

Article

Towards Intelligent Unmanned Adversarial Games: A Reinforcement Learning Framework with the PHP-ROW Method

Guoqing Shi , Yi Cao, Dinghan Wang , Qiming Yang, Jiandong Zhang  and Zhuoyong Shi 

School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China; shiguqing@nwpu.edu.cn (G.S.); caoyi@mail.nwpu.edu.cn (Y.C.); yangqm@nwpu.edu.cn (Q.Y.); shizy@mail.nwpu.edu.cn (Z.S.)

* Correspondence: 2018301718@mail.nwpu.edu.cn (D.W.); jdzhang@nwpu.edu.cn (J.Z.)

Abstract: This study introduces a novel framework for intelligent unmanned BVR maneuver control within the context of adversarial games. The emphasis lies on three pivotal aspects: situational awareness, maneuver decision-making, and precise maneuver control. Within this paradigm, our unmanned aerial vehicles (UAVs) can assimilate crucial situational information through constructed situational vectors and execute sophisticated maneuvers, effectively addressing the intricacies of dynamic flight environments and various unpredictable scenarios within the game setting. To achieve granular maneuver control, this research introduces the Priority Heading Polling–Random Observation Weight (PHP-ROW) method, underpinned by deep reinforcement learning. This approach integrates two primary components: (1) the priority heading polling (PHP) mechanism, which governs the extent of flight trajectories while emphasizing heading control, and (2) the random observation weight (ROW) technique, which adeptly moderates the influence of roll angle rewards during the learning phase. The superiority of the PHP-ROW method is showcased by contrasting it against the conventional proximal policy optimization (PPO) algorithm. Conclusively, the utility and efficacy of the presented framework are corroborated through human–machine adversarial game simulations in a hyper-realistic environment. This investigation provides foundational theoretical and empirical contributions to the realm of intelligent unmanned aerial maneuver control, promising significant implications for the evolution of aviation technology in adversarial game contexts.

Keywords: artificial intelligence; deep reinforcement learning; Bayes network; priority heading polling; random observation weight; UAV; BVR



Academic Editor: Abdessattar Abdelkefi

Received: 5 March 2025

Revised: 15 April 2025

Accepted: 22 April 2025

Published: 24 April 2025

Citation: Shi, G.; Cao, Y.; Wang, D.; Yang, Q.; Zhang, J.; Shi, Z. Towards Intelligent Unmanned Adversarial Games: A Reinforcement Learning Framework with the PHP-ROW Method. *Drones* **2025**, *9*, 331. <https://doi.org/10.3390/drones9050331>

Copyright: © 2025 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

Traditional aerial games challenge the piloting skills of players, especially in close-range scenarios where quick reactions are paramount. With the continuous advancement of unmanned systems technology and information technology, unmanned aerial vehicles (UAVs) are increasingly utilized in various tasks, from data collection to simulated challenges [1]. They offer advantages such as cost-effectiveness, high maneuverability, and advanced stealth capabilities. Furthermore, the extensive use of unmanned and intelligent systems in various real-world applications underscores the future direction of aerial operations [2–4]. The next evolution of aerial games will likely be dominated by unmanned, autonomous, intelligent, and beyond-visual-range capabilities [5].

The core challenge of intelligent beyond-visual-range UAV aerial games revolves around the development of autonomous aerial control (AAC) capabilities. These capabilities enable UAVs to operate effectively without human intervention, achieving situational awareness through environmental sensing and analysis. However, UAVs face several key obstacles, including difficulties in reliably detecting and interpreting complex scenarios, making real-time maneuvering decisions in dynamic environments, and executing precise control commands in highly unstable flight conditions. To address these issues, our approach focuses on enhancing UAVs' ability to autonomously assess their surroundings, process real-time data, and adapt their flight strategy accordingly. Specifically, we utilize reinforcement learning algorithms to improve decision-making and control performance, ensuring that UAVs can autonomously navigate and perform precise maneuvers. This enables UAVs to handle the challenges posed by beyond-visual-range aerial games, where rapid, accurate decision-making and control are critical.

Research methodologies related to autonomous aerial maneuver decision-making fall into three primary categories: mathematical modeling, computer-based traversal search, and intelligent algorithms [6].

Mathematical modeling methods frame typical aerial game tasks as mathematical optimization problems. The literature [7–9] adopts the homicidal chauffeur problem as a representative model. However, this approach comes with constraints like planar and constant speed conditions, limiting its practical applicability. The literature [10,11] delves into pursuit–evasion (PE) scenarios, with predefined roles for pursuer and evader. Such models may not always resonate with modern aerial games where roles can interchange dynamically. The literature [12] also examines multi-objective optimization problems, determining game outcomes based on flight data. These methods often grapple with limited applicability and challenges in deriving analytical solutions.

Computer-based traversal search methodologies discretize potential maneuver control commands, calculating optimal decisions based on situational awareness. This strategy is prevalent in decision-making processes. The literature [13–16] investigates the A* algorithm and its variants for pathfinding and obstacle navigation. Other literature [17,18] delves into the influence map method, offering a clear representation of key aerial game factors, albeit at the cost of computational complexity. Studies [19,20] explore adaptive dynamic programming (ADP), which, through data-driven techniques, determines search intervals for commands. Its practical application remains challenging due to inherent constraints.

Intelligent algorithms transform situational awareness data into maneuver decisions, employing operations like feature extraction, clustering, and fitting on extensive datasets. This category encompasses neural networks [21], fuzzy logic [22], genetic algorithms [23], reinforcement learning [24], Bayesian networks [25], and other intelligent algorithms [26,27], where neural networks offer powerful function approximation capabilities, making them suitable for modeling complex nonlinear dynamics. However, they typically require large labeled datasets and may lack interpretability. Fuzzy logic systems excel at incorporating human-like reasoning into control systems and are effective in dealing with uncertainties, but they struggle with high-dimensional input spaces and require expert-designed rule sets. Genetic algorithms are useful for global optimization and parameter tuning in UAV control strategies, yet they are often computationally expensive and slow to converge in real-time applications. Bayesian networks provide a probabilistic framework for reasoning under uncertainty and are particularly effective in tasks involving causal inference; however, they require well-defined prior knowledge and can become intractable in complex domains. Reinforcement learning stands out by enabling agents to learn optimal control strategies directly from interaction with the environment, without requiring explicit modeling of dynamics. RL is especially well-suited for scenarios with delayed rewards and sparse

feedback, making it a focal point for many researchers in the field of autonomous UAV control. Notably, reinforcement learning research is a focal point for many experts.

Reinforcement learning is a branch of machine learning focused on how agents learn to make decisions by interacting with an environment to achieve a specific goal. Unlike supervised learning, which relies on labeled data, RL agents learn from trial and error, receiving feedback in the form of rewards.

At the core of reinforcement learning are two entities: an agent and an environment. The agent perceives the current state of the environment, selects an action based on a policy π , and receives a reward as feedback. This process repeats over time, forming a sequence of interactions that continues until the task (or episode) is completed. This loop follows the Markov decision process (MDP) framework, as illustrated in Figure 1. Through continuous interaction, the agent uses the collected data—states, actions, and rewards—to improve its policy, aiming to discover an optimal strategy that maximizes long-term cumulative rewards.

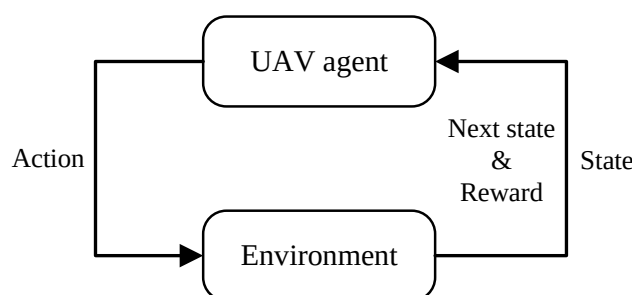


Figure 1. Markov decision process (MDP).

Deep reinforcement learning is an advanced framework that combines the decision-making capabilities of reinforcement learning with the powerful representation learning abilities of deep neural networks. While traditional reinforcement learning struggles in high-dimensional or continuous state spaces due to the need for explicit state–action mappings, DRL overcomes these limitations by using neural networks to approximate value functions, policies, or both.

DRL methods can be broadly categorized into value-based methods and policy-based methods, where value-based methods (e.g., deep Q-networks) aim to learn the value of taking certain actions in given states and select actions by maximizing estimated values. Policy-based methods directly optimize the policy that maps states to actions, which can better handle continuous action spaces and stochastic policies.

This integration allows DRL to effectively handle tasks with large or continuous state and action spaces, improve convergence stability, and enable more scalable and generalizable learning across complex environments.

The literature [28,29] delves into close-range aerial game maneuver decision-making using deep reinforcement learning methods, yet it focuses solely on a three-degree-of-freedom aircraft model, diverging from the more realistic six-degrees-of-freedom model employed in advanced aerial games. Furthermore, close-quarter engagements are becoming less predominant in contemporary aerial games, with beyond-visual-range engagements becoming more central. Other literature [30,31] adopts a hierarchical reinforcement learning strategy, crafting a semi-Markov decision model and training sub-policies and policy selectors to achieve autonomous UAV maneuver decision-making. This method, however, necessitates meticulous reward function design for each maneuver, complicating the reward structuring for intricate maneuvers. Moreover, it is mainly tailored for theoretical maneuver decision-making research and poses challenges for precise control. Study [32] presents an end-to-end reinforcement learning technique for beyond-visual-range aerial

game decision-making, yet confronts issues like sparse rewards, prolonged training duration, and convergence obstacles. Studies [33,34] employ a two-dimensional environment for its simulations, proving to be a hurdle in mimicking genuine game scenarios. Study [35] establishes a high-accuracy model grounded on actual UAV metrics, examines reinforcement learning strategies for dynamic target tracking, and introduces the IFHER method to transmute failed flight paths into triumphant experiences, enhancing trajectory efficiency and expediting training convergence. However, dynamic target tracking is merely a secondary objective in beyond-visual-range aerial games. Other literature [36,37] proposes a deep reinforcement learning-oriented method for autonomous UAV evasion in game settings. Still, its specificity to certain evasion models makes it less adaptable to parameter alterations. Study [38] explores UAV waypoint navigation control using the DDPG algorithm and emulates chase-evade scenarios. Nevertheless, this method curtails roll and pitch angles during training to guarantee convergence and stability, making it apt only for waypoint navigation and not ideal for intricate and agile game environments.

