electronics

MDPI

Article

UAV Autonomous Aerial Combat Maneuver Strategy Generation with Observation Error Based on State-Adversarial Deep Deterministic Policy Gradient and Inverse Reinforcement Learning

Weiren Kong *, Deyun Zhou, Zhen Yang, Yiyang Zhao and Kai Zhang

School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China; dyzhou@nwpu.edu.cn (D.Z.); nwpuyz@mail.nwpu.edu.cn (Z.Y.); zhaoyiyang@mail.nwpu.edu.cn (Y.Z.); zhangkainwpu@mail.nwpu.edu.cn (K.Z.)

* Correspondence: k@mail.nwpu.edu.cn

Received: 17 June 2020; Accepted: 8 July 2020; Published: 10 July 2020

Abstract: With the development of unmanned aerial vehicle (UAV) and artificial intelligence (AI) technology, Intelligent UAV will be widely used in future autonomous aerial combat. Previous researches on autonomous aerial combat within visual range (WVR) have limitations due to simplifying assumptions, limited robustness, and ignoring sensor errors. In this paper, in order to consider the error of the aircraft sensors, we model the aerial combat WVR as a state-adversarial Markov decision process (SA-MDP), which introduce the small adversarial perturbations on state observations and these perturbations do not alter the environment directly, but can mislead the agent into making suboptimal decisions. Meanwhile, we propose a novel autonomous aerial combat maneuver strategy generation algorithm with high-performance and high-robustness based on state-adversarial deep deterministic policy gradient algorithm (SA-DDPG), which add a robustness regularizers related to an upper bound on performance loss at the actor-network. At the same time, a reward shaping method based on maximum entropy (MaxEnt) inverse reinforcement learning algorithm (IRL) is proposed to improve the aerial combat strategy generation algorithm's efficiency. Finally, the efficiency of the aerial combat strategy generation algorithm and the performance and robustness of the resulting aerial combat strategy is verified by simulation experiments. Our main contributions are three-fold. First, to introduce the observation errors of UAV, we are modeling air combat as SA-MDP. Second, to make the strategy network of air combat maneuver more robust in the presence of observation errors, we introduce regularizers into the policy gradient. Third, to solve the problem that air combat's reward function is too sparse, we use MaxEnt IRL to design a shaping reward to accelerate the convergence of SA-DDPG.

Keywords: aerial combat; reinforcement learning; robustness; sensor errors; network training; UAV

1. Introduction

Compared with human-crewed aircraft, military UAVs have attracted much attention for their low cost, long flight time, and fearless sacrifice [1]. With the development of sensor technology, computer technology, and artificial intelligence technology, the operational performance of military UAVs has been significantly improved, and the range of tasks that can be performed has been continuously expanded. Although military UAVs can perform reconnaissance and ground attack missions, most control decisions are made by ground station controllers. Because it is difficult to adapt to the fast and changeable air battle scene, the traditional ground station command operation is difficult to command the UAV for aerial combat [2]. Moreover, since the weapons and sensors

mounted by UAVs are less than the human-crewed aircraft, compared with the aerial combat of the human-crewed aircraft, UAVs should be fought in the close range. Therefore, the autonomous aerial combat of UAVs in close range is an important research topic.

This study focuses on gun-based aerial combat WVR, referred to as dogfighting. Although missiles have developed into major equipment for beyond-visual-range operations, particularly for second-generation fighters and later [3], their effectiveness and death toll have been lower than expected. Therefore, as the next-generation fighter, unmanned combat with guns is considered the key to WVR combat.

