploration capacity. Yang et al. [34] applied the proximal policy optimization algorithm [35] (PPO) to UAV air games and verified the algorithm's effectiveness through air game confrontation

simulations. Zhang et al. [36] integrated the concept of final reward with proximal strategy optimization, proposing an FRE-PPO-based air game agent training framework to improve air game agents' maneuver decision-making ability through self-play training methods. Fan et al. [37] proposed an autonomous maneuver decision-making model based on the Asynchronous Advantage Actor–Critic (A3C) algorithm, establishing a two-layer reward mechanism combining internal rewards and sparse rewards to promote faster convergence. By combining deep reinforcement learning with stochastic game theory, Cao et al. [38] designed a MinimaxDDQN network to effectively enhance the UCAV's autonomous game capability. Aiming at the problem of the generalization performance of the SAC algorithm, Li et al. [39] proposed a PSP-SAC algorithm, which can realize sample sharing and strategy sharing in various game environments, and it significantly improves the generalization performance compared with independent training. In order to solve the diversity of air game samples, Li et al. [40] also proposed a SAC algorithm based on expert actors, using a small amount of expert experience to increase the diversity of samples, which can greatly improve the exploration and utilization efficiency of deep reinforcement learning. In general, reinforcement-learning-based methods are applicable to various complex confrontation scenarios and can achieve online real-time decision-making. However, these methods exhibit low training data utilization efficiency, and the reinforcement learning algorithms employed are challenging to converge.

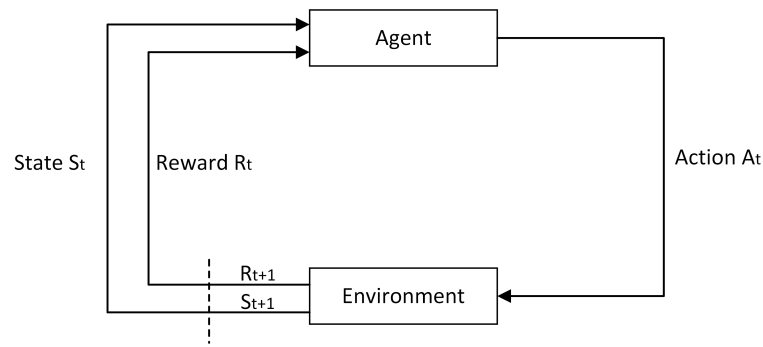
Traditional methods for UAV maneuver decision-making suffer from high computational complexity and exploding dimensions. However, for air games, reinforcement-learning-based maneuver decision-making can leverage the benefits of this learning approach in complex environments, thereby improving the dynamic adaptability of UAVs. Therefore, reinforcement learning is used for UAV maneuver decision-making in this paper. Despite the advantages of this approach, there are challenges related to the low utilization efficiency of training data and difficulties with algorithm convergence when applying reinforcement learning to air game decision-making. In real-world environment, the flying experience of air game experts is more effective than the strategies explored by novice pilots, which means incorporating expert experience into the training of UAV agents may provide a more effective strategy. Based on this idea, this paper proposes a maneuver decision algorithm based on a Dueling Double Deep Q Network (D3QN) with an expert experience storage mechanism (EESM). Firstly, the air game problem is modeled based on the MDP framework, encompassing the aircraft dynamics model, attack determination area, state space, action space, and reward function. To prevent the waste of expert experience in the replay buffer during reinforcement learning algorithm training, an expert experience storage mechanism is introduced, which aims to enhance experience storage efficiency and algorithm convergence speed. Subsequently, D3QN is incorporated into UAV air game maneuver decision-making, leading to the design of a D3QN maneuver decision algorithm based on the expert experience storage mechanism. Lastly, simulation experiments are conducted, with results indicating that the proposed mechanism accelerates the algorithm's convergence, and the suggested maneuver decision algorithm can effectively strike target UAVs by adopting various fixed strategies.

## 2. Air Game Environment Modeling

### 2.1. Markov Decision Process

The goal of reinforcement learning is to facilitate the agent's ability to learn strategies that maximize numerical rewards through continuous trial and error. The Markov decision process serves as a mathematical framework ideally suited for addressing reinforcement learning problems [41].

Within this framework, the agent selects an action and the environment responds by presenting a new state based on the chosen action. Concurrently, the environment generates reward that continuously encourage the agent to learn a strategy that maximizes the total reward. The interaction between the environment and the agent is illustrated in Figure 1.



**Figure 1.** The interaction between the environment and the agent.

As depicted in Figure 1, at each discrete time  $t$ , the agent observes the environment's state  $S_t \in \mathcal{S}$  and chooses an action  $A_t \in \mathcal{A}(s)$  based on the current strategy. In the following moment, the agent receives the immediate reward  $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ , which is given by the environment. Subsequently, the environment transitions to a new state  $S_{t+1}$  according to the action  $A_t$ .

Typically, the function  $p$  is employed to define the dynamic characteristics of MDP. For any  $s', s \in \mathcal{S}$ ,  $r \in \mathbb{R}$ , and  $A_t \in \mathcal{A}(s)$ , the following definitions are given:

$$p(s', r | s, a) \doteq \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\} \quad (1)$$

The agent aims to maximize the sum of future rewards, considering discount factors. The expected return is defined as follows:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (2)$$

To assess the agent's performance in a particular state, the concepts of action-value function (Q-function) and value function are introduced as follows:

$$\begin{aligned} v_{\pi}(s) &\doteq \mathbb{E}_{\pi}[G_t | S_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right], \\ Q_{\pi}(s, a) &\doteq \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \end{aligned} \quad (3)$$

The advantage function refers to the advantage of an action  $a$  relative to the average value in a particular state; it can be described by the following formula:

$$A_{\pi}(s, a) = Q_{\pi}(s, a) - v_{\pi}(s) \quad (4)$$

## 2.2. Aircraft Motion Modeling

Within the air game environment, our UAV and the target UAV engage in an air game within a three-dimensional space. To accurately depict the reinforcement-learning air game environment, modeling the motion of the UAV is essential.

The aircraft motion model outlines the updated state of the UAV under specific actions, signifying the transition from  $S_t$  to  $S_{t+1}$ . The primary focus of this article is the maneuver decision-making algorithm, with an emphasis on updating the UAV's position and attitude. As a result, this article introduces a particle model and treats the UAV as a particle. The UAV coordinate system employs local east, north, and up (ENU) coordinates OXYZ, where the OX-axis represents east, the OY-axis represents north, and the OZ-axis

points vertically skyward. The following kinematic equations for position information in the OXYZ coordinate system can be derived as follows:

$$\begin{cases} \frac{dx}{dt} = v \cos \theta \sin \psi \\ \frac{dy}{dt} = v \cos \theta \cos \psi \\ \frac{dz}{dt} = v \sin \theta \end{cases} \quad (5)$$

where  $x, y,$  and  $z$  denote the UAV's coordinates in the ENU coordinates OXYZ;  $v$  denotes the UAV'S velocity vector;  $dx/ dt, dy/ dt,$  and  $dz/ dt$  are the projections of  $v$  in the ENU coordinates OXYZ; and  $\theta$  and  $\psi$  symbolize the pitch and yaw angles of the UAV, respectively.

As shown in Figure 2,  $\theta$  is defined as the angle between the UAV velocity vector  $v$  and the projections of  $v$  in the O-XY-plane, and  $\psi$  is defined as the angle between the projections of  $v$  in the O-XY-plane and and the OY-axis. In the ENU coordinate system, the kinetic equations can be derived as below:

$$\begin{cases} \frac{dv}{dt} = g(n_x - \sin \theta) \\ \frac{d\theta}{dt} = \frac{g}{v}(n_z \cos \phi - \cos \theta) \\ \frac{d\psi}{dt} = \frac{gn_z \sin \phi}{v \cos \theta} \end{cases} \quad (6)$$

where  $dv/ dt$  denotes the change rate of velocity with time,  $d\theta/ dt$  denotes the change rate of pitch angle with time,  $d\psi/ dt$  denotes the change rate of yaw with time, and  $g$  refers to the acceleration of gravity. The control variables of the UAV are represented by  $[n_x, n_z, \phi]$ , where  $n_x$  refers to the longitudinal acceleration of the UAV,  $n_z$  refers to the normal acceleration of the UAV, and  $\phi$  refers to the roll angle of the UAV. In summary, given the current state of the UAV and its control variables, the state of the UAV at the next moment can be determined through Equations (5) and (6).