In summary, intelligent unmanned beyond-visual-range aerial games face the following three main challenges:

1. **Simulation Fidelity Issue:** Much of the current research employing deep reinforcement learning algorithms is limited to simplified 2D environments or three degrees of freedom, resulting in diminished simulation precision.
2. **Research Completeness Issue:** There exists a gap in comprehensive and in-depth studies in the realm of beyond-visual-range (BVR) intelligent games.
3. **Reward Function Design Challenge:** Crafting a fitting reward function for intricate maneuvers using deep reinforcement learning presents a significant hurdle.

This paper introduces a framework for intelligent unmanned beyond-visual-range aerial games with the following primary contributions:

1. **Addressing the Simulation Fidelity Issue:** We employed the deep reinforcement learning PPO method for maneuver control and developed a 6-DOF high-fidelity UAV model and simulation environment. To tackle the intricacy of reward function design, we ushered in the PHP-ROW training method. This approach expedites UAV control, achieving the intended heading, altitude, and speed, thereby formulating action commands. It not only boosts interpretability but also smoothes the transition from simulations to real-world UAVs.
2. **Enhancing Research Completeness:** A rigorous examination of the pivotal components in BVR aerial games was undertaken, leading to the formulation of a key situational vector and a performance function, constituting the situational awareness module in our framework.
3. **Introducing a Novel Reward Function Design:** We designed six maneuvers—directional navigation, altitude adjustment, objective targeting, trajectory optimization, evasive action, and agile turns. Utilizing a Bayesian network structure and expert dataset-based training, we achieved inference from situational data to maneuver actions.
4. **A Holistic BVR Game Workflow:** By amalgamating situational awareness, maneuver decision-making, and precise control, we accomplished a comprehensive BVR game sequence. The efficiency of our framework was subsequently ascertained through human-machine BVR game simulations.

The subsequent sections of this paper are structured as follows: Section 2 outlines the problem, encompassing the six-degrees-of-freedom UAV dynamics model and the creation of situational awareness vectors. Section 3 details the PHP-ROW method, delineating the design of the deep reinforcement learning-based maneuver control algorithm.

Section 4 elucidates the entire process and methodologies of beyond-visual-range UAV aerial games, spanning situational awareness, maneuver decision-making, and precise control. Section 5 curates waypoint navigation tasks and human–machine BVR game scenarios, executing experiments and evaluating outcomes. Section 6 concludes the paper, broaching potential avenues for future exploration.

2. Problem Formulation

In this section, the key concepts and problem models are introduced: Section 2.1 establishes a six-degrees-of-freedom dynamics model to simulate the motion state of UAVs in the game environment, Section 2.2 designs basic situational vectors and performance functions to establish an accurate situational awareness model, and Section 2.3 implements autonomous situational awareness and tactical decision-making through the combination of fuzzy methods and Bayesian network inference.

2.1. UAV Dynamic Model

In the study of high-precision unmanned aerial vehicle (UAV) simulations for beyond-visual-range aerial games, a primary task is to construct a detailed six-degrees-of-freedom dynamics model to simulate the UAV's motion behavior in various in-game scenarios. This model should encompass the UAV's movement in six degrees of freedom: three translational degrees of freedom (displacement along the X, Y, and Z axes) and three rotational degrees of freedom (roll, pitch, and yaw), as depicted in Figure 2.

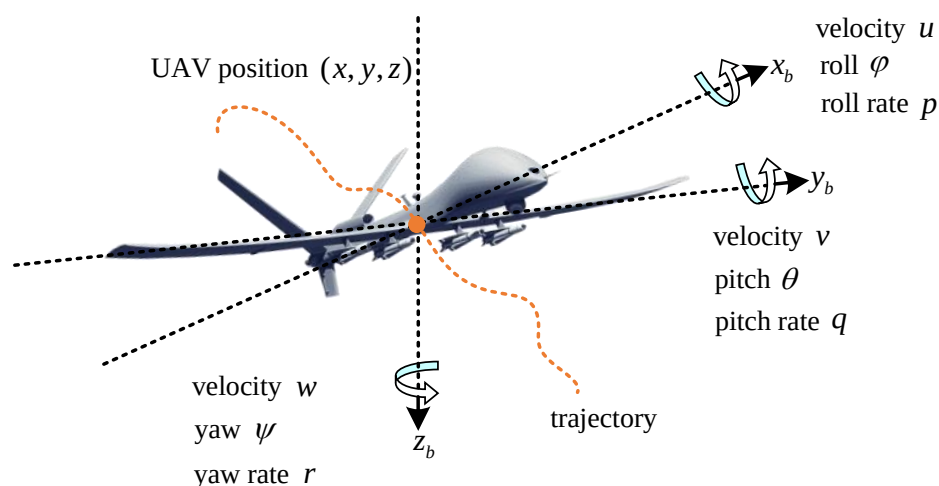


Figure 2. The 6-DOF UAV model.

To accurately simulate the UAV's dynamic behavior within aerial game scenarios, it is essential to consider the aircraft's physical characteristics, propulsion system, aerodynamic performance, and control system. First, a point-mass dynamics model for the aircraft will be established, describing the motion of a point mass in six degrees of freedom. Following this, the aircraft's rotational motion, encompassing roll, pitch, and yaw, will be taken into account. This step requires considering the aircraft's moment of inertia matrix and the torques produced to induce such rotational movements. The aircraft's propulsion system, which includes engine thrust and torques, will be integrated, along with the impact of aerodynamic forces and moments.

Ultimately, we aim to develop a comprehensive six-degrees-of-freedom dynamics model that can simulate the UAV's maneuvers in various in-game scenarios. This model will facilitate high-fidelity UAV flight testing within a simulation environment, evaluating its performance in beyond-visual-range aerial games.

The body-fixed coordinate system acceleration is defined as follows:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \frac{1}{m} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} + C_g^b \begin{bmatrix} 0 \\ 0 \\ G \end{bmatrix} \quad (1)$$

where u, v, w represent the velocities along the body-fixed coordinate system's x, y, z axes, p, q, r represent the angular velocities along the x, y, z axes, F_x, F_y, F_z represent the external forces acting along the x, y, z axes, m denotes the mass of the aircraft, G denotes the acceleration due to gravity, and C_g^b represents the transformation matrix from the ground coordinate system to the body-fixed coordinate system. The body-fixed coordinate system angular acceleration is defined as follows:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} \left(- \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} \right) \quad (2)$$

where I represents the moment of inertia of the aircraft, and M_x, M_y, M_z denote the external moments acting along the body-fixed coordinate system's x, y, z axes, respectively. Integration yields the body-fixed coordinate system velocities and angular velocities, shown as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \int \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} dt \quad (3)$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \int \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} dt \quad (4)$$

Converting body-fixed coordinate system angular velocities to Euler angular velocities can be expressed as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = C_b^e \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (5)$$

where ϕ, θ, ψ represent the roll, pitch, and yaw angles of the body-fixed coordinate system relative to the ground coordinate system, and C_b^e represents the transformation matrix from body-fixed coordinate system angular velocities to Euler angular velocities.

Converting body-fixed coordinate system velocities to ground coordinate system velocities can be defined as follows:

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = (C_g^b)^T \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (6)$$

where V_x, V_y, V_z represent the velocities along the ground coordinate system's x, y, z axes, respectively.

Integrating ground coordinate system velocities yields the aircraft's position in the ground coordinate system, whereas integrating Euler angular velocities provides the Euler angles, shown as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \int \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} dt \quad (7)$$

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \int \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} dt \quad (8)$$

2.2. Situational Awareness Model

In beyond-visual-range aerial games, precise situational awareness and an in-depth grasp of the game environment are vital. To meet this goal, we design essential situational vectors and performance functions. These vectors and functions guide unmanned aircraft in understanding the current state of the game and offer pivotal data for intelligent decision-making.

We outline and create these key situational vectors, encompassing crucial details like the positions, velocities, altitudes, and orientations of the unmanned aircraft and other game entities. Specifically, this scenario is designed to simulate a typical airspace encounter where two UAVs—red (the ego UAV) and blue (the target or intruder UAV)—interact in a shared environment. The setup captures key relative motion parameters: distance difference ΔD , altitude difference ΔH , relative heading angles ϕ_A and ϕ_T , and velocities V_A and V_T . These variables are commonly used in both air combat and collision avoidance contexts. These variables will thoroughly portray the dynamic situation within the game and endow the unmanned aircraft with a complete understanding of the game environment, as illustrated in Figure 3.