Since the early 1960s, a great deal of research has been done on autonomous aerial combat, and some remarkable research results have been published. The problem of aerial combat is modeled as pursuit-evasion games [4–6], and various theoretical and optimal control schemes provide solutions for autonomous aerial combat. Using differential game theory [7], the aerial combat model is modeled as a deterministic, complete information pursuer-evader game model. In Reference [8], approximate dynamic programming (ADP), a real-time autonomous one-to-one aerial combat method, was studied, and the results were tested in real-time indoor autonomous vehicle test environments (RAVEN). ADP differs from classical dynamic programming in that it constructs a continuous function to approximate future returns. ADP does not need to perform future reward calculations for each discrete state. Therefore, its real-time performance is reliable. A more complex two-on-one combat was treated as a differential game in three-dimensional space [9]. These theoretical approaches make possible the mathematical interpretation of the problem of aerial combat. However, they may include oversimplified assumptions, such as fixed roles or two-dimensional motion, to reduce complexity and computational time.

Some approaches are to model autonomous aerial combat as rule-based heuristic systems that imitate the behavior of human pilots. In Reference [10–12], the maneuvering library design, control application, and maneuvering identification based on the basic fighter maneuvers (BFM) expert system are presented. Based on a combination of the BFM library, target prediction, and impact point calculations, an autonomous aerial combat framework for two-on-two engagements was proposed in [9]. At the same time, the influence graph model is used to model the pilot's decision-making process choosing the right maneuver at every moment in the aerial combat [13,14]. The rule-based system produces real and reasonable simulation results. However, it is difficult to deal with all combat situations, modify previously designed rules, and add new rules.

Artificial neural network and reinforcement learning methods have recently exhibited an improved performance by generating effective new tactics for various simulation environments. Based on the artificial neural network, aerial combat maneuver decision-making is learned from a large number of aerial combat samples with strong robustness [15–18]. However, aerial combat samples need to include multiple groups of time series and aerial combat results. Due to the difficulty in obtaining samples for aerial combat, it is necessary to mark samples at each sampling time manually. Therefore, aerial combat samples' problem limits the application of the method of generating an aerial combat maneuver strategy based on a neural network. In Reference [19], an algorithm based on DDPG and a new training method was proposed to reduce training time and simultaneously obtain sub-optimal but effective training results. A discrete action space was used, and the aerial combat maneuvering decisions of an opponent aircraft were considered. However, the current reinforcement learning methods all consider that the acquired aerial combat status is accurate, which is inconsistent with the real aerial combat. In addition, reinforcement learning has also been applied in the fields of UAV flight control and multi UAVs cooperative flight control [20–22]. In Reference [20], deep Q-network, policy gradient and DDPG are used to design the 2-DOF flight attitude simulator control system and the feasibility of model-free reinforcement learning algorithm is proved through the experimental results. In Reference [21], the output reference model tracking control for a nonlinear real-world two-inputs–two-outputs aerodynamic system is solved by iterative model-free approximate value iteration (IMF-AVI), and theoretical analysis shows convergence of the IMF-AVI while accounting for

approximation errors and explains for the robust learning convergence of the NN-based IMF-AVI. In Reference [22], a UAV control policy based on DDPG to address the combination problem of 3-D mobility of multiple UAVs and energy replenishment scheduling, which ensures energy efficient and fair coverage of each user in a large region and maintains the persistent service.

Further practical improvements are required for the WVR autonomous aerial combat. We propose a novel the autonomous aerial combat maneuver strategy generation algorithm with high-performance and high-robustness based on the SA-DDPG algorithm. In order to consider the error of the aircraft sensors, we model the aerial combat WVR as a state-adversarial Markov decision process (SA-MDP), which introduce the small adversarial perturbations on state observations and these perturbations do not alter the environment directly, but can mislead the agent into making suboptimal decisions. SA-DDPG introduce a robustness regularizers related to an upper bound on performance loss at the actor-network to improve the robustness of the aerial combat strategy. At the same time, a reward shaping method based on MaxEnt IRL is proposed to improve the efficiency of the aerial combat strategy generation algorithm. Finally, the aerial combat strategy generation algorithm's efficiency and the performance and robustness of the resulting aerial combat strategy are verified by simulation experiments. Our contribution in this paper is a novel autonomous aerial combat maneuver strategy generation algorithm with high-performance and high-robustness based on SA-DDG. Unlike existing methods, the observation errors of UAV is introduced into the air combat model, and regularizer is introduced into the policy gradient to make the strategy network of air combat maneuver more robust. Finally, to solve the problem that air combat's reward function is too sparse, we use MaxEnt IRL to design a shaping reward to accelerate the convergence of SA-DDPG.