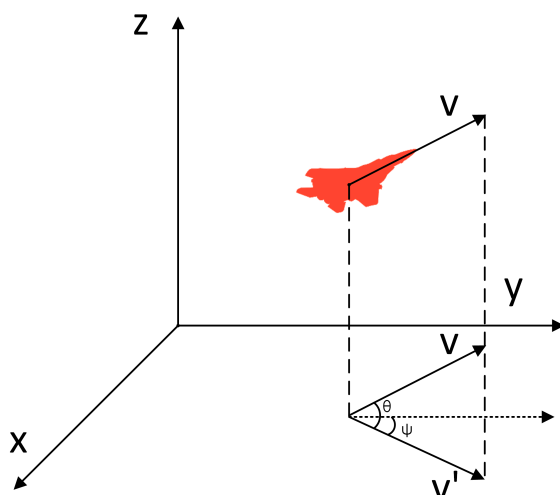


Figure 2. The particle model of the aircraft.

### 2.3. Maneuver Decision Modeling by Deep Reinforcement Learning

#### 2.3.1. Framework of Model

The air game maneuver decision algorithm proposed in this paper is based on reinforcement learning, which typically utilizes MDP as a theoretical framework. Thus, it is

necessary to model the maneuver decision process and clarify the interaction between the UAV agent and the air game simulation environment. In the air game simulation environment, the goal of our UAV agent is to complete the task of striking the target, which entails keeping the target within attack range as much as possible while ensuring the UAV’s safety. Figure 3 illustrates the framework of the UAV agent interacting with the environment.

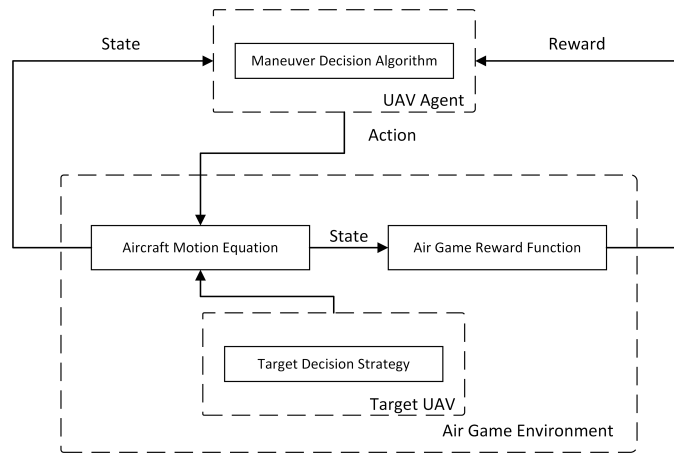


Figure 3. The framework of the UAV agent interacting with the environment.

As depicted in Figure 3, the UAV agent receives state information from the environment and selects action using the maneuver decision algorithm. The air game environment then employs the aircraft motion equation described in Section 2.2 to calculate the next state of the UAV, taking the current state and maneuvering action of both UAVs as input while simultaneously calculating the current reward value using the reward function. After the calculation, the air game environment provides feedback to the UAV agent in terms of state and reward value. The UAV agent updates the decision-making algorithm based on the reward value and state information and employs the updated algorithm to complete the maneuver decision for the next moment.

### 2.3.2. Attack Determination Model

The attack decision model outlines the ultimate goal of the UAV agent and provides a detailed definition of the conditions required for the UAV agent to complete the air game cycle. The objective of the UAV agent is to acquire a higher reward value and successfully strike the target UAV.

Figure 4 defines the attack range of the UAV, where  $\theta_{attack}$  denotes the attack angle and  $D_{attack}$  denotes the attack distance. When the target UAV enters the attack zone of our UAV, this indicates that our UAV has successfully executed the attack. Equation (6) mathematically represents the decision condition for the success of a UAV attack:

$$\begin{cases} d < D_{attack} \\ q_U < \theta_{attack} \end{cases} \tag{7}$$

where  $q_U$  denotes the angle between the UAV’s speed and the line connecting the UAV and the target, and  $d$  denotes the distance between the UAV and the target. Given UAV information  $(x_U, y_U, z_U, \theta_U, \psi_U, \phi_U)$  and target information  $(x_T, y_T, z_T, \theta_T, \psi_T, \phi_T)$ ,  $q_U$  and  $d$  can be calculated by the following equation:

$$\begin{aligned} d &= \sqrt{(x_U - x_T)^2 + (y_U - y_T)^2 + (z_U - z_T)^2} \\ q_U &= \arccos\{[(x_T - x_U) \cos \phi_U \cos \theta_U \\ &\quad + (y_T - y_U) \sin \phi_U \cos \theta_U + (z_T - z_U) \sin \theta_U] / d\} \end{aligned} \tag{8}$$

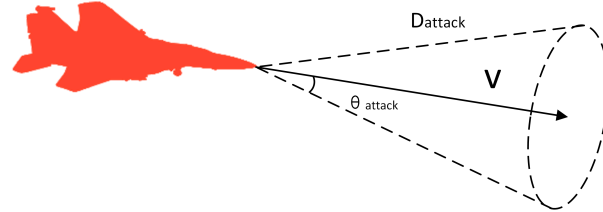


Figure 4. The attack determination model of the UAV agent.

### 2.3.3. State Definition

The UAV state space represents the environment information perceived by the UAV agent. When designing the state space, it is necessary to filter and extract the environmental information. This information includes both directly related information about the goal and indirectly related information. The goal of the UAV agent is to complete a strike on the target, and the directly related information includes the distance  $d$  between the UAV and the current attack angle  $q_U$  of the UAV. The indirectly relevant information is the environment information that is not directly related to the agent’s goal, but the agent can learn the strategy to achieve the goal through this indirect information. The indirectly related information for the UAV agent includes position information  $(x_U, y_U, z_U, x_T, y_T, z_T)$ , attitude information  $(\theta_U, \psi_U, \phi_U, \theta_T, \psi_T, \phi_T)$ , velocity information  $(v_U, v_T)$ , and attack information  $q_T$ . By integrating and normalizing the directly related information and profile-related information, the designed state space is illustrated in Table 1.

Table 1. The state space of the UAV agent.

State	Definition	State	Definition
$s_1$	$\frac{x_U - x_T}{d_0}$	$s_8$	$\frac{q_T}{\pi}$
$s_2$	$\frac{y_U - y_T}{d_0}$	$s_9$	$\frac{\theta_U}{2\pi}$
$s_3$	$\frac{z_U - z_T}{d_0}$	$s_{10}$	$\frac{\psi_U}{2\pi}$
$s_4$	$\frac{d}{d_0}$	$s_{11}$	$\frac{\phi_U}{2\pi}$
$s_5$	$\frac{v_U - v_0}{v_1}$	$s_{12}$	$\frac{\theta_T}{2\pi}$
$s_6$	$\frac{v_T - v_0}{v_1}$	$s_{13}$	$\frac{\psi_T}{2\pi}$
$s_7$	$\frac{q_U}{\pi}$	$s_{14}$	$\frac{\phi_T}{2\pi}$

Where  $d_0, v_0, v_1$  are normalized constants. The position information adopts normalized relative coordinate information  $(\frac{x_U - x_T}{d_0}, \frac{y_U - y_T}{d_0}, \frac{z_U - z_T}{d_0})$ , which can make the UAV agent have good performance in different coordinate systems. The variable  $q_T$  is the departure angle of the target, which is obtained by calculating the angle between the target’s speed and the line connecting the UAV and the target:

$$q_T = \arccos\{[(x_T - x_U) \cos \phi_T \cos \theta_T + (y_T - y_U) \sin \phi_T \cos \theta_T + (z_T - z_U) \sin \theta_T] / d\} \tag{9}$$

### 2.3.4. Action Definition

According to Equation (6),  $[n_x, n_z, \phi]$  are selected as the control variables of the UAV:  $n_x$  represents the longitudinal overload of the UAV, indicating the acceleration capability of the UAV along the direction of the flight speed;  $n_z$  represents the normal overload of the UAV, which is vertical to the flight speed direction; and  $\phi$  is the roll angle of the UAV. Attitude correction can be achieved through the control of  $\phi$  and  $n_z$ . Using  $[n_x, n_z, \phi]$  as the



maneuver action implies that the action space of the UAV agent is continuous. Directly discretizing the continuous action space can lead to the curse of dimensionality, making it challenging for the algorithm to converge.