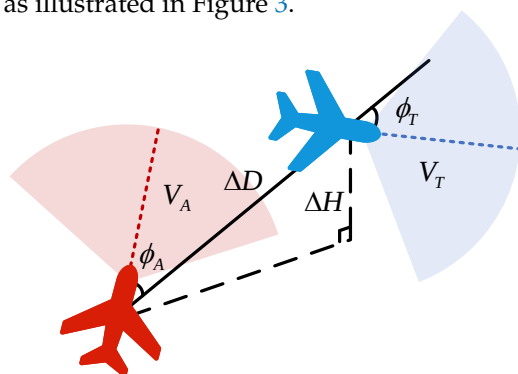


Figure 3. Situation between red and blue UAVs.

To achieve comprehensive game situational awareness and facilitate autonomous beyond-visual-range operations for unmanned aerial vehicles (UAVs) within our aerial game environment, we have pinpointed key situational elements to formulate a performance function. The situational data include:

1. Binary Variables: alert signal W , target identification signal S , guidance signal G , operational boundary effectiveness F , and enemy entity action trigger L . These variables can only assume values of 0 or 1.
2. Performance Variables: proximity score S_D , altitude score S_H , velocity score S_V , and angular score S_A are the four performance variables. We normalize the values of these performance variables within the range of $[-1, 1]$.

Proximity score S_D and altitude score S_H are used to quantify the distance between our aircraft and the target, with a lower value indicating closer spatial proximity between the two.

$$S_D = \frac{\Delta D}{100} \quad (9)$$

$$S_H = \frac{\Delta H}{10000} \quad (10)$$

Velocity score S_V describes the difference in velocity between two aircraft and is used to quantify the dynamic relative state between them. These scores can effectively assess potential threats in air combat and provide a scientific basis for tactical decision-making.

$$S_V = \frac{(V_T - V_A)}{340} \quad (11)$$

Angular score S_A describes the angular difference between two aircraft, and the relative heading angle is usually in the range $[-180^\circ, 180^\circ]$, expressed by $\Delta\phi$, so the angular score can be expressed as

$$\Delta\phi = \begin{cases} \phi_T - \phi_A - 360^\circ, & \text{if } \phi_T - \phi_A > 180^\circ \\ \phi_T - \phi_A + 360^\circ, & \text{if } \phi_T - \phi_A < -180^\circ \\ \phi_T - \phi_A, & \text{otherwise} \end{cases} \quad (12)$$

$$S_A = \frac{\Delta\phi}{180} \quad (13)$$

These situational factors are pivotal for informed decision-making in autonomous beyond-visual-range UAV game scenarios, facilitating dynamic evaluation and adaptation to evolving conditions within the game environment.

2.3. Maneuver Control Decision Model

In this section, we introduce six long-range game maneuver actions and break them down based on three criteria: desired altitude, desired velocity, and desired heading. Next, we design the Bayesian network topology and utilize a fuzzification method for data processing to simplify the data complexity. Finally, we detail the training and inference process of the Bayesian network, achieving the transformation from the game environment to the optimal maneuver decision.

2.3.1. Game Maneuver Design

In this section, we design six game maneuver actions and conditionally decompose them based on the desired heading, desired altitude, and desired velocity. The game maneuver actions and their decomposition are presented in Table 1.

Table 1. Six tactical maneuver designs.

Tactical Action	Desired Heading (Rad)	Desired Altitude (m)	Desired Velocity (M)
Directional Attack	P_{az}	2000	0.59
Climbing Expansion	T_{az}	4000	0.59
Tail Escape	$M_{az} + \pi$	2000	0.59
Offset Guidance	$G_{az} + \frac{\pi}{6}$	2000	0.59
S Maneuver	C_{az}	2000	0.59
Weapon Launch	-	-	-

1. Directional Attack Maneuver: This maneuver has relatively low maneuverability requirements. It involves directing the unmanned aircraft towards a specific direction based on provided point penetration location information to complete a directional attack mission.
2. Climbing Expansion Maneuver: This maneuver has moderate maneuverability requirements. It involves flying towards a target based on radar-detected target azimuth information while climbing in altitude. This helps improve the weapon's attack envelope.

3. Tail Escape Maneuver: This maneuver has high maneuverability requirements. It demands the unmanned aircraft to rapidly change direction to evade incoming missiles.
4. Offset Guidance Maneuver: This maneuver has relatively low maneuverability requirements. It requires the radar to simultaneously illuminate both the missile and the target, placing the target at the edge of the radar's detection range.
5. S Maneuver: The maneuver's maneuverability requirements depend on the specific situation. Performing large-angle turns can also be considered a high-mobility action. S maneuver is typically used for tasks such as searching for targets within a specific range and depleting the energy of pursuing missiles. Another characteristic of the S maneuver is the turn and heading-holding times. For long-duration maneuvers and large turn radii, it can be divided into multiple heading commands at a specific granularity, forming the S maneuver.
6. Weapon Launch Maneuver: This maneuver requires maintaining heading, altitude, and speed. The key to this maneuver is the timing of the decision to launch weapons.

In Table 1, P_{az} represents the geographical azimuth for point penetration, T_{az} denotes the azimuth angle for radar-detected targets, M_{az} indicates the geographical azimuth of incoming missiles, G_{az} signifies the geographical azimuth for guided missiles, and C_{az} represents the desired heading for the turning control of the S maneuver. We have set the default values for desired velocity at 0.59 m/s and desired altitude at 2000/4000 km, but these values can be adjusted as needed.

2.3.2. Bayesian Formula and Bayesian Network

Unlike the traditional frequentist approach, Bayesianism places greater emphasis on prior information, asserting that parameters are not fixed and immutable, but instead possess a prior distribution based on past observational data. This prior distribution is not necessarily accurate and can be updated based on subsequent observational data, yielding the posterior distribution of parameters. This approach enables a more accurate and comprehensive understanding of the parameters, shown as

$$P(\theta|X = x_i) = \frac{P(X = x_i|\theta)P(\theta)}{P(X = x_i)} \quad (14)$$

where $P(\theta)$ represents the prior probability of the parameter θ , encapsulating our initial beliefs or knowledge about θ before any data observations; $P(X|\theta)$ denotes the likelihood function, indicating the probability of observing data x_i given a specific value of the parameter θ ; $P(X = x_i)$ is a fixed constant representing the probability of observing the specific data point x_i ; and, finally, $P(\theta|X = x_i)$ quantifies the posterior probability of the parameter θ after observing the sample $X = x_i$, reflecting our updated beliefs regarding θ based on the observed data.

Bayesian networks are graphical models depicting probabilistic relationships among a set of variables. They have the capability to capture relationships between variables and can update beliefs about the target variable based on new data. Bayesian networks are typically represented as

$$N = (G, D) \quad (15)$$

where $G = (V, E)$ is a directed acyclic graph (DAG) with nodes V and directed edges E ; D represents a set of local probability distributions associated with each variable in V .

According to the Markov assumption, we can combine the conditional distributions of individual variables to obtain the joint probability distribution of V , shown as

$$P(V = v) = \prod_{x_i \in v} P(x_i|u_X) \quad (16)$$

where $U_X \subset V$ represents the parent nodes of $X \in V$, and $P(x_i|u_X)$ represents the local probability of variable X , which can be obtained from D . Consequently, (G, D) uniquely determines the joint probability distribution of V .

2.3.3. Tactical Maneuver Inference

In the preceding sections, we have constructed critical situational information. However, when the unmanned aircraft perceives situational data, it needs to autonomously decide which tactical maneuver to take. We design a Bayesian network topology based on the constructed situational information and the available tactical actions. To obtain the conditional probability distribution tables for the Bayesian network, we train the Bayesian network using an expert experience dataset. Ultimately, we accomplish autonomous situational awareness and tactical decision-making through a combination of fuzzy methods and Bayesian network inference.

We assume that the situational vector and advantageous variables are mutually independent and adopt a head-to-head Bayesian network topology, with the situational vector and advantageous variables serving as parent nodes of the Bayesian network, and the decision variable as the unique shared child node. To simplify the problem, we employ a discrete Bayesian network, necessitating the fuzzification of continuous advantageous variables. We assume that S_A, S_H, S_V can be fuzzified as P, Z, N using Gaussian membership functions, and S_D can be fuzzified as L, S using Gaussian membership functions. Additionally, W, S, G, V, F can be fuzzified as Y, N using triangular membership functions.

Next, we need to determine the conditional probability distribution tables (CPD) for this Bayesian network. CPDs can be filled in by experts or learned from data. We have constructed a training dataset for network training based on situational information obtained from simulated adversarial experiments and the corresponding maneuver actions. The training process is illustrated in Figure 4a.

Data learning methods include maximum likelihood estimation (MLE) and Bayesian estimation. In this paper, we adopt the Bayesian estimation method. We assume a Dirichlet distribution as the prior distribution for the decision variable nodes. Our decision variable nodes encompass the six maneuver variables. Therefore, we define the distribution parameters and their prior distribution as

$$\theta_i = P(X = x_i), i \in \{0, 1, \dots, 5\} \quad (17)$$

$$P(\theta) = \frac{\Gamma(\alpha)}{\prod_{i=0}^5 \Gamma(\alpha_i)} \prod_{i=0}^5 \theta_i^{\alpha_i-1} \quad (18)$$

Because the Dirichlet distribution is a conjugate prior for the multinomial distribution, we can utilize the collected data as samples and employ Bayesian estimation to learn the parameters. When new situational information becomes available, we can use it as posterior information to estimate the posterior probabilities of the decision variable nodes, serving as the basis for our tactical decisions. Here, we use the exact inference variable elimination method. The advantage of this method is that it can provide decision probabilities based on partial situational information. It allows us to focus on partial situational information and still reliably perform decision reasoning when only partial situational information is available. The network inference process is illustrated in Figure 4b.