The remainder of this paper is organized as follows. Section 2 explains and defines the aerial combat model based on SA-MDP. Next, the specific theory and techniques for autonomous aerial combat maneuver strategy generation based on SA-DDPG are described in Section 3. A reward shaping method based on MaxEnt IRL is proposed in Section 4. Section 5 details the virtual combat environment and analyzes the performance of the proposed algorithm. This paper is concluded in Section 6. In this study, the UAV piloted by the proposed algorithm, and the target is referred to as the attacker and the target, respectively, for simplicity.

2. Aerial Combat Modeling

2.1. State-Adversarial Markov Decision Process (SA-MDP)

Markov decision process (MDP) is widely used in robotics, economics, manufacturing, and automatic control [23]. MDP is a mathematical framework for probabilistic modeling of the interactions between agents and the environment [24]. Agent is assumed to be learner or decision-maker, interacting with the environment. It receives a reward and a representation of the environment's state at each time step and imposes an action on the environment that may change its future state. This interaction between the agent and the environment is shown in Figure 1a.

A typical MDP is represented using a 6-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{D}, \mathcal{R})$, where $\mathcal{S} = \{s_0, s_1, ..., s_n\}$ is a set of possible states that represent a dynamic environment, $\mathcal{A} = \{a_0, a_1, ..., a_n\}$ is a set of available actions that the agent can select at a certain state, $\mathcal{T}$ is the state transition function which is determined by the current state and the next action of the agent. MDP assumes that performing an operation in a given state depends only on the current state-action pair, not on the previous states and actions, that is,

$$\mathbb{P}\left(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0\right) = \mathbb{P}\left(s_{t+1}|s_t, a_t\right) \tag{1}$$

Equation (1) is called Markov property [25]. So, its mapping relationship is shown in Equation (2).

$$\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \to [0, 1] \tag{2}$$

$\mathcal{T}$ is the state transition probability matrix that provides the probability of the system transition between every pair of the states, $\gamma \in [0, 1)$ is the discount rate that guarantees the convergence of total returns. $\mathcal{D}$ is the initial state distribution, and $\mathcal{R}$ is the reward function, which specifies the reward for taking action in a state whose mapping relationship is shown in Equation (3).

$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \tag{3}$$

The core objective of an MDP is to find a policy π for the agent, where the policy $\pi : \mathcal{S} \to \mathcal{A}$ specifies the action to take at the current state, where $\pi(s) = a$ denotes that action a is always executed in-state s following the policy π. Moreover, the goal is to find the optimal policy of π^* that maximizes the cumulative discounted reward over an infinite horizon:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}\left(s_t, \pi(s_t)\right)\right] \tag{4}$$

The value function of policy π at state s is the expected discounted return of starting in-state s and executing the policy. The value function can be computed as:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{t+k} \mid s_t = s\right] \tag{5}$$

The Q-value function of policy π at state s and using action a is the expected discounted return of starting in-state s and using action a and executing the policy. The Q-value function can be computed as:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{t+k} \mid s_t = s, a_t = a\right] \tag{6}$$

The Bellman equation [26] for $V^{\pi}(s)$ and $Q^{\pi}(s, a)$ can be computed as:

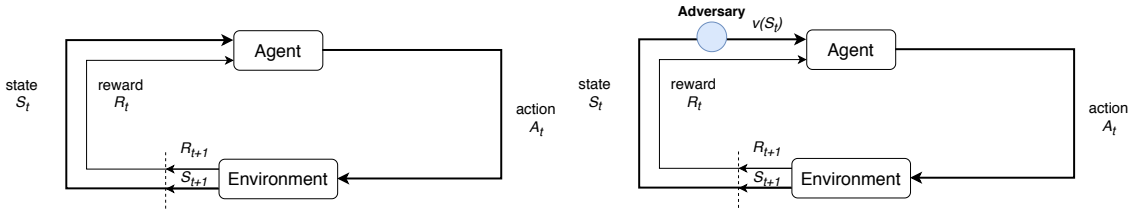
$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{t+k} \mid s_t = s\right] \\ &= \mathbb{E}_{\pi}\left[\mathcal{R}_t + \gamma \mathcal{R}_{t+1} + \gamma^2 \mathcal{R}_{t+2} \ldots \mid s_t = s\right] \\ &= \mathbb{E}_{\pi}\left[\mathcal{R}_t + \gamma(\mathcal{R}_{t+1} + \gamma \mathcal{R}_{t+2} \ldots) \mid s_t = s\right] \\ &= \mathbb{E}_{\pi}\left[\mathcal{R}_t + \gamma V^{\pi}(s_{t+1}) \mid s_t = s\right] \\ &= \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) \left[\mathcal{R}(s, \pi(s), s') + \gamma V^{\pi}(s')\right] \end{aligned} \tag{7}$$

The calculation process of the Bellman equation for $Q^{\pi}(s, a)$ is similar to $V^{\pi}(s)$:

$$Q^{\pi}(s, a) = \sum_{s' \in \mathcal{S}} P(s'|s, a) \left[\mathcal{R}(s, a, s') + \gamma Q^{\pi}(s', \pi(s'))\right] \tag{8}$$

In SA-MDP [27], an adversary $v(s) : \mathcal{S} \to \mathcal{S}$ is introduced shown in Figure 1b. The role of the adversary is to perturb the observation of the agent, such that the action is taken as $\pi(v(s))$, however, the environment still transits from state s rather than $v(s)$ to the next state. Since $v(s)$ may be different from s, the agent's action from $\pi(v(s))$ may be sub-optimal, and thus the adversary can reduce the reward earned by the agent. In practical control problems, the uncertainty of measurement or state estimation is often reflected as the noise in the worst case. In order to model the uncertainty of measurement of the UAV, we assume that the error between the measured value $v(s)$ and the real value s satisfies the small disturbance constraint, so we restricted $v(s)$ to choose the value inside the

$\mathcal{I}_\infty$ ball containing s. Formally, $v(s) \in B(s)$ is assumed, where $B(s)$ is a small set of states, $s \in B(s)$ and $B(s)$ is a set of task-specific "neighbouring" states of s.



(**a**) MDP (**b**) SA-MDP

Figure 1. The diagrammatic sketchs of MDP and SA-MDP.

The definitions of adversarial value and action-value functions under v is similar to the definition of regular MDP:

$$V_v^\pi(s) = \mathbb{E}_{\pi \circ v}\left[\sum_{k=0}^{\infty} \gamma^t \mathcal{R}_{t+k} \mid s_t = s\right] \tag{9}$$

$$Q_v^\pi(s,a) = \mathbb{E}_{\pi \circ v}\left[\sum_{k=0}^{\infty} \gamma^t \mathcal{R}_{t+k} \mid s_t = s, a_t = a\right] \tag{10}$$

where $\pi \circ v$ is the policy under observation perturbations: $\pi(v(s))$. The Bellman Equations of $V_v^\pi(s)$ and $Q_v^\pi(s,a)$ for fixed v can be computed as:

$$V_v^\pi(s) = \sum_{s' \in \mathcal{S}} P(s'|s, \pi(v(s)))\left[\mathcal{R}(s, \pi(v(s)), s') + \gamma V^\pi(s')\right] \tag{11}$$

$$Q_v^\pi(s,a) = \sum_{s' \in \mathcal{S}} P(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma Q^\pi(s', \pi(v(s')))\right] \tag{12}$$