In real air games, NASA has defined a library of fundamental flight maneuvers for aircraft, including straight flight, acceleration, deceleration, left turn, right turn, upward flight, and downward flight [42]. The combination of different basic maneuvers can create more advanced stunts (e.g., Immelmann turns, loops, etc.). In this paper, taking into account air game background knowledge, the action space is defined as a set of seven basic actions, which avoids the curse of dimensionality caused by directly discretizing the continuous action space. The basic maneuvering action of the UAV can be expressed by setting the values of different control variables. Table 2 displays the action space of the UAV in this study and the corresponding control value settings.

**Table 2.** The action space and the settings of the corresponding control values.

Action	Definition	$n_x$	$n_z$	$\phi$
$a_1$	straight flight	0	1	0
$a_2$	acceleration	2	1	0
$a_3$	deceleration	−2	1	0
$a_4$	left turn	0	8	$-\arccos(1/8)$
$a_5$	right turn	0	8	$\arccos(1/8)$
$a_6$	upward flight	0	8	0
$a_7$	downward flight	0	8	$\pi$

### 2.3.5. Reward Function Definition

The reward function's design should ensure that the agent can accomplish its objective while maximizing the reward. In this study, the UAV agent's goal is to complete the task of striking the target, which entails keeping the target within attack range as much as possible while ensuring the UAV's safety. For the UAV agent's goal, the reward function design can be separated into objective rewards and sub-objective rewards.

The objective reward pertains to the positive reward  $R_0$  given to the UAV agent when the goal is achieved, which is typically unbiased towards the goal. When the target is within the UAV's attack range, a positive reward value  $R_p$  is given to the UAV agent; when the UAV is within the target's attack range, a negative reward value  $R_n$  is given to the UAV agent. To encourage the UAV agent to complete the strike mission as quickly as possible, a time penalty  $R_t$  is introduced to guide the UAV towards swift mission completion.  $R_0$  can be expressed using the following equation:

$$R_0 = \begin{cases} R_p, & \text{if the UAV completes attack mission;} \\ R_n, & \text{if the UAV is wrecked;} \\ R_t, & \text{else.} \end{cases} \quad (10)$$

However, relying solely on objective rewards may result in sparse rewards, which could leave the UAV agent without proper guidance in exploration direction. By incorporating additional reward or penalty components based on the objective reward, the reward function becomes denser, providing adequate guidance to the UAV agent. This can accelerate the convergence of the deep reinforcement learning algorithm and enhances performance. Auxiliary reward components are designed by breaking down the objectives into sub-objectives and assigning different weights according to each sub-objective's contribution to the overall goal realization. The goal of the UAV strike mission is divided into three sub-objectives:

1. The distance between the UAV and the target should be within the attack range;
2. The angle between the UAV and the target should be outside the attack angle of the target;

3. The angle between the UAV and the target should be within the attack angle of the UAV.

According to the sub-goals of the UAV agent,  $R_1$ ,  $R_2$ ,  $R_3$  are introduced.  $R_1$  is the distance reward function, which increases as the UAV gets closer to the target. The distance reward function  $R_1$  can guide the UAV agent to reduce the distance to the target and to rapidly enter the UAV's attack range. The distance reward function  $R_1$  is defined as follows:

$$R_1 = 1 - \left(\frac{d}{D_r}\right)^{0.4} \quad (11)$$

where  $D_r$  is the distance reward constant.  $R_2$  is the angle reward function, considering both the UAV's attack angle and the target's departure angle. The angle reward function  $R_2$  can direct the UAV agent to maintain an offensive position and approach the target while ensuring its own safety. The angle reward function  $R_2$  is defined as follows:

$$R_2 = \frac{\pi - (q_U + q_T)}{\pi} \quad (12)$$

$R_3$  is the attack reward function, which increases as the target approaches the attack angle. The attack reward function  $R_3$  encourages the UAV agent to actively attack and is defined using the following equation:

$$R_3 = 1 - \left(\frac{q_U}{\pi}\right)^{0.4} \quad (13)$$

Based on the contributions of  $R_1$ ,  $R_2$ , and  $R_3$  in the process of achieving the goal,  $c_1$ ,  $c_2$ , and  $c_3$  are introduced as the weights of each sub-objective reward function. The reward function  $R_t$  of the UAV agent is composed of the objective reward function and the sub-objective reward functions as follows:

$$R_t = R_0 + c_1 * R_1 + c_2 * R_2 + c_3 * R_3 \quad (14)$$

### 3. UAV Maneuver Decision Algorithm

#### 3.1. Expert Experience Storage Mechanism

Traditional reinforcement learning algorithms tend to discard transitions after learning from them once, leading to inefficient use of experience. Moreover, strong correlation between adjacent transitions is not conducive to the agent learning. In order to alleviate the problems of data correlation and experience waste, experience replay technology is introduced in the DQN algorithm [43]. Experience replay stores the agent's experience at each time step in a dataset, accumulating transitions from multiple episodes into a replay buffer  $R$ , and uniformly samples randomly from  $R$  when performing an update, where  $R = \tau_1, \dots, \tau_N, \tau_t = (s_t, a_t, r_t, s_{t+1})$ . The current oldest transition is deleted when a new transition is stored and the replay buffer is full.

Fedus et al. [44] studied the impact of three factors in the replay buffer, namely replay capacity, age of the oldest policy, and replay ratio. The experimental results proved that an increase in replay capacity greatly improves the performance of n-step DQN series algorithms. However, not all transitions hold equal importance. Based on this idea, Schaul et al. [45] proposed a prioritized experience replay (PER) mechanism, using TD-error as a judgment basis. The experimental results showed that DQN with prioritized experience replay outperformed the original DQN. Continuing with this idea, Karimpanal et al. [46] optimized the experience storage process by collecting high-reward sequences during interactions and using prioritized experience replay during the learning process.

However, in aerial combat games, the goal is not to obtain higher reward values but to successfully attack the target UAV. In a real air game environment, the flying experience of an air game expert is often more valuable than the strategies independently explored by novice pilots. Consequently, when training the UAV agent, the replay buffer should

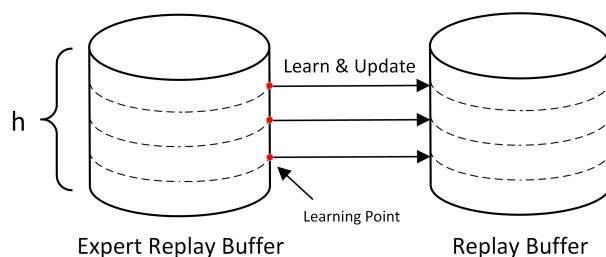
ideally contain a higher proportion of expert experience. Although obtaining genuine expert experience can be challenging, the agent may successfully strike the target during its exploration process. Compared to random exploration experience, instances of successful target strikes can be considered as the UAV agent’s expert experience.

Based on the above analysis, this paper proposes an expert experience storage mechanism to optimize the experience storage process and uses the expert experience collected by UAV agents during the exploration process to enhance algorithm performance. When storing transitions in the replay buffer, the sequence of transitions is also recorded. In addition to experience replay technology, the proposed mechanism incorporates goal attainment determination; specifically, it assesses whether the UAV has successfully completed its strike mission. When the goal is achieved, the recorded sequence  $\tau$  is regarded as an expert experience sequence from which the agent can learn. The recorded sequence  $\tau$  is stored in both the original replay buffer and the expert replay buffer. Same as for the original replay buffer, the oldest expert experience sequence will be removed when the expert replay buffer is full. To balance the UAV agent’s exploration and utilization of expert experience in the air game environment, two measures are proposed: centralized learning and replay buffer replacement.

Centralized learning refers to setting multiple learning points in the replay buffer, and when the amount of stored experience reaches the learning point, the existing expert experience is learned multiple times. During UAV exploration, updates typically follow the principle of obtaining higher reward values. However, inaccurately designed reward functions can lead to situations where the agent achieves high reward values but fails to accomplish the goal. Learning from the expert replay buffer can rectify the strategy update direction, shifting the agent from being reward-oriented to goal-oriented.

Nonetheless, during the initial stage of training, learning from an expert experience sequence may not effectively correct the direction or might even steer it in the wrong direction. Hence, a centralized learning method is necessary, which entails the agent learning multiple times from the experience sequence. However, this strategy can cause the agent to become trapped in a local optimum at the beginning of training.