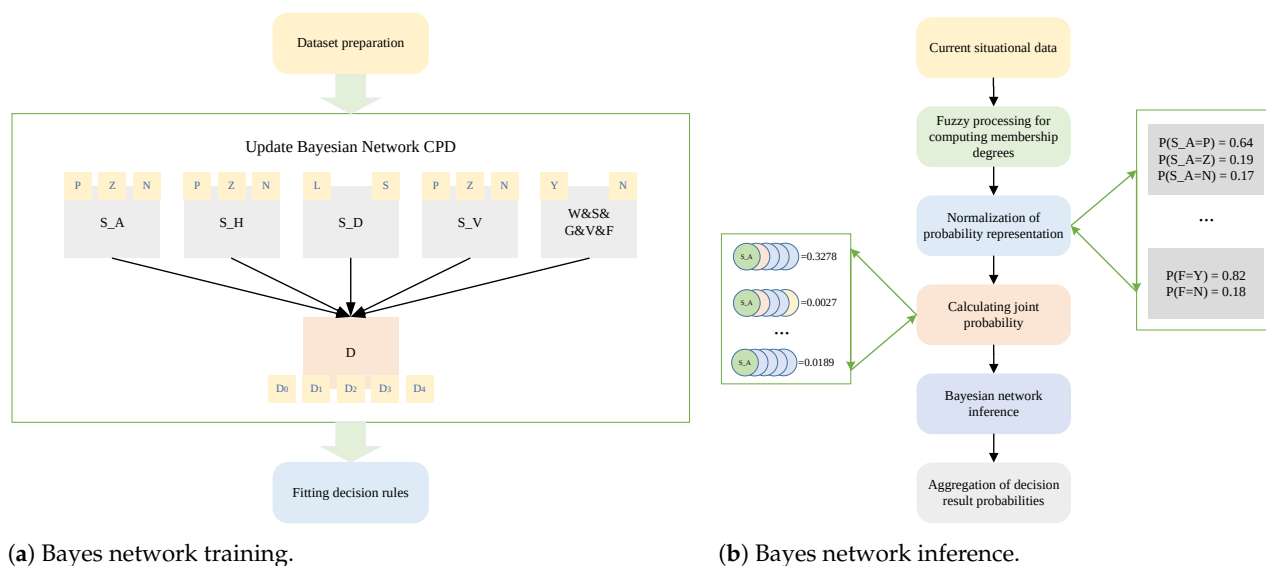


Figure 4. Bayes network process structure.

3. PHP-ROW Method and PPO Algorithm for Tactical Control

In this section, the proposed algorithm framework is introduced. First, the priority heading polling (PHP) and the random observation weight (ROW) methods are introduced. Second, the implementation of PPO-clip algorithm, and a reasonable state-action space, reward function, and termination condition are designed according to the UAV adversarial environment, and finally, a complete training architecture is established.

3.1. PHP-ROW Method

To ensure both flight stability and high maneuverability simultaneously, we have introduced a training framework that incorporates the priority heading polling (PHP) method and the random observation weight (ROW) method. The detailed description of this framework is as follows:

1. **Priority Heading Polling (PHP):** Within every 250 simulated time steps, we check whether the unmanned aircraft has reached the desired heading. If the desired heading is achieved, we set new target values for heading, altitude, and velocity, selecting these new targets randomly within a predefined range. If the desired heading is not reached, the current round is terminated. The advantages of this method include:
 - (a) By employing the priority heading polling, we can control the length of the unmanned aircraft's flight trajectory during the early stages, thus promoting exploration.
 - (b) We sample from a uniform distribution $[0, 1]$ and use the sampled value as the weight for the roll reward function. We keep the random observation weight unchanged and only resample it at the end of each episode. This limits the impact of the roll reward on the training process and helps balance the trade-off between rapid turning and flight stability.
2. **Random Observation Weight (ROW):** We sample from a uniform distribution within the range $[0, 1]$ and utilize the sampled values as weights for the roll angle reward function. This effectively constrains the influence of roll angle rewards on the training process, helping to balance the trade-off between rapid maneuvering and flight stability.

The architecture of the proposed training framework is depicted in Figure 5.

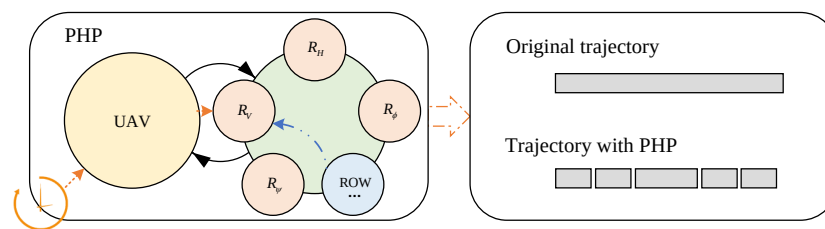


Figure 5. PHP-ROW method.

3.2. PPO-Clip Algorithm

The PPO algorithm, as a state-of-the-art algorithm developed by OpenAI, is widely used in various fields, including the recent popular application of ChatGPT-4. To understand the PPO algorithm, it is essential to first grasp the concept of policy gradient. Policy gradient is a type of policy-based method that directly employs neural networks to approximate the policy function. The core update formula for the policy network parameters is

$$\nabla \bar{R}_\theta = \mathbb{E}_{\tau \sim p_\theta(\tau)} [R(\tau) \nabla \log p_\theta(\tau)] \quad (19)$$

In this formula, τ represents a sample trajectory of the agent in one episode, following the probability distribution $p_\theta(\tau)$ defined by the policy parameters θ ; $R(\tau)$ represents the cumulative discounted reward for that trajectory. The objective is to compute the gradient of the expected cumulative discounted reward with respect to the policy parameters. By performing gradient ascent on the policy parameters, we aim to improve the expected cumulative discounted reward.

In conventional policy gradient (PG) algorithms, a single trajectory sample corresponds to a singular update of the policy network parameters. In contrast, the proximal policy optimization (PPO) algorithm amalgamates the merits of both the advantage actor-critic (A2C) and trust region policy optimization (TRPO) algorithms, affording the capacity to employ trajectories sampled under the old policy for multiple iterations of policy network parameter updates. This capability arises from the introduction of importance sampling, ensuring unbiasedness, and the imposition of a clipping mechanism that imposes constraints upon the disparity between the two policy functions, specifically with respect to the ratio.

We introduce importance sampling, shown as

$$\nabla \bar{R}_\theta = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)} R(\tau) \nabla \log p_\theta(\tau) \right] \quad (20)$$

where θ' represents policy network parameters that are close to θ . We aim to perform sampling on the policy with parameters θ' and use the sampled data to update θ .

So far, we have been considering the entire trajectory. Now, we will decompose the trajectory into action–state sequence pairs (s, a) . Additionally, we aim to avoid blindly increasing the selection probability of corresponding actions during updates just because the returns are positive, which might lead to a decreased probability of selecting better actions due to low sampling frequency. Therefore, we introduce a baseline, which is the negation of the state value function $V_\pi(s)$, to adjust the selection probabilities of different actions during updates. This type of return function with a baseline is referred to as the advantage function, defined as shown in (21). Intuitively, this formula represents the difference between the estimate of the expected future return after taking the current action in a specific state and the estimate before taking the action. This difference measures the goodness of the action.

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s) \quad (21)$$

As a result, the original equation is transformed into the final expression.

$$\mathbb{E}_{(s_t, a_t) \sim \pi_{\theta'}} \left[\frac{p_{\theta}(s_t | a_t)}{p_{\theta'}(s_t | a_t)} A^{\theta'}(s_t, a_t) \nabla \log p_{\theta}(a_t^n | s_t^n) \right] \quad (22)$$

Actually, we are optimizing (23).

$$J^{\theta'}(\theta) = \mathbb{E}_{(s_t, a_t) \sim \pi_{\theta'}} \left[\frac{p_{\theta}(s_t | a_t)}{p_{\theta'}(s_t | a_t)} A^{\theta'}(s_t, a_t) \right] \quad (23)$$

Finally, we use a clip operation to constrain the policies corresponding to parameters θ and θ' from differing too much, shown as

$$J_{clip}^{\theta'}(\theta) \approx \sum_{(s_t, a_t)} \min \left(\frac{p_{\theta}(s_t | a_t)}{p_{\theta'}(s_t | a_t)} A^{\theta'}(s_t, a_t), \right. \\ \left. \text{clip} \left(\frac{p_{\theta}(s_t | a_t)}{p_{\theta'}(s_t | a_t)}, 1 - \varepsilon, 1 + \varepsilon \right) A^{\theta'}(s_t, a_t) \right) \quad (24)$$

where ε serves as a hyperparameter that restricts the difference between the target policy and the action policy.

3.3. High Performance Training Method for Tactics Control

Traditional control methods, such as rule-based logic or model predictive control, often rely on precise mathematical models of the environment and predefined decision-making logic. However, in complex and dynamic scenarios such as UAV tactical confrontation, these methods can struggle to adapt to unpredictable opponent behaviors, high-dimensional state spaces, and partial observability.

In contrast, PPO—a model-free deep reinforcement learning algorithm—offers several advantages for this task. For example, it offers adaptability, stability, robustness, exploration and generalization.