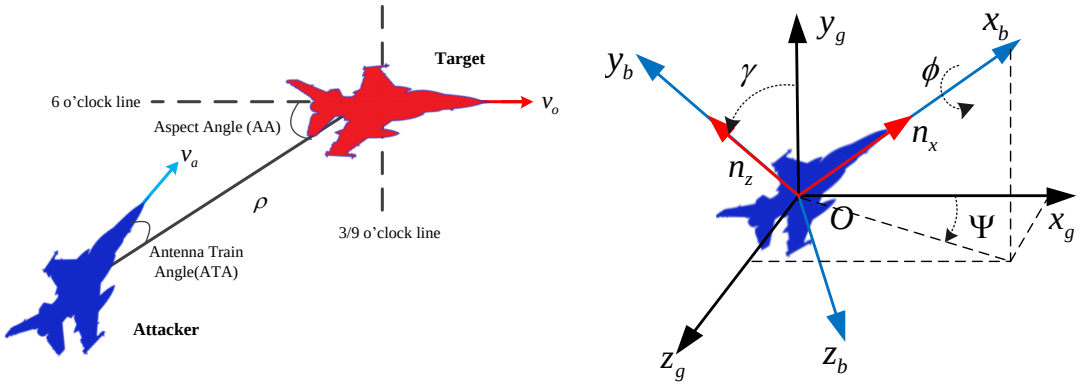
2.2. System Modeling

2.2.1. State Transitions

As for the state information s of the aircraft, this paper uses the ground coordinate system to describe the coordinates of the aircraft and represents the path of the aircraft in the process of aerial combat. Use the coordinate system in Figure 2b to describe the forces on the aircraft. The unbalance of torques in the model of state transitions is ignored, which plays a role in simplifying the model since the research focuses on the air maneuver decision algorithm. Therefore, the three-degree-of-freedom (3-DoF) aircraft particle motion model is selected to analyze its force characteristics. After coordinate transformation, a simplified aircraft dynamics equation can be obtained, as shown in Equation (13).

$$\begin{aligned} \frac{\mathrm{d}v}{\mathrm{d}t} &= g(n_x - \sin\gamma) \\ \frac{\mathrm{d}\gamma}{\mathrm{d}t} &= \frac{g}{v}(n_z \cos\phi - cos\gamma) \\ \frac{\mathrm{d}\Psi}{\mathrm{d}t} &= \frac{g n_z \sin\phi}{v\cos\gamma} \end{aligned} \tag{13}$$

where v is the speed of the aircraft, g represents the acceleration of gravity, n_x and n_z indicate the tangential overload and the normal overload, γ, Ψ and ϕ respectively indicate the aircraft's pitch angle, yaw angle, and roll angle. The following formula can be obtained by analyzing the force of the aircraft. n_x is tangential overload; n_z is normal overload. n_x is used to change the speed of the aircraft; n_z and ϕ determine the rate of change of pitch angle and yaw angle, which can change the flight direction and altitude.



(**a**) Combat geometry and parameters (**b**) Aircraft 3-DoF particle model

Figure 2. The geometric situation of the aerial combat.

According to the simplified dynamics equation of the aircraft, the change of UAV coordinates with time can be obtained, and the moving differential equations of the aircraft can be obtained, as shown in Equation (14).

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= v\cos\gamma\cos\Psi \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= v\cos\gamma\sin\Psi \\ \frac{\mathrm{d}z}{\mathrm{d}t} &= v\sin\gamma \end{aligned} \tag{14}$$

As can be seen from Equation (14), when the initial velocity v, the initial pitch angle γ, the initial yaw angle Ψ of the aircraft is given, the coordinates of the aircraft in the ground coordinate system can be obtained through integral calculation.