An intuitive solution is to perform centralized learning on the agent each time a certain amount of successful experience is gathered. As illustrated in Figure 5, the replay buffer is evenly divided into  $h$  parts, with the capacity line of each part serving as the UAV agent’s learning point. When the stored experience reaches the learning point, the existing expert experience is learned  $k$  times by the agent.



**Figure 5.** Centralized learning and replay buffer replacement during the process of EESM.

During the exploration process, the UAV agent employs the expert replay buffer to conduct gradual update learning. After centralized learning, the agent’s strategy direction is adjusted. At this point, continuing to use the original replay buffer for exploration is not suitable; instead, exploration should be based on the expert experience replay buffer. Consequently, the proposed algorithm replaces the original replay buffer simultaneously with centralized learning each time, facilitating exploration grounded in expert experience.

See Algorithm 1 for a more formal description of the proposed algorithm.

**Algorithm 1:** Expert Experience Storage Mechanism

---

**Input:** a reinforcement learning off-policy algorithm  $\mathbb{A}$ , self-learning time  $k$ , storage constant  $h$ , and expert replay buffer size  $N$

Initialize  $\mathbb{A}$

Initialize replay buffer  $R$

Initialize expert replay buffer  $R'$

**for** episode = 1,  $M$  **do**

  Initialize transition sequence  $\tau = \{ \}$

  Initialize state  $s_1$

**for**  $t = 1, T - 1$  **do**

    Select action  $a_t$  using the policy from  $\mathbb{A}$

    Execute action  $a_t$  and observe reward  $r_t$  and  $s_{t+1}$

    Store transition  $(s_t, a_t, r_t, s_{t+1})$  in the replay buffer  $R$

    Add transition  $(s_t, a_t, r_t, s_{t+1})$  to the transition sequence  $\tau$

    Sample a minibatch transition  $\tau_{batch}$  from the replay buffer  $R$

    Update weights through a minibatch transition  $\tau_{batch}$  and  $\mathbb{A}$

**end**

**if** goal achieved **then**

    Store transition sequence

$\tau = \{(s_1, a_1, r_1, s_2), (s_2, a_2, r_2, s_3), \dots, (s_t, a_t, r_t, s_{t+1}, \dots)\}$  in the expert replay buffer  $R'$

$m =$  current buffer memory mount

**if** current buffer memory mount  $m > \frac{N}{h} * n$  **then**

$n \leftarrow n + 1$

$R \leftarrow R'$

**for**  $j = 1, k$  **do**

        Sample a minibatch transition  $\tau_{batch}$  from the replay buffer  $R'$

        Update weights through a minibatch transition  $\tau_{batch}$  and  $\mathbb{A}$

**end**

**end**

**end**

**end**

---

### 3.2. Dueling Double Deep Q Network Algorithm

DDPG [32] and PPO [35] are algorithms widely used for decision-making in continuous action space games, while DQN series of algorithms are typically used for tasks with a discrete action space. In this paper, Dueling Double Deep Q Network is selected as the algorithm for UAV maneuver decision-making. D3QN algorithm integrates the central concepts of both Double DQN [47] and Dueling DQN [48] algorithms, enhancing performance beyond the original DQN. This section provides an overview of the key components of the D3QN algorithm.

In conventional Q-learning, a Q-function is introduced, which is the expected future reward for performing a specific action in a specific state. The value of each Q-function is recorded through the Q-table, which determines the agent's decisions based on an  $\epsilon$ -greedy strategy [43]. The Q-table iteratively updates  $Q(s_t, a_t)$  utilizing the Bellman equation:

$$Q(s, a) \leftarrow Q(s, a) + l \left[ r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \quad (15)$$

where  $l$  denotes the learning rate, a hyperparameter that influences if and when the objective function converges to a local minimum. Through continuous trial and error, the agent eventually learns a convergent Q-table, which enables it to select the action that yields the maximum expected reward.

The advancement of deep learning has facilitated the integration of reinforcement learning and deep learning. The DQN algorithm, proposed by the DeepMind team in 2015 [43], employs a neural network in place of the Q-table. The current state serves as the input to the neural network, which produces Q values as output. The DQN model comprises two networks: the predict Q network and the target Q network. The predict Q network is employed to predict the Q value of each action corresponding to the current state, while the target Q network is used to predict the Q value of each action in the subsequent state. When calculating the loss function in the network, the target Q network will not be updated with the model at any time. After a certain number of iterations, the parameters of the predict Q network are transferred to the target network for correction. The objective function of network update can be defined as follows:

$$Y_t^{\text{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-) \quad (16)$$

However, there is an overestimation problem in DQN. After Q network initialization, the estimation of the Q value is random. Due to the maximization operation used in updating the Q value, the final Q value is often overestimated. To alleviate this problem, the Double DQN algorithm separates action selection and evaluation. In the predict Q network, it finds the action that corresponds to the maximum Q value and then uses this selected action to calculate the target Q value in the target Q network. Based on this idea, Double DQN modifies the objective function:

$$Y_t^{\text{DoubleDQN}} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a}{\operatorname{argmax}} Q(S_{t+1}, a; \theta_t), \theta_t^-\right) \quad (17)$$

The model structure of Double DQN is identical to that of DQN, encompassing both the predict Q network and the target Q network. The optimal action selection in DQN is based on the parameter  $\theta^-$  of the target Q network, while the optimal action selection in Double DQN is based on the parameter  $\theta$  of the predict Q network currently being updated.

Double DQN retains the model structure of DQN while enhancing the algorithm's performance through the modification of the objective function. Contrasting with Double DQN, Dueling DQN concentrates on the neural network structure and optimizes the Q network model. The Dueling DQN algorithm introduces a new neural network structure known as the dual network.

As shown in Figure 6, the output of the Dueling DQN algorithm comprises two branches, which are the state value function  $V$  and the advantage function value  $A$  for each action. The Q value is represented by the sum of  $V$  and  $A$ . To enhance the distinguishability of  $V$  and  $A$ , Dueling DQN centralizes the advantage function and imposes constraints using the following equation:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left( A(s, a; \theta, \alpha) - \max_{a' \in |\mathcal{A}|} A(s, a'; \theta, \alpha) \right) \quad (18)$$

where  $\beta$  is a local network parameter unique to the value function and  $\alpha$  is a local network parameter unique to the advantage function.

The D3QN algorithm is established upon the foundation of the DQN algorithm, integrating concepts from both the Double DQN algorithm and the Dueling DQN algorithm to optimize the DQN model structure and optimize the objective function.

### 3.3. Maneuver Decision Algorithm Based on Dueling Double Deep Q Network with Expert Experience Storage Mechanism

Based on D3QN and EESM, a UAV air game maneuver decision algorithm is developed. The comprehensive algorithm framework is illustrated in Figure 7.

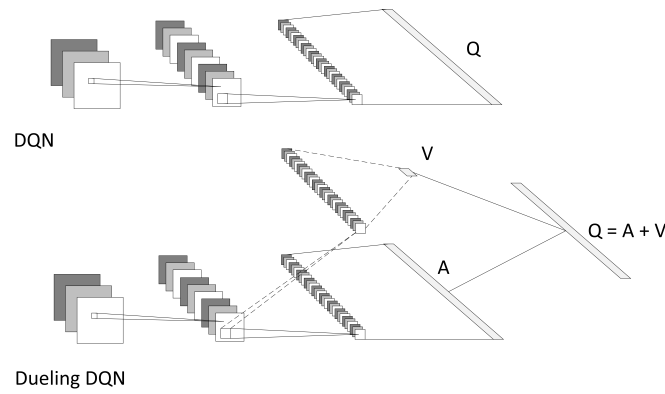


Figure 6. The structure of the DQN model and Dueling DQN.

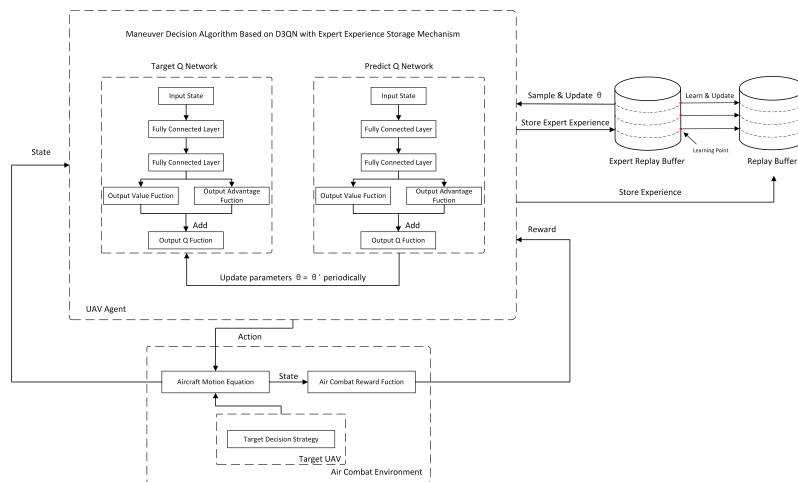


Figure 7. The comprehensive framework of the maneuver decision algorithm based on D3QN with EESM.