3.3.1. State Space

The construction of state features is crucial for training; good features can accelerate convergence and reduce the parameter space. We ultimately consider and design 13 state variables, covering the following: the difference between the desired altitude and the current altitude of the aircraft, ΔH ; the difference between the desired heading and the current heading of the aircraft, $\Delta \psi$; the difference between the desired velocity and the current velocity of the aircraft, ΔV ; the current altitude of the aircraft, H_{abs} ; the roll angle ϕ ; the pitch angle θ ; the aircraft's velocities in the x, y, and z axes, V_x , V_y , V_z ; the aircraft's airspeed V ; and the random observation weight ROW, shown as

$$S = [\Delta H, \Delta \psi, \Delta V, H_{abs}, \cos \phi, \sin \phi, \sin \theta, \cos \theta, V_x, V_y, V_z, V, ROW] \quad (25)$$

3.3.2. Action Space

The action space consists of four control parameters:

$$A = [C_a, C_e, C_r, C_t] \quad (26)$$

where C_a represents command given to the ailerons, which normalized between -1 and 1 . The ailerons are the movable control surfaces on the wings of an aircraft that control roll (or banking). A value of -1 typically corresponds to full left deflection, a value of 1 to full right deflection, and 0 to a neutral or centered position. The parameter C_e represents command given to the elevator, normalized between -1 and 1 . The elevator is a movable control

surface attached to the horizontal stabilizer of an aircraft, controlling pitch. A value of -1 typically corresponds to full downward deflection (nose down), a value of 1 to full upward deflection (nose up), and 0 to a neutral position. The parameter C_r represents command given to the rudder, normalized between -1 and 1 . The rudder is a movable control surface attached to the vertical stabilizer of an aircraft, controlling yaw (or direction). A value of -1 typically corresponds to full left deflection, a value of 1 to full right deflection, and 0 to a neutral position. The parameter C_t represents command given to the throttle, normalized between 0 and 1 . The throttle controls the power output of the engine(s). A value of 0 corresponds to no power (idle), and a value of 1 corresponds to full power.

3.3.3. Reward and Termination

We expect the UAV to fly according to the desired heading, altitude, and speed. Therefore, we have defined the following three reward functions:

$$R_\psi = e^{-\left(\frac{\Delta\psi}{\pi/36}\right)^2} \quad (27)$$

$$R_H = e^{-\left(\frac{\Delta H}{15.24}\right)^2} \quad (28)$$

$$R_V = e^{-\left(\frac{\Delta V}{20}\right)^2} \quad (29)$$

The above three functions are the heading, altitude, and velocity reward functions, which are Gaussian distributions, with the expected values as means and our desired error control as standard deviations. Specifically, we aim to control the heading error within $\frac{\pi}{36}$ rad, altitude error within 15.24 m, and speed error within 20 m/s.

In addition, the UAV's roll angle determines its turning ability. A greater roll angle increases the lift force component in the radial direction, allowing for faster turns and sharper turning radius, thereby enhancing maneuverability. Therefore, we have also designed a roll reward and introduced ROW to balance turning capability and flight stability, shown as (30).

$$R_\phi = ROW \cdot e^{-\left(\frac{\phi}{0.3}\right)^2} \quad (30)$$

When the UAV descends to a critical altitude of 100 km, we impose a penalty and terminate the episode.

$$R_{PH} = -1 \quad (31)$$

Additionally, we have defined other termination conditions:

1. Based on the PHP, if the UAV reaches the polling time but has not reached the desired heading;
2. Descending to a critical altitude: 1 km;
3. Exceeding the time limit: 1000 simulation steps;
4. Altitude exceeding 100 km;
5. Rotational angular velocities p, q, r exceeding 10^3 rad/s;
6. Speed exceeding 50 M
7. Acceleration exceeding 10 g.

3.3.4. Training Structure

We adopted an efficient training architecture and detailed design, central to which is a hybrid learning mechanism that combines orthogonal initialization with dynamic learning rate adjustments to achieve a more efficient training process and superior performance. In the initialization phase, the action network employs orthogonal initialization, setting a gain factor of 0.01 to ensure that all potential actions can be fairly explored from the

outset. Simultaneously, the value network's gain factor is set to 1, laying the groundwork for subsequent dynamic adjustments.

Furthermore, the architecture establishes a large-scale update cycle, with the number of updates determined by dividing the total time steps by the batch size. With each update, the algorithm dynamically adjusts the learning rate using a predetermined formula to accommodate the current training progress. Moreover, within the update cycle, the algorithm performs multiple steps to gather essential training data, including state observations, behavioral decisions, and reward logging. All data are properly stored during collection and utilized for the next phase of neural network training.

Following this, we employ generalized advantage estimation (GAE) to calculate advantage values, which, combined with state values, constitute an overall evaluation of actions, known as returns. Then, having acquired a series of observations, actions, advantage values, and returns, the algorithm enters a more refined update cycle. In this phase, experiential data are divided into several smaller batches, each used for independent updates to the neural network parameters.

The algorithm here adopts a hybrid strategy, comparing the performances of old and new policies and introducing entropy loss to encourage exploratory behavior. Ultimately, through a comprehensive consideration of policy loss, value loss, and entropy loss, the algorithm achieves a holistic optimization of the behavioral policy. This optimization process incorporates a gradient clipping mechanism. After each mini-batch learning session, gradients are checked for size, appropriately clipped, and then applied to update the neural network's parameters. The pseudocode for the tactical control training is shown as Algorithm 1; the training process is shown as Figure 6.

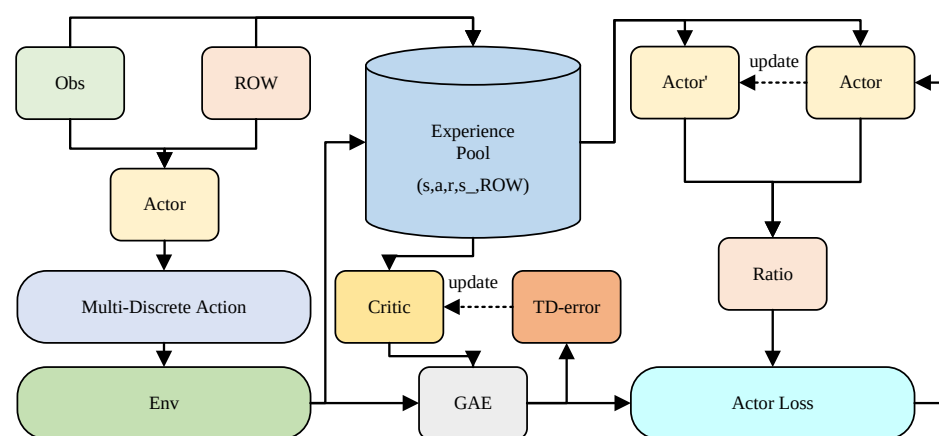


Figure 6. PHP-ROW PPO training process.

Table 2 shows the hyperparameter settings for training the tactical control training model.

Table 2. Hyperparameters of PHP-ROW PPO training model.

Hyperparameter	Value	Hyperparameter	Value
epoch	4	env-num	2
buffer-size	5000	hidden-size	128×128
mini-batch-size	2000	entropy-coef	0.001
clip-param(ϵ)	0.2	gae- λ	0.95
γ	0.99	learning-rate(α)	0.0003

Algorithm 1 PHP-ROW PPO tactical control training algorithm.

```

1: Initialize parameters.
2: Set  $num\_updates = \frac{total\_timesteps}{batch\_size}$ 
3: for  $update = 1$  to  $num\_updates$  do
4:   Set learning rate decay:  $lr_{now} = 1.0 - \frac{(update-1.0)}{num\_updates} \cdot lr$ 
5:   for  $step = 1$  to  $batch\_size$  do
6:     Increment global step count based on the number of parallel environments
7:     Store observations,  $O$ , and done flags,  $D$ 
8:     Input  $O$  into neural network (policy and value networks) to obtain actions,  $A$ , log
       probabilities,  $L$ , and estimated state values,  $V$ 
9:     Store  $V$ ,  $A$ , and  $L$ 
10:    m one step in the vector environment to obtain next observation,  $O'$ , reward,  $R$ ,
      and done flag,  $D'$ 
11:    Store  $R$ , update episode reward ( $ER$ ) and episode length ( $EL$ )
12:  end for
13:  Estimate the state value of  $O'$  using the value network, calculate advantages,  $Adv$ ,
    using GAE, and store the sum of  $Adv$  and  $V$  as returns,  $Ret$ 
14:  Obtain a full batch of  $O$ ,  $L$ ,  $A$ ,  $Adv$ ,  $Ret$ , and  $V$  for one rollout under  $batch\_size$ 
15:  for  $epoch = 1$  to  $epochs = 4$  do
16:    Shuffle the experience buffer
17:    for  $batch = 1$  to  $num\_batches$  with  $mini\_batch\_size = \frac{batch\_size}{num\_batches}$  do
18:      Input shuffled  $O$  and  $A$  into the neural network to compute new  $L$ , entropies,  $E$ ,
        and  $V'$ 
19:      Calculate the ratio,  $ratio = \frac{L'}{L}$  where  $L'$  are the new log probabilities
20:      Calculate policy loss,  $PL$ , based on  $Adv$ ,  $ratio$ , and clipping parameter,  $clip$ 
21:      Calculate value loss,  $VL$ , as the MSE between  $V'$  and  $Ret$ 
22:      Calculate the mean of  $E$  as entropy loss,  $EL$ 
23:      Calculate total loss:  $loss = PL - ent\_coef \cdot EL + VL$  where  $ent\_coef$  is the en-
        tropy coefficient
24:      Update parameters with gradient clipping
25:    end for
26:  end for
27: end for

```

4. Workflow of Intelligent Unmanned Aerial Vehicle Beyond-Visual-Range Adversarial Game

In this section, We propose an advanced framework for an intelligent beyond-visual-range six-degrees-of-freedom unmanned aerial vehicle adversarial game. This framework comprises four main components: tactical control, tactical decision-making, situational awareness, and proximal policy optimization training. The system structure is shown in Figure 7.