Single order inertial links are introduced into the three control commands to make the control process of the aircraft model more consistent with the real aircraft control process. The acceleration commands $(n_{xcmd}, n_{zcmd}, \phi_{cmd})$ due to actuator dynamics are given by

$$\begin{aligned} \frac{n_x}{n_{xcmd}} &= \frac{1}{1+\tau_x s} \\ \frac{n_z}{n_{zcmd}} &= \frac{1}{1+\tau_z s} \\ \frac{\phi}{\phi_{cmd}} &= \frac{1}{1+\tau_\phi s} \end{aligned} \tag{15}$$

where τ_x, τ_z and τ_ϕ are time constants of n_x, n_z and ϕ, respectively.

2.2.2. State Definition

The state of aerial combat can be fully described using the current motion parameters of the attacker and the target; that is, the inertial coordinates parameters, the velocity parameters and the attitude angle parameters of the two aircrafts. The position, velocity and attitude angle parameters of the agent and the opponent can be represented as follow:

$$\mathcal{S}_{motion} = [x_a, y_a, z_a, x_o, y_o, z_o, v_a, v_o, \gamma_a, \gamma_o, \phi_a, \phi_o, \Psi_a, \Psi_o] \tag{16}$$

where x_a, y_a, z_a are the inertial coordinate parameters of the attacker in the ground coordinate system; x_o, y_o, z_o are the parameters of the inertial coordinate of the opponent in the ground coordinate system; v_a is the velocity parameter of the attacker and v_o is the velocity parameter of the target; γ_a is the pitch

angle of the attacker and γ_o is the pitch angle of the target; ϕ_a the roll angle of the attacker and ϕ_o is the roll angle of the target; Ψ_a the yaw angle of the attacker and Ψ_o is the yaw angle of the target.

In this paper, the depth neural network (DNN) is used to approximate the critic network and actor-network. In order to ensure the good convergence performance of the deep neural network, the above parameters are normalized to the aerial combat state-space vector. State-space vectors are displayed in Table 1.

Table 1. The state space for the aerial combat model.

State	Definition	State	Definition
s_1	$\frac{x_a}{x_{max}-x_{min}}$	s_2	$\frac{x_o}{x_{max}-x_{min}}$
s_3	$\frac{y_a}{y_{max}-y_{min}}$	s_4	$\frac{y_o}{y_{max}-y_{min}}$
s_5	$\frac{z_a}{z_{max}-z_{min}}$	s_6	$\frac{z_o}{z_{max}-z_{min}}$
s_7	$\frac{v_a}{v_{max}-v_{min}}$	s_8	$\frac{v_o}{v_{max}-v_{min}}$
s_9	$\frac{\gamma_a}{2\pi}$	s_{10}	$\frac{\gamma_o}{2\pi}$
s_{11}	$\frac{\phi_a}{2\pi}$	s_{12}	$\frac{\phi_o}{2\pi}$
s_{13}	$\frac{\Psi_a}{2\pi}$	s_{14}	$\frac{\Psi_o}{2\pi}$

Then, the errors of the position, height, speed, and attitude sensors of the attacker and the errors of the position, height, speed and attitude sensors of the target are modeled as a disturbance of the information around the real state, which can be random or has antagonistic characteristics. In the paper, a random disturbance is used as an error in the state:

$$B_\epsilon(s) = \{\hat{s} : s - \epsilon \leq \hat{s} \leq s + \epsilon\} \tag{17}$$

where $B_\epsilon(s)$ is the bounded set of the state vector, it is an $\mathcal{I}_\infty$ ball around the state vector s. The perturbation range ϵ on each state feature is determined individually, depending on the standard deviation of that state feature.

2.2.3. Action Definition

In this paper, a continuous action space is used to control the aircraft. There are three continuous control commands referred to in Section 2.2.1 to control the aircraft. Therefore, the action space can be indicated by:

$$(n_{xcmd}, n_{zcmd}, \phi_{cmd}) \tag{18}$$

2.2.4. Reward Function and Terminal Condition

In real WVR aerial combat scenes, both aircraft maneuver at the same time; therefore, one aircraft will be located at the tail of the other aircraft. This ensures stable locking on the other party, and therefore, the opponent aircraft finds it challenging to get rid of the lock, and then attack the different plane. The combat geometry and parameters shown in Figure 2a are employed.