The current state and reward values are calculated from the UAV air game environment through Equations (5) and (6) and the reward function. The predict Q network takes the current state value as input. After processing through two fully connected layers, the predict Q network calculates the value function and advantage function separately and employs Equation (18) to derive a series of Q values. The  $\epsilon$ -greedy strategy is used to complete the action selection and interact with the UAV air game environment.

Throughout the training process, the UAV agent acquires strategy by sampling from the replay buffer, which comprises both the original replay buffer and an expert replay buffer. The original replay buffer serves to disrupt the correlation between samples and update the network parameters at each training step, while the expert replay buffer encompasses the experience of the UAV successfully attacking the target aircraft. Centralized learning is conducted when the mission is accomplished. Upon the accumulation of a specific amount of expert experience, the contents of the expert replay buffer are duplicated into the original replay buffer.

During the process of sampling from the replay buffer, the UAV state value is simultaneously transmitted to both the target Q network and the predict Q network. The predict Q network supplies the predict Q value for action selection, while the target Q value is used for calculating the objective function in the network parameter update. Through the target Q network and the predict Q network outputs, the parameter update for the predict Q network can be accomplished based on the objective value in Equation (17). Utilizing the fixed target network method, the parameters of the predict Q network are copied to the target Q network at regular intervals. When the UAV successfully strikes the target during the training phase, expert experience is sampled from the expert replay buffer

for centralized learning. The learning process is identical to that of ordinary experience sampled from the replay buffer.

The training process of the proposed algorithm is presented in Algorithm 2.

---

**Algorithm 2:** UAV Maneuver Decision Algorithm

---

**Input:** self-learning time  $K$ , storage constant  $h$ , expert replay buffer size  $N$ , discount factor  $\gamma$ , batch size  $B$ , learning rate  $l$ , copy network step  $C$ ,  $\alpha$ , and  $\beta$

Initialize predict Q network parameter  $\theta$

Initialize target Q network parameter  $\theta^-$

Initialize replay buffer  $R$

Initialize expert replay buffer  $R'$

**for** episode = 1,  $M$  **do**

Initialize transition sequence  $\tau = \{ \}$

Initialize state  $s_1$

**for**  $t = 1, T - 1$  **do**

Calculate the Q value through the predict Q network

Select action  $a_t$  using  $\epsilon$ -greedy policy

Excute action  $a_t$  and observe reward  $r_t$  and  $s_{t+1}$

Store transition  $(s_t, a_t, r_t, s_{t+1})$  in the replay buffer  $R$

Add transition  $(s_t, a_t, r_t, s_{t+1})$  to the transition sequence  $\tau$

Sample a minibatch transition  $\tau_{batch}$  from the replay buffer  $R$

**for**  $j = 1, B$  **do**

Compute TD-error

$$Y_j^{\text{DoubleDQN}} \equiv R_{j+1} + \gamma Q\left(S_{j+1}, \underset{a}{\operatorname{argmax}} Q(S_{j+1}, a; \theta_j), \theta_j^-\right)$$

Update weights

$$\theta_{j+1} = \theta_j + l \left( Y_j^{\text{DoubleDQN}} - Q(S_j, A_j; \theta_j) \right) \nabla_{\theta_j} Q(S_j, A_j; \theta_j)$$

**end**

Every  $C$  steps, copy weights into target network  $\theta^- \leftarrow \theta$

**end**

**if** goal achieved **then**

Store transition sequence

$\tau = \{(s_1, a_1, r_1, s_2), (s_2, a_2, r_2, s_3), \dots, (s_t, a_t, r_t, s_{t+1}, \dots)\}$  in the expert replay buffer  $R'$

$m =$  current buffer memory mount

**if** current buffer memory mount  $m > \frac{N}{h} * n$  **then**

**end**

$n \leftarrow n + 1$

$R \leftarrow R'$

**for**  $k = 1, K$  **do**

Sample a minibatch transition  $\tau_{batch}$  from the replay buffer  $R'$

**for**  $j = 1, B$  **do**

Compute TD-error

$$Y_j^{\text{DoubleDQN}} \equiv R_{j+1} + \gamma Q\left(S_{j+1}, \underset{a}{\operatorname{argmax}} Q(S_{j+1}, a; \theta_j), \theta_j^-\right)$$

Update weights

$$\theta_{j+1} = \theta_j + l \left( Y_j^{\text{DoubleDQN}} - Q(S_j, A_j; \theta_j) \right) \nabla_{\theta_j} Q(S_j, A_j; \theta_j)$$

**end**

**end**

**end**

**end**

---

## 4. Simulation Experiment and Analysis

### 4.1. Simulation Experiment Basic Setup

This study establishes a simulation environment on a Windows 10 operating system with hardware support from an NVIDIA RTX2060 GPU and an Intel Core i7-9700k processor. The proposed UAV maneuver decision-making algorithm is implemented using the Pytorch module.

The basic parameters of the simulation environment are shown in Table 3. The threshold distance  $D_{threshold} = 5000$  m, state space normalization parameter  $v_1 = 20$  m/s,  $v_0 = 80$  m/s, UAV velocity  $v \in [60$  m/s, 100 m/s], attack distance  $D_{attack} = 500$  m, attack angle  $\theta_{attack} = 60^\circ$ .

**Table 3.** The basic parameters of the simulation environment.

Parameter	Value
Threshold distance $D_{threshold}$	5000 m
State space normalization parameter $v_1$	20 m/s
State space normalization parameter $v_0$	80 m/s
UAV velocity range	[60 m/s, 100 m/s]
attack distance $D_{attack}$	500 m
learning rate $l$	0.0003
initial greedy coefficient $\epsilon$	1
greedy coefficient $\epsilon$ decay rate	$3 \times 10^{-5}$
discount factor $\gamma$	0.98
reward factor $c_1$	0.7
reward factor $c_2$	0.15
reward factor $c_3$	0.15
replay buffer size	$10^6$
batch size	256
storage constant $h$	10
self-learning time $k$	10
number of hidden layers	2
number of network cells in each layer	256

The D3QN network and evaluation network in this study both have two hidden layers, with each layer consisting of 256 network cells. The Adam optimizer is used to train the D3QN network with a learning rate  $l$  set to 0.0003. The initial greedy coefficient  $\epsilon$  is set to 1 and decays to 0.05 with a decay rate of  $3 \times 10^{-5}$ . The discount factor  $\gamma = 0.98$ , while the reward factors are  $c_1 = 0.7$ ,  $c_2 = c_3 = 0.15$ . The replay buffer size is set to  $10^6$  and the batch size to 256. The self-learning time  $k$  is set to 10.

Two simulation experiment were conducted in order to verify the proposed method using the basic parameters mentioned above. The ablation experiment was designed to test the effectiveness of the expert experience storage mechanism, while an air game simulation experiment was set up in a simulation environment to verify the performance of the maneuver decision algorithm.

### 4.2. Expert Experience Storage Mechanism Ablation Experiment

In this paper, an expert experience storage mechanism is proposed to solve exploration efficiency in the training model. Prioritized experience replay [45] is an experience-replay-based reinforcement learning technique that adjusts the sampling probability of experience samples based on their priority. To evaluate the effectiveness of the expert experience storage mechanism, ablation studies are performed on the basis of two mechanisms, and four groups of experiment are compared:

1. Training and testing with both PER and the expert experience storage mechanism;
2. Training and testing with only PER and without the expert experience storage mechanism;
3. Training and testing with only the expert experience storage mechanism and without PER;
4. Training and testing without either PER or the expert experience storage mechanism.