The situational awareness component is responsible for gathering state information from the environment. We construct a utility function to build situational awareness and employ fuzzy logic techniques to discretize continuous situational vectors.

The tactical decision-making component consists of a Bayesian network and a library of tactical maneuvers. We establish training and testing datasets to train and assess the Bayesian network's ability to make correct tactical maneuvers based on the given situation.

The PPO training component utilizes the PPO algorithm to train the UAV agent to fly according to desired headings, speeds, and altitudes. To enhance the UAV's ability to respond to unforeseen circumstances, we introduce a random observation weight control for roll reward functions and create a parallel training framework that combines high maneuverability and stability.

The tactical control component employs the actor network trained by PPO to map observations to UAV control inputs. The complete BVR game workflow entails pre-training the Bayesian network responsible for tactical decision-making and the PPO model responsible for tactical control. The situational awareness component senses and processes situational information. The Bayesian network makes decisions on the optimal tactical actions under the current situation, which are then further decomposed into decision criteria and desired headings, speeds, and altitudes. These constitute observation inputs to the PPO actor network, which ultimately outputs UAV control inputs to execute tactical control.

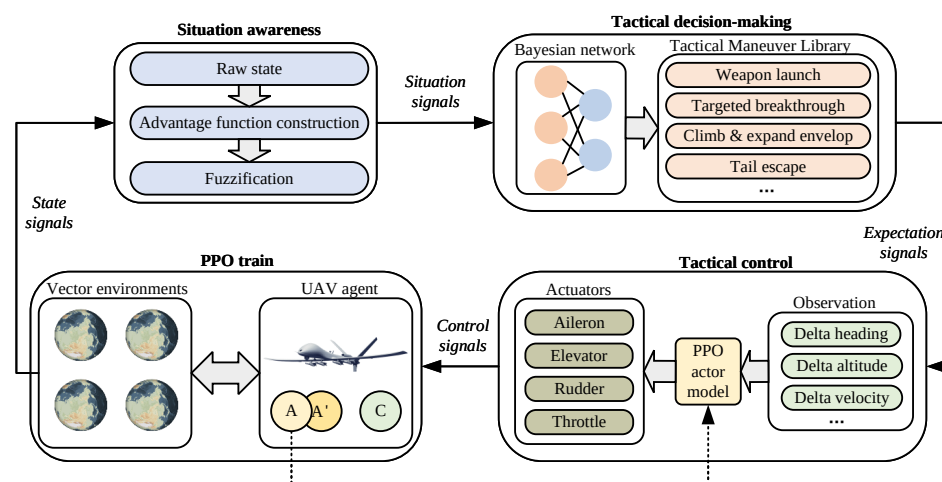


Figure 7. System structure.

5. Experiments and Results

In this section, the experimental platform setup is described, and two types of experiments are designed to prove the superiority of the proposed algorithm framework.

5.1. Experimental Platform

In our experiments, we employed the hardware and software platforms shown in Table 3.

Table 3. Hardware platform.

Hardware	Specification	Manufacturer	Location
CPU	Intel i7-10700KF	Intel	Santa Clara, CA, USA
GPU	Nvidia RTX 3080	NVIDIA	Santa Clara, CA, USA
Memory	16 GB	Kingston	Fountain Valley, CA, USA
GPU Memory	8 GB	-	-

Software Platform:

1. Algorithm Implementation: We used PyCharm (manufactured by JetBrains Corporation, Prague, Czech Republic) and Python (version:3.8) for algorithm development.
2. Environment Development: The environment was developed using Visual Studio 2019 (manufactured by Microsoft Corporation, Washington, DC, USA) and programmed in C++ (version:11).
3. Experiment Visualization: Tacview (version:1.9.0) was utilized for experiment visualization.

5.2. Experiment 1: Waypoint Navigation and Target Tracking

We designed four target waypoints located at distances of 2 km, 4 km, 6 km, and 8 km from the origin, all aligned along a straight line. The waypoint switching time was set to 400 simulation steps, with a total simulation duration of 3000 steps. The initial altitude was set to 20,000 feet, and the initial velocity was set to 300 m/s. This experiment primarily assessed the UAV's turning maneuverability. The experimental results are illustrated in Figures 8 and 9.

By comparing Figure 8a,b and Figure 8c,d, it is evident that the training curve with the PHP-ROW method converges much faster. The model reaches convergence around the 10 millionth step, whereas without the PHP-ROW method, the convergence is not achieved even by the end of the experiment, with significantly longer exploration times.

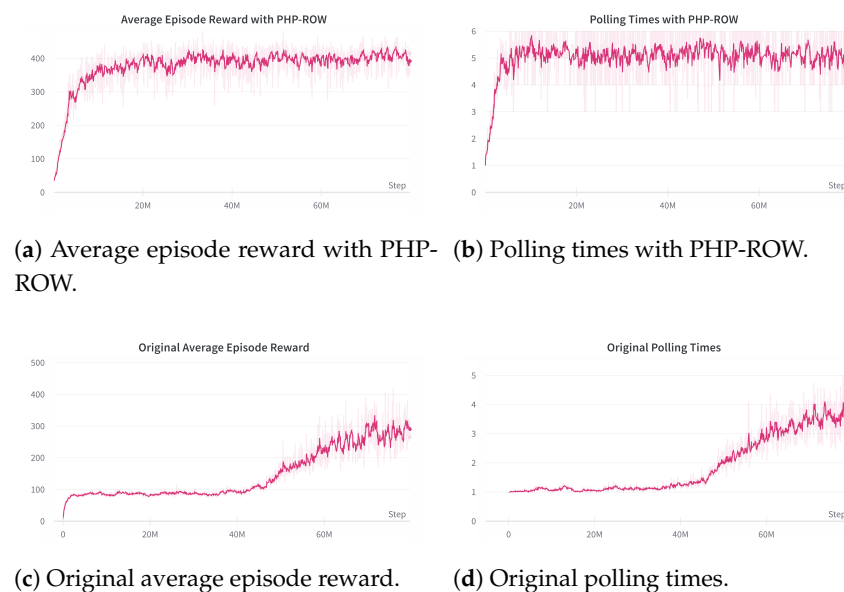


Figure 8. Training result curve comparison.

We compared the scenarios of full roll restriction and no roll restriction. Over a total of 200 experiments, we recorded the average number of rolls during level flight for the three models, as presented in Table 4. The model with full roll restriction cannot complete high-curvature turns but maintains smooth flight without frequent rolling maneuvers. Conversely, the model with no roll restriction exhibits rolling maneuvers even during level flight, resulting in lower stability and the potential for speed loss and altitude drop, as illustrated in Figure 9.

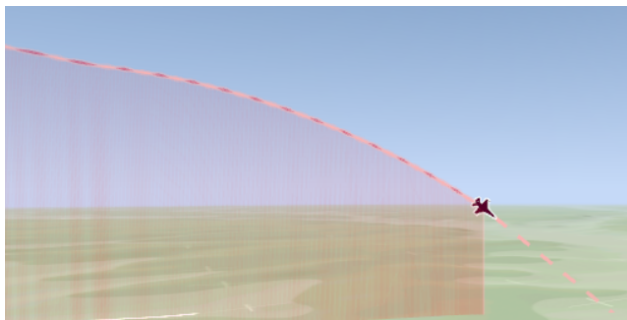
We also calculated the average roll angles during high-curvature turns for the three models, as shown in Table 4. The results indicate that our proposed algorithm allows for high-curvature turns while maintaining stable flight without frequent rolling maneuvers. This enriches the diversity of tactical control options.

Table 4. Comparison of three models.

Model Type	Average Number of Rolls During Level Flight (Count)	Average Roll Angle During High Curvature Turns (Rad)
PHP-ROW	0	1.521
Full roll restriction	0	1.813
No roll restriction	92	0.282

Finally, we conducted a PE problem, and the results revealed that the model employing the PHP-RGW algorithm (blue aircraft) is capable of tracking the target (red aircraft) inside the triangle with a larger turning radius, as illustrated in Figure 10.

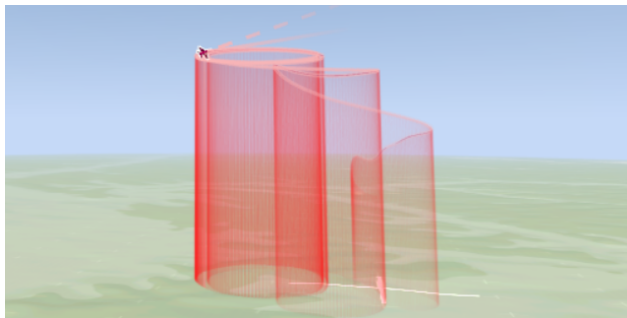
Additionally, to more intuitively demonstrate the significant enhancement in steering maneuverability afforded by the PHP-RGW method, we compared the disparity between the initial heading and the desired heading in the waypoint task between the PPO maneuver control models trained with this method and those without. We employed the ratio of the difference between the actual heading and the desired heading throughout the process to the difference between the initial heading and the desired heading as a normalized metric. As shown in Figure 11, it is evident that the model trained with the PHP-RGW method can stabilize well around the desired value in about 40 steps, whereas the original method requires nearly 200 steps.



(a) Without roll restriction (side view).



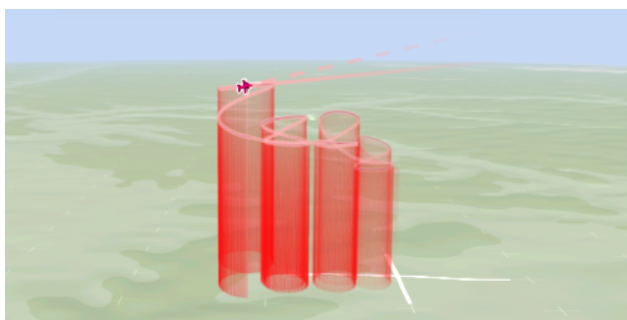
(b) Without roll restriction (top view).



(c) Navigation with restriction (side view).



(d) Navigation with restriction (top view).



(e) Navigation with PHP-RGW (side view).



(f) Navigation with PHP-RGW (top view).

Figure 9. Training results comparison.

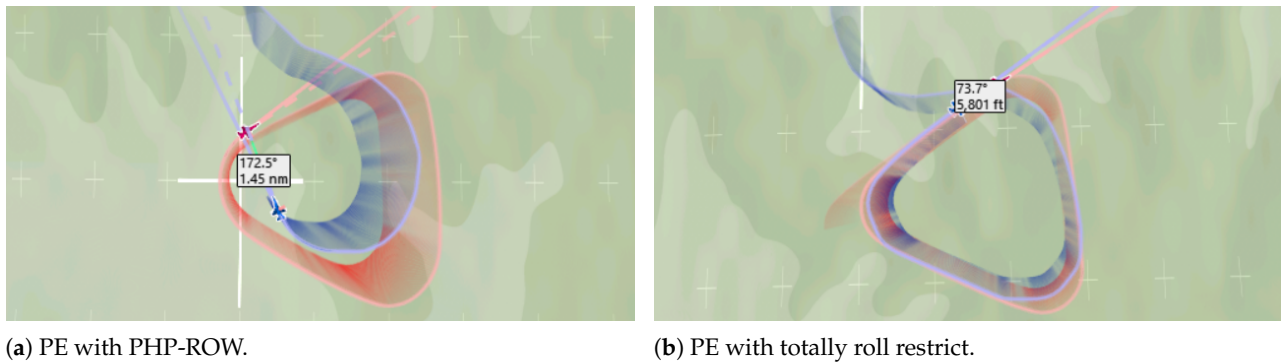


Figure 10. Comparison of PE results between two models.

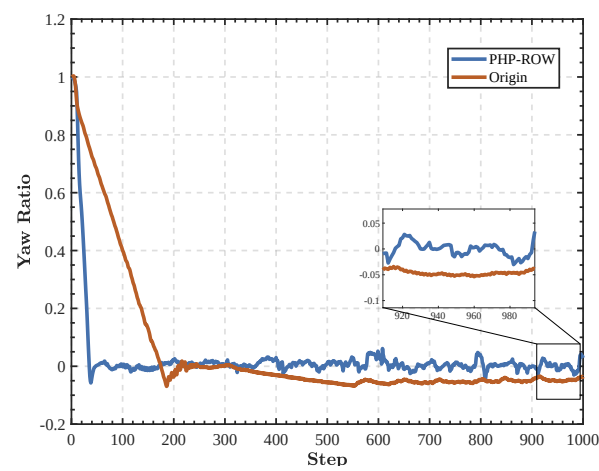


Figure 11. Tracking error in heading between PHP-ROW and original method.

5.3. Experiment 2: Engaging in a 1v1 Full-Cycle BVR Adversarial Game Against Experienced Human Players

AI followed the procedure depicted in Figure 7, with two human players who possessed 100 hours of simulated flight experience.

We conducted a beyond-visual-range game experiment across 20 rounds on a high-fidelity simulation platform that utilized real UAV parameters, shown as Figures 12 and 13. During the experiment, we recorded two key metrics: the average number of aircraft crashes caused by operator error on both sides and the average number of crashes resulting from engagements, shown as Table 5 and Figure 14. The results demonstrate that, in the course of the engagement, AI was capable of executing directional breakthrough maneuvers in the initial phase, transitioning to climb and expand maneuvers upon detecting the target, and ultimately launching missiles and performing biased guidance upon entering the attack zone. In contrast, human players exhibited unstable UAV control and lower efficiency in executing tactical maneuvers, making it challenging to strike a balance between weapon deployment and missile evasion.

We have documented the maneuver decisions made by the Bayesian network at different moments during one round of the AI–human adversarial game, as shown in Figure 15. It is observable that the initial stage tends to favor direct breakthrough maneuvers. After a period, it executes S maneuvers to search for the target. Upon target discovery, it carries out a climbing envelopment maneuver to complete attack positioning. When attack conditions are met, it conducts weapon firing and biased guidance. This process is repeated multiple times. During execution, when faced with threats from enemy missiles, it performs evasion maneuvers. After annihilating the target, it continues with direct breakthrough maneuvers until the round concludes.

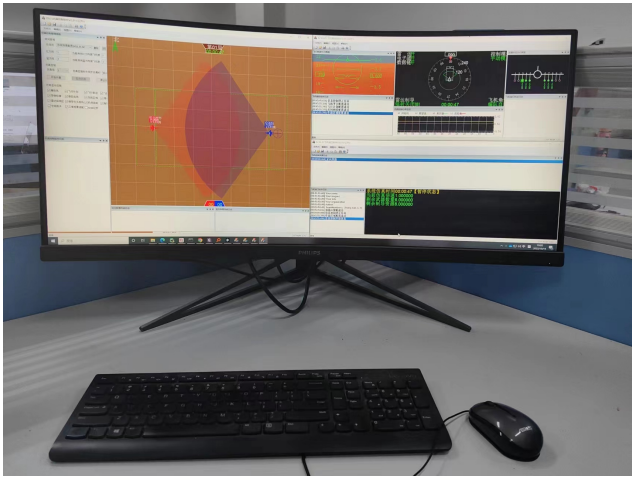


Figure 12. Manvs. AI battle platform.

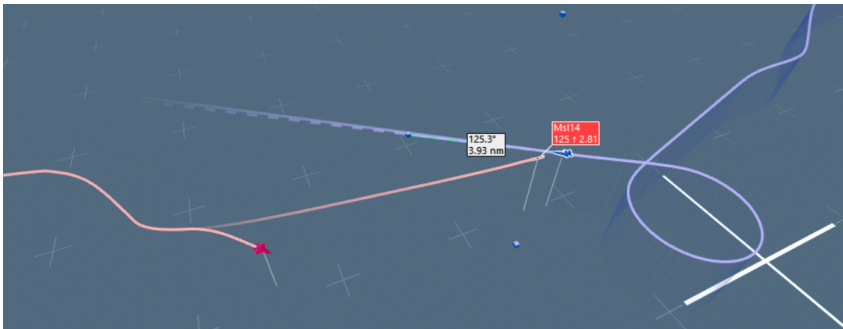


Figure 13. Man vs. AI battle process.

Table 5. Comparison of operational proficiency.

Operator	Average Operator Error Times (Count)	Average Engagement Times (Count)
AI	0	0.1
Human Player 1	0.5	0.15
Human Player 2	0.35	0.2

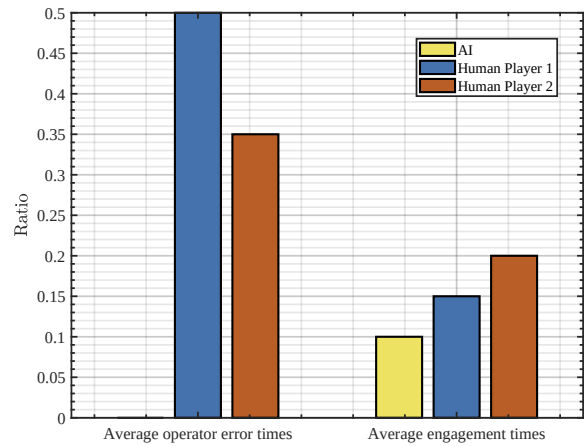


Figure 14. Comparison of operator error times and engagement times.

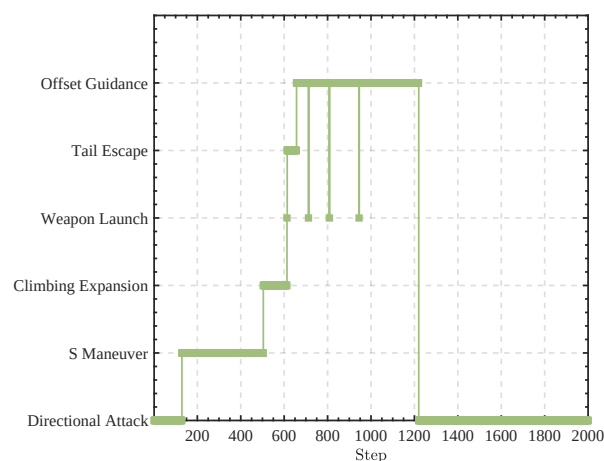


Figure 15. Bayesian network decision-making.

6. Conclusions

This paper delves into the realm of intelligent beyond-visual-range unmanned aerial vehicle (UAV) adversarial games, conducting an analysis of existing research accomplishments and identified shortcomings. Confronting issues such as inadequate simulation fidelity, restricted maneuverability, and a dearth of systematic research, we propose an intelligent unmanned beyond-visual-range adversarial game framework. Experimental results affirm that the PHP-ROW method introduced within this framework adeptly balances stability and heightened maneuverability in tactical control, concurrently bolstering exploratory training during the initial phases. The crafted situational information provides a comprehensive depiction of the battlefield landscape, exhibiting robust representativeness in tactical maneuvering and accommodating diverse maneuverability prerequisites. The Bayesian network architecture embedded within the framework adeptly deduces the optimal tactical maneuvers based on situational insights. The trained tactical control model precisely executes tactical actions, culminating in the successful fulfillment of unmanned beyond-visual-range adversarial game missions. Our forthcoming research endeavors will emphasize the expansion of situational insights and tactical maneuvering scales, alongside the exploration of intelligent beyond-visual-range adversarial game scenarios involving multiple UAV formations.