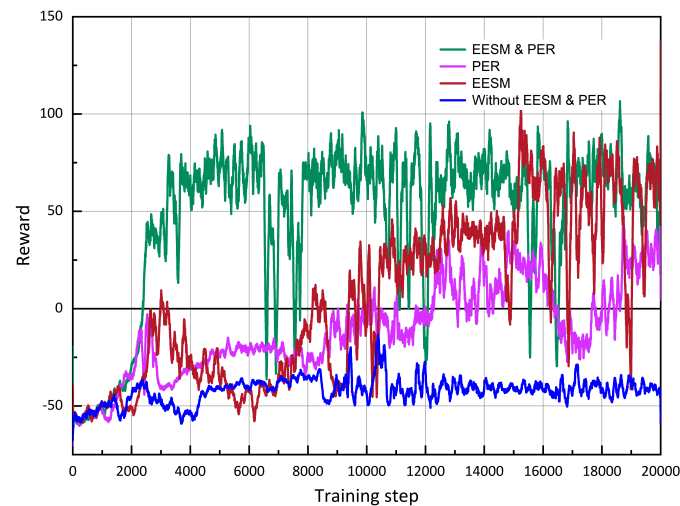
In each experiment, the initial UAV situation of each training step is shown in Table 4. The target UAV adopts the strategy of uniform speed and direct flight, which mean the target UAV executes  $a_1$  in the action space. Our UAV is initially behind the target UAV, and the previously stated maneuver decision algorithm training method is used to learn the decision-making process.

**Table 4.** The initial UAV situation at each training step.

	Initial Position ( $x, y, z$ )	$v$ (m/s)	$\theta$	$\psi$	$\phi$
Our UAV	(10,300, 20,000, 25,000)	60	0	0	0
Target UAV	(11,300, 21,000, 26,000)	60	0	0	0

During the training process, the total reward value and experience replay time obtained are recorded in each episode. The total reward value received in each episode can be used to reflect the performance of the algorithm under different sampling mechanisms. The experience replay time refers to the time it takes for the replay buffer to be sampled from storage, indicating the efficiency of different sampling mechanisms. The effectiveness of the proposed mechanism is validated by comparing the total reward values and experience replay times obtained for each experiment.

The training curves for various sampling mechanisms are presented in Figure 8.



**Figure 8.** Comparison of the learning curves with different experiment setting. The training reward is calculated as mentioned in Section 2.3.4. The final presented reward is a smoothed average reward function, which can reflect the overall performance of the UAV agent.

As shown in Figure 8, without the use of EESM and PER mechanisms, the maximum average reward value obtained by the UAV agent is  $-13.5$ . When the original sampling mechanism was used, the UAV agent received a relatively low reward, indicating that the agent did not learn the strategy to obtain higher rewards as the training steps increased. On the other hand, when the PER mechanism was used, the UAV agent obtained a maximum average reward value of  $42.5$ , which was higher than that obtained by using the original sampling mechanism. The proposed EESM mechanism allowed the UAV agent to achieve a maximum average reward value of  $102.3$  in the air game simulation environment, outperforming the PER mechanism and the original sampling mechanism. This suggests that the EESM mechanism is effective in helping the UAV agent learn better attack strategies. The maneuver decision-making algorithm that uses both PER and EESM mechanisms achieves a maximum average reward value of  $106.6$ , which is similar to the reward obtained by using the EESM alone. However, with PER and EESM, the algorithm's overall convergence

speed is faster, suggesting that PER has a certain acceleration effect on the algorithm's convergence speed when fully utilizing expert experience.

Table 5 displays the experience replay times for different sampling mechanisms. The EESM mechanism has an experience replay time of 178.5 ms, slightly higher than the experience replay time without the EESM and PER mechanisms, which is 101.0 ms. The experience replay times for the PER mechanism and the combined PER and EESM mechanisms are 1149.2 ms and 1577.0 ms, respectively. These replay times indicate that the proposed EESM mechanism has a slightly slower replay speed compared to the original mechanism but is much faster than the PER mechanism, which is typically implemented using a sumtree.

**Table 5.** The details of each ablation experiment.

	PER	EESM	Experience Replay Time	Max Average Reward
Experiment 1	✓	✓	1577.0 ms	106.6
Experiment 2	✓		1149.2 ms	42.5
Experiment 3		✓	178.5 ms	102.3
Experiment 4			101.0 ms	−13.5

The overall algorithm performance should be assessed by considering the training curve and experience replay time. By comparing the reward curves of the four experimental groups, it is observed that the EESM and PER mechanisms improve the UAV intelligent agent's learning effect, and the maneuver decision algorithm that uses the EESM mechanism obtains higher reward values. Then, the algorithm that uses both EESM and PER mechanisms achieves approximately the maximum reward value compared to the algorithm that only uses the EESM mechanism but with a faster convergence rate. In addition, it is found that the PER mechanism is time-consuming, while the experience replay time using the EESM mechanism is slightly higher than the original sampling mechanism but reduced by 81.3% compared to the time taken using the PER mechanism. Taking into account both the maximum reward value and sampling time, the following can be concluded:

1. Without using the EESM and PER mechanisms, although the experience replay time of the algorithm is less, the UAV intelligent agent has difficulty learning an effective strategy for attacking the target UAV.
2. The UAV intelligent agent can achieve a certain training effect when only using the PER mechanism, but the replay experience time is greatly increased. The algorithm's performance is further improved after introducing the proposed EESM mechanism.
3. The EESM mechanism proposed in this study can significantly improve the performance of the algorithm with less replay time: it reduces the replay time of the algorithm by 81.3% compared to the PER mechanism and makes the UAV intelligent agent obtain a higher maximum average reward value.

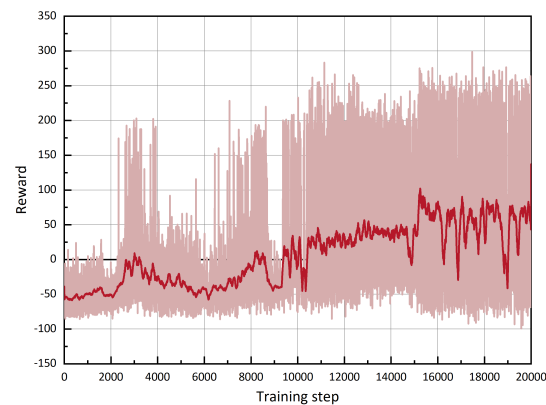
#### 4.3. Air Game Simulation Experiment Based on Maneuver Decision Algorithm

In this section, an air game simulation experiment was conducted using the maneuver decision algorithm with the proposed expert experience storage mechanism. In the simulation experiment, our UAV maneuver decision algorithm adopts the proposed expert experience storage mechanism. The initial situation of the UAV in each training step was the same as the parameters presented in Table 3. The target UAV adopts three different strategies:

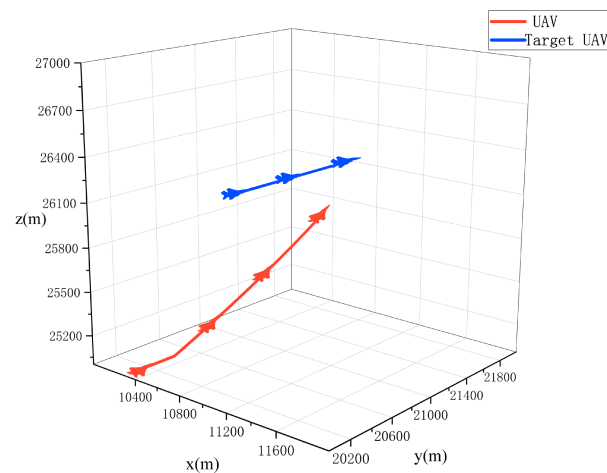
1. Uniform speed and direct flight: Target UAV executes  $a_1$  in the action space.
2. Uniform speed and hovering: Target UAV executes  $a_4$  in the action space.
3. Uniform speed and serpentine maneuver: Target UAV executes  $a_4$  and  $a_5$  periodically, with the period being 35 simulation steps.

Figures 9 and 10 show the training curve and typical air game flight trajectory of our UAV agent when the target UAV adopts the strategy of uniform speed and direct

flight. As shown in Figure 9, as the number of training steps increases, the proposed autonomous maneuver decision-making algorithm gradually converges and finally obtains the maximum reward value of 298.7. As depicted in Figure 10, the initial position of the target UAV is above our UAV in horizontal flight. Our UAV observes the state information of the target UAV and uses the proposed autonomous maneuver decision-making algorithm to select the appropriate maneuvering actions. After flying straight for a brief period, our UAV initiates a climb and attacks the target UAV positioned from below.