Author Contributions: Funding acquisition, J.Z.; Methodology, Y.C.; Methodology, G.S. and Q.Y.; Validation, D.W., Q.Y., Z.S., G.S. and D.W.; Writing—original draft, J.Z. and D.W.; Writing—review and editing, Q.Y. and D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Aeronautical Science Foundation of China (20220013053005).

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

DURC Statement: Current research is limited to the Unmanned Adversarial Game, which is beneficial in the aerospace field and does not pose a threat to public health or national security. Authors acknowledge the dual-use potential of the research involving the military and confirm that all necessary precautions have been taken to prevent potential misuse. As an ethical responsibility, authors strictly adhere to relevant national and international laws about DURC. Authors advocate for responsible deployment, ethical considerations, regulatory compliance, and transparent reporting to mitigate misuse risks and foster beneficial outcomes.

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

References

1. Mahadevan, P. The military utility of drones. In *CSS Analyses in Security Policy*; Center for Security Studies: Zurich, Switzerland, 2010; Volume 78.
2. Liu, Y.; Luo, Z.; Liu, Z.; Shi, J.; Cheng, G. Cooperative Routing Problem for Ground Vehicle and Unmanned Aerial Vehicle: The Application on Intelligence, Surveillance, and Reconnaissance Missions. *IEEE Access* **2019**, *7*, 63504–63518. [\[CrossRef\]](#)
3. Zhang, J.; Shi, Z.; Zhang, A.; Yang, Q.; Shi, G.; Wu, Y. UAV Trajectory Prediction Based on Flight State Recognition. *IEEE Trans. Aerosp. Electron. Syst.* **2023**, *60*, 2629–2641. [\[CrossRef\]](#)
4. Liu, Y.; Liu, Z.; Shi, J.; Wu, G.; Chen, C. Optimization of base location and patrol routes for unmanned aerial vehicles in border intelligence, surveillance, and reconnaissance. *J. Adv. Transp.* **2019**, *2019*, 9063232. [\[CrossRef\]](#)
5. Hsiao, F.B.; Lai, Y.C.; Tenn, H.K.; Hsieh, S.Y.; Chen, C.C.; Chan, W.L.; Hirst, R. The Development of an unmanned aerial vehicle system with surveillance, watch, autonomous flight and navigation capability. In Proceedings of the 21st Bristol UAV Systems Conference, Bristol, UK, 11–12 April 2006; pp. 16–19.
6. Dong, Y.; Ai, J.; Liu, J. Guidance and control for own aircraft in the autonomous air combat: A historical review and future prospects. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2019**, *233*, 5943–5991. [\[CrossRef\]](#)
7. Isaacs, R. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*; Courier Corporation: Chelmsford, MA, USA, 1999.
8. Kelley, H.J.; Lefton, L. Estimation of weapon-radius vs maneuverability tradeoff for air-to-air combat. *Aiaa J.* **1977**, *15*, 145–148. [\[CrossRef\]](#)
9. Davidovitz, A.; Shinar, J. Eccentric two-target model for qualitative air combat game analysis. *J. Guid. Control. Dyn.* **1985**, *8*, 325–331. [\[CrossRef\]](#)
10. Othling, W.L. Application of Differential Game Theory to Pursuit-Evasion Problems of Two Aircraft. Ph.D. Thesis, Air Force Institute of Technology, School of Engineering, Dayton, OH, USA, 1970.
11. Shinar, J. Solution techniques for realistic pursuit-evasion games. In *Advances in Control and Dynamic Systems*; Academic Press: Cambridge, MA, USA, 1981; Volume 17, pp. 63–124.
12. Ardema, M.; Heymann, M.; Rajan, N. Combat games. *J. Optim. Theory Appl.* **1985**, *46*, 391–398. [\[CrossRef\]](#)
13. Ma, L.; Zhang, H.; Meng, S.; Liu, J. Volcanic ash region path planning based on improved A-star algorithm. *J. Adv. Transp.* **2022**, *2022*, 1–20. [\[CrossRef\]](#)
14. Chen, T.; Zhang, G.; Hu, X.; Xiao, J. Unmanned aerial vehicle route planning method based on a star algorithm. In Proceedings of the 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, 31 May–2 June 2018; pp. 1510–1514.
15. Tseng, F.H.; Liang, T.T.; Lee, C.H.; Der Chou, L.; Chao, H.C. A star search algorithm for civil UAV path planning with 3G communication. In Proceedings of the 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kitakyushu, Japan, 27–29 August 2014; pp. 942–945.
16. Li, J.; Liao, C.; Zhang, W.; Fu, H.; Fu, S. UAV Path Planning Model Based on R5DOS Model Improved A-Star Algorithm. *Appl. Sci.* **2022**, *12*, 11338. [\[CrossRef\]](#)
17. Virtanen, K.; Raivio, T.; Hamalainen, R.P. Modeling pilot's sequential maneuvering decisions by a multistage influence diagram. *J. Guid. Control. Dyn.* **2004**, *27*, 665–677. [\[CrossRef\]](#)
18. Virtanen, K.; Karelaiti, J.; Raivio, T. Modeling air combat by a moving horizon influence diagram game. *J. Guid. Control. Dyn.* **2006**, *29*, 1080–1091. [\[CrossRef\]](#)
19. Ma, Y.; Ma, X.; Song, X. A case study on air combat decision using approximated dynamic programming. *Math. Probl. Eng.* **2014**, *2014*, pp.1–10. [\[CrossRef\]](#)
20. Teng, T.H.; Tan, A.H.; Tan, Y.S.; Yeo, A. Self-organizing neural networks for learning air combat maneuvers. In Proceedings of the The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–8.
21. M, P.S. Neural networks for pattern recognition. *Libr. Manag.* **2012**, *33*, 261–271.
22. 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. [\[CrossRef\]](#)
23. 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. [\[CrossRef\]](#)
24. Richard Sutton, A.G.B. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
25. Huang, C.; Dong, K.; Huang, H.; Tang, S.; Zhang, Z. Autonomous air combat maneuver decision using Bayesian inference and moving horizon optimization. *J. Syst. Eng. Electron.* **2018**, *29*, 86–97.
26. Zhou, D.; Sun, G.; Lei, W.; Wu, L. Space noncooperative object active tracking with deep reinforcement learning. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 4902–4916. [\[CrossRef\]](#)
27. Zhuang, X.; Li, D.; Li, H.; Wang, Y.; Zhu, J. A dynamic control decision approach for fixed-wing aircraft games via hybrid action reinforcement learning. *Sci. China Inf. Sci.* **2025**, *68*, 132201. [\[CrossRef\]](#)

28. Li, L.; Zhou, Z.; Chai, J.; Liu, Z.; Zhu, Y.; Yi, J. Learning continuous 3-DOF air-to-air close-in combat strategy using proximal policy optimization. In Proceedings of the 2022 IEEE Conference on Games (CoG), Beijing, China, 21–24 August 2022; pp. 616–619.
29. 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](#)]
30. Zhang, J.; Wang, D.; Yang, Q.; Shi, G.; Lu, Y.; Zhang, Y. Multi-Dimensional Decision-Making for UAV Air Combat Based on Hierarchical Reinforcement Learning. *Acta Armamentarii* **2023**, *44*, 1547.
31. Li, B.; Zhang, H.; He, P.; Wang, G.; Yue, K.; Neretin, E. Hierarchical Maneuver Decision Method Based on PG-Option for UAV Pursuit-Evasion Game. *Drones* **2023**, *7*, 449. [[CrossRef](#)]
32. Piao, H.; Sun, Z.; Meng, G.; Chen, H.; Qu, B.; Lang, K.; Sun, Y.; Yang, S.; Peng, X. Beyond-visual-range air combat tactics auto-generation by reinforcement learning. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 June 2020; pp. 1–8.
33. Kong, W.; Zhou, D.; Zhang, K.; Zhen, Y. Air combat autonomous maneuver decision for one-on-one within visual range engagement base on robust multi-agent reinforcement learning. In Proceedings of the 2020 IEEE 16th International Conference on Control & Automation (ICCA), Singapore, 9–11 October 2020; pp. 506–512.
34. Ma, X.; Xia, L.; Zhao, Q. Air-combat strategy using deep Q-learning. In Proceedings of the 2018 Chinese Automation Congress (CAC), Xi'an, China, 30 November–2 December 2018; pp. 3952–3957.
35. Hu, Z.; Gao, X.; Wan, K.; Li, J. Imaginary filtered hindsight experience replay for UAV tracking dynamic targets in large-scale unknown environments. *Chin. J. Aeronaut.* **2023**, *36*, 377–391.
36. Hu, D.; Yang, R.; Zuo, J.; Zhang, Z.; Wu, J.; Wang, Y. Application of deep reinforcement learning in maneuver planning of beyond-visual-range air combat. *IEEE Access* **2021**, *9*, 32282–32297. [[CrossRef](#)]
37. Lee, G.T.; Kim, C.O. Autonomous control of combat unmanned aerial vehicles to evade surface-to-air missiles using deep reinforcement learning. *IEEE Access* **2020**, *8*, 226724–226736. [[CrossRef](#)]
38. De Marco, A.; D'Onza, P.M.; Manfredi, S. A deep reinforcement learning control approach for high-performance aircraft. *Nonlinear Dyn.* **2023**, *111*, 17037–17077. [[CrossRef](#)]

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.