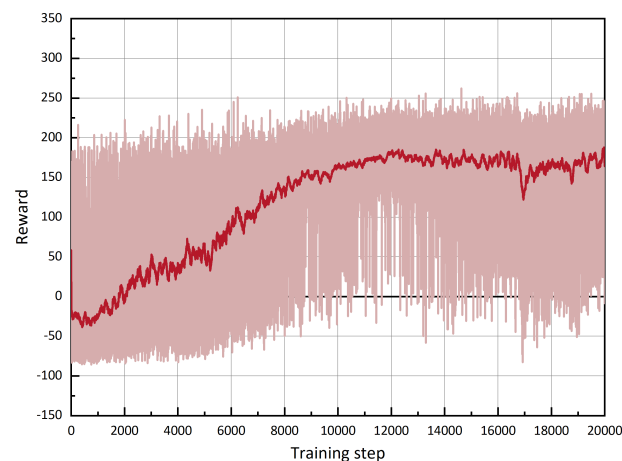


**Figure 9.** The reward value obtained by the UAV agent during the training episode wherein the target adopts the strategy of uniform speed and direct flight.

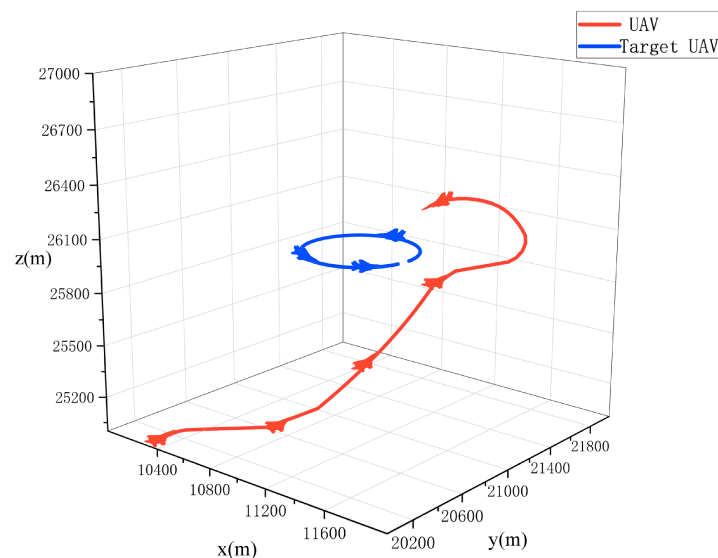


**Figure 10.** The typical flight trajectory during the training episode wherein the target adopts the strategy of uniform speed and direct flight.

The results of the air game simulation experiment with the target UAV adopting the uniform speed and hovering strategy is shown in Figures 11 and 12. As illustrated in Figure 11, the proposed autonomous maneuver decision-making algorithm gradually converges with increasing training steps, achieving a maximum reward of 262.3. Figure 12 depicts the target UAV hovering with our UAV's initial position below. Using the proposed autonomous maneuver decision-making algorithm, our UAV takes the state information of the aerial game as inputs and outputs real-time maneuvering actions. Then, our UAV adjusts its flight direction and performs a climb to occupy the optimal attack position above the target UAV to complete the attack.

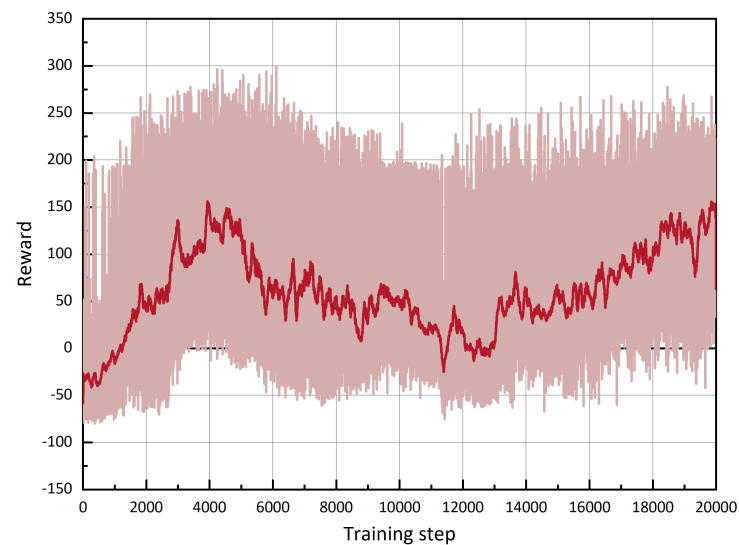


**Figure 11.** The reward value obtained by the UAV agent during the training episode wherein the target adopts the strategy of uniform speed and hovering.

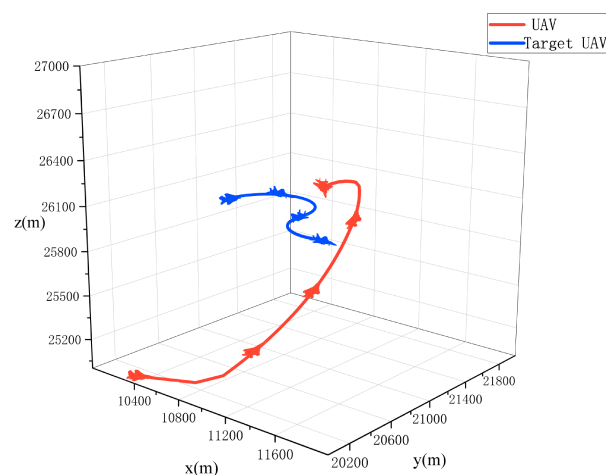


**Figure 12.** The typical flight trajectory during the training episode wherein the target adopts the strategy of uniform speed and hovering.

The results of the air game simulation experiment with the target UAV adopting uniform speed and serpentine maneuver is shown in Figures 13 and 14. As illustrated in Figure 13, the proposed autonomous maneuver decision-making algorithm gradually converges, ultimately achieving a maximum reward of 299.6. Figure 14 shows the target UAV performing a serpentine maneuver while our UAV adopts the proposed algorithm. As the target UAV performs the serpentine maneuver, our UAV adjusts its direction and moves straight for a short period. Finally, our UAV climbs and positions itself above the enemy UAV for an optimal attack.



**Figure 13.** The reward value obtained by the UAV agent during the training episode wherein the target adopts the strategy of uniform speed and serpentine maneuver.



**Figure 14.** The typical flight trajectory during the training episode wherein the target adopts the strategy of uniform speed and serpentine maneuver.

## 5. Conclusions

When utilizing deep reinforcement learning for UAV air game maneuver decision-making, poor algorithm performance may result from sample exploration and utilization inefficiency. To address this issue, this paper proposes an expert experience storage mechanism that can effectively enhance algorithm performance with less experience replay time by utilizing successful expert experience in UAV air games. Based on the Dueling Double Deep Q Network and expert experience storage mechanism, a UAV air game maneuver decision-making model is established in this study, and its effectiveness is verified through simulation experiments. The results demonstrate that the proposed expert experience storage mechanism can improve the algorithm's performance with less experience replay time. Furthermore, the proposed maneuver decision-making algorithm is effective in providing decisions to strike target UAVs employing different fixed strategies.

In future work, an analysis of the algorithm's computational complexity and inference speed will be conducted with respect to the limited computational load of UAVs, and the aircraft will be modeled in detail on a semi-physical simulation platform to simulate the real flight conditions of the UAV.

**Author Contributions:** Conceptualization, J.Z. and Z.M.; methodology, J.Z.; software, J.Z. and J.H.; validation, J.Z. and Z.W.; writing—original draft preparation, J.Z.; writing—review and editing, J.Z. and L.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Natural Science Foundation (NSF) of China (No. 61976014). This work was also supported by the Fundamental Research Funds for the Central Universities.

**Data Availability Statement:** Not applicable.

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

**Sample Availability:** Samples of the compounds ... are available from the authors.

## References

- Kong, W.; Zhou, D.; Yang, Z.; Zhao, Y.; Zhang, K. UAV autonomous aerial combat maneuver strategy generation with observation error based on state-adversarial deep deterministic policy gradient and inverse reinforcement learning. *Electronics* **2020**, *9*, 1121. [[CrossRef](#)]
- 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* **2019**, *8*, 363–378. [[CrossRef](#)]
- Myerson, R.B. *Game Theory: Analysis of Conflict*; Harvard University Press: Cambridge, MA, USA, 1997.
- Shachter, R.D. Evaluating influence diagrams. *Oper. Res.* **1986**, *34*, 871–882. [[CrossRef](#)]
- Chen, X.; Li, Q.; Sun, Y. Key technologies for air combat intelligent decision based on game confrontation. *Command. Inf. Syst. Technol.* **2021**, *12*, 1–6.
- Virtanen, K.; Karelähti, J.; Raivio, T. Modeling air combat by a moving horizon influence diagram game. *J. Guid. Control. Dyn.* **2006**, *29*, 1080–1091. [[CrossRef](#)]
- Lin, Z.; Ming'an, T.; Wei, Z.; Shengyun, Z. Sequential maneuvering decisions based on multi-stage influence diagram in air combat. *J. Syst. Eng. Electron.* **2007**, *18*, 551–555. [[CrossRef](#)]
- Pan, Q.; Zhou, D.; Huang, J.; Lv, X.; Yang, Z.; Zhang, K.; Li, X. Maneuver decision for cooperative close-range air combat based on state predicted influence diagram. In Proceedings of the 2017 IEEE International Conference on Information and Automation (ICIA), Macau, China, 18–20 July 2017; pp. 726–731.
- Xie, R.Z.; Li, J.Y.; Luo, D.L. Research on maneuvering decisions for multi-UAVs Air combat. In Proceedings of the 11th IEEE International Conference on Control & Automation (ICCA), Taichung, Taiwan, 18–20 June 2014; pp. 767–772.
- Weintraub, I.E.; Pachter, M.; Garcia, E. An introduction to pursuit-evasion differential games. In Proceedings of the 2020 American Control Conference (ACC), Online, 1–3 July 2020; pp. 1049–1066.
- Mukai, H.; Tanikawa, A.; Tunay, I.; Ozcan, I.; Katz, I.; Schättler, H.; Rinaldi, P.; Wang, G.; Yang, L.; Sawada, Y. Sequential linear-quadratic method for differential games with air combat applications. *Comput. Optim. Appl.* **2003**, *25*, 193–222. [[CrossRef](#)]
- Fu, L.; Wang, X. The short-range dogfight combat model of modern fighter based on differential games. In Proceedings of the 2011 Chinese Control and Decision Conference (CCDC), Mianyang, China, 23–25 May 2011; pp. 4082–4084.
- Park, H.; Lee, B.Y.; Tahk, M.J.; Yoo, D.W. Differential game based air combat maneuver generation using scoring function matrix. *Int. J. Aeronaut. Space Sci.* **2016**, *17*, 204–213. [[CrossRef](#)]
- Başpınar, B.; Koyuncu, E. Assessment of aerial combat game via optimization-based receding horizon control. *IEEE Access* **2020**, *8*, 35853–35863. [[CrossRef](#)]
- He, M.; Wang, X. Nonlinear Differential Game Guidance Law for Guarding a Target. In Proceedings of the 2020 6th International Conference on Control, Automation and Robotics (ICCAR), Singapore, 20–23 April 2020; pp. 713–721.
- Smith, R.E.; Dike, B.; Mehra, R.; Ravichandran, B.; El-Fallah, A. Classifier systems in combat: two-sided learning of maneuvers for advanced fighter aircraft. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 421–437. [[CrossRef](#)]
- Sprinkle, J.; Eklund, J.M.; Kim, H.J.; Sastry, S. Encoding aerial pursuit/evasion games with fixed wing aircraft into a nonlinear model predictive tracking controller. In Proceedings of the 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601), Nassau, The Bahamas, 14–17 December 2004; Volume 3, pp. 2609–2614.
- 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](#)]
- Duan, H.; Li, P.; Yu, Y. A predator-prey particle swarm optimization approach to multipleUCAV air combat modeled by dynamic game theory. *IEEE/CAA J. Autom. Sin.* **2015**, *2*, 11–18.
- Ji, H.; Yu, M.; Han, Q.; Yang, J. Research on the air combat countermeasure generation based on improved TIMS model. *IOP J. Phys. Conf. Ser.* **2018**, *1069*, 012039. [[CrossRef](#)]
- Han, T.; Wang, X.; Liang, Y.; Ku, S. Study onUCAV robust maneuvering decision in automatic air combat based on target accessible domain. *IOP J. Phys. Conf. Ser.* **2019**, *1213*, 052004. [[CrossRef](#)]
- Tan, M.; Tang, A.; Ding, D.; Xie, L.; Huang, C. Autonomous Air Combat Maneuvering Decision Method ofUCAV Based on LSHADE-TSO-MPC under Enemy Trajectory Prediction. *Electronics* **2022**, *11*, 3383. [[CrossRef](#)]

23. Ruan, W.; Duan, H.; Deng, Y. Autonomous Maneuver Decisions via Transfer Learning Pigeon-Inspired Optimization for UCAVs in Dogfight Engagements. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 1639–1657. [[CrossRef](#)]
24. Machmudah, A.; Shanmugavel, M.; Parman, S.; Manan, T.S.A.; Dutykh, D.; Beddu, S.; Rajabi, A. Flight trajectories optimization of fixed-wing UAV by bank-turn mechanism. *Drones* **2022**, *6*, 69. [[CrossRef](#)]
25. Rodin, E.Y.; Amin, S.M. Maneuver prediction in air combat via artificial neural networks. *Comput. Math. Appl.* **1992**, *24*, 95–112. [[CrossRef](#)]
26. Schvaneveldt, R.W.; Goldsmith, T.E.; Benson, A.E.; Waag, W.L. *Neural Network Models of Air Combat Maneuvering*; Technical Report; New Mexico State University: Las Cruces, NM, USA, 1992.
27. Teng, T.H.; Tan, A.H.; Tan, Y.S.; Yeo, A. Self-organizing neural networks for learning air combat maneuvers. In Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, 10–15 June 2012; pp. 1–8.
28. Geng, W.X.; Ma, D.Q. Study on tactical decision of UAV medium-range air combat. In Proceedings of the 26th Chinese Control and Decision Conference (2014 CCDC), Changsha, China, 31 May–2 June 2014; pp. 135–139.
29. 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](#)]
30. Çintaş, E.; Özyer, B.; Şimşek, E. Vision-based moving UAV tracking by another UAV on low-cost hardware and a new ground control station. *IEEE Access* **2020**, *8*, 194601–194611. [[CrossRef](#)]
31. 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](#)]
32. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
33. Wan, K.; Gao, X.; Hu, Z.; Wu, G. Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning. *Remote Sens.* **2020**, *12*, 640. [[CrossRef](#)]
34. Yang, K.; Dong, W.; Cai, M.; Jia, S.; Liu, R. UCAV air combat maneuver decisions based on a proximal policy optimization algorithm with situation reward shaping. *Electronics* **2022**, *11*, 2602. [[CrossRef](#)]
35. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
36. Zhang, H.; Wei, Y.; Zhou, H.; Huang, C. Maneuver Decision-Making for Autonomous Air Combat Based on FRE-PPO. *Appl. Sci.* **2022**, *12*, 10230. [[CrossRef](#)]
37. Fan, Z.; Xu, Y.; Kang, Y.; Luo, D. Air Combat Maneuver Decision Method Based on A3C Deep Reinforcement Learning. *Machines* **2022**, *10*, 1033. [[CrossRef](#)]
38. 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](#)]
39. Li, B.; Huang, J.; Bai, S.; Gan, Z.; Liang, S.; Evgeny, N.; Yao, S. Autonomous air combat decision-making of UAV based on parallel self-play reinforcement learning. *CAAI Trans. Intell. Technol.* **2022**, *8*, 64–81. [[CrossRef](#)]
40. 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**. [[CrossRef](#)]
41. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
42. Austin, F.; Carbone, G.; Falco, M.; Hinz, H.; Lewis, M. Automated maneuvering decisions for air-to-air combat. In Proceedings of the Guidance, Navigation and Control Conference, Monterey, CA, USA, 17–19 August 1987; p. 2393.
43. 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.
44. Fedus, W.; Ramachandran, P.; Agarwal, R.; Bengio, Y.; Laroche, H.; Rowland, M.; Dabney, W. Revisiting fundamentals of experience replay. In Proceedings of the International Conference on Machine Learning, PMLR, Online, 13–18 July 2020; pp. 3061–3071.
45. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
46. Karimpanal, T.G.; Bouffanais, R. Experience replay using transition sequences. *Front. Neurobot.* **2018**, *12*, 32. [[CrossRef](#)] [[PubMed](#)]
47. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
48. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 1995–2003.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.