In Figure 2a, λ_a is the antenna train angle (ATA) of the attacker, which is the angle between the line-of-sight (LoS) vector and the attacker's velocity vector. ϵ_a is the aspect angle (AA) of the attacker, which is the angle between the LoS vector and the target's velocity vector. ϵ_a can be obtained in terms of the velocity of vector $\boldsymbol{v_a}$ and the LoS vector $\boldsymbol{\rho}$, where $\boldsymbol{\rho}$ denotes the LoS vector between the attacker and the target. The ATA and AA are allowed to take any value between $\pm 180^\circ$. ATA and AA can be obtained from Equations (19) and (20).

$$\lambda_a = \cos^{-1}\left[\frac{\boldsymbol{V_r} \times \boldsymbol{\rho}}{|\boldsymbol{V_r}||\boldsymbol{\rho}|}\right] \tag{19}$$

$$\epsilon_a = \cos^{-1}\left[\frac{\boldsymbol{V_b} \times \boldsymbol{\rho}}{|\boldsymbol{V_b}||\boldsymbol{\rho}|}\right] \tag{20}$$

As can be seen from Figure 2a, a smaller value of $|\lambda_a|$ means that the attacker is more accurate at aiming at the target, so $|\lambda_a|$ is inversely proportional to the shooting chance or offensive advantage. Similarly, a smaller $|\epsilon_a|$ indicates a lower probability of being hit by the target; therefore, $|\epsilon_a|$ is inversely proportional to the survival advantage. A termination condition of WVR aerial combat and rewards can be designed through the above-mentioned quantitative analysis of the aerial combat objectives and related parameters. The reward function can be designed using Equation (21) as follows:

$$\mathcal{R} = \begin{cases} 1.0, & |\lambda_a(s)| < 30^\circ \wedge |\epsilon_a(s)| < 60^\circ \\ -1.0, & |\lambda_o(s)| < 30^\circ \wedge |\epsilon_o(s)| < 60^\circ \\ 0, & otherwise \end{cases} \tag{21}$$

When a aircraft (assumed to be the attacker) satisfies the condition $|\lambda_a(s)| < 30^\circ \wedge |\epsilon_a(s)| < 60^\circ$ and satisfies this condition for a period of time, it is believed that the attacker can maintain the condition and complete the attack and destruction of the target. In this paper, since there is no killing simulation of UAVs, the problem is simplified as follows: when the attacker continuously tracks the target for five decision periods (e.g., the attacker is rewarded with one, five times over), it is considered that the attacker is stable tracking the target and the attacker wins. Similarly, when the target continuously tracks the attacker for five decision periods (e.g., the attacker is rewarded with -1, five times over), it is considered that the target is stable tracking the attacker and the attacker is defeated.

3. Autonomous Aerial Combat Maneuver Strategy Generation of UAV within Visual Range Based on SA-DDPG

3.1. Deep Deterministic Policy Gradient (DDPG)

DDPG is an algorithm which concurrently learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function and uses the Q-function to learn the policy. DDPG algorithm is successfully used in continuously high dimensional. DDPG is an actor-critic and model-free algorithm based on the deterministic policy gradient [28]. In other words, in DDPG, the actor directly maps states to actions instead of outputting the probability distribution across a discrete action space. In the DDPG algorithm, a policy parameter θ exists in the value function $Q(s, a)$, and this function must be derived from the policy. The gradient of the objective $J(\theta) = \mathbb{E}_{s\sim\boldsymbol{\mu}}[G]$ can be expressed as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^{\pi_\theta}}[\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s, a)|_{a=\pi_\theta(s)}] \tag{22}$$

In Equation (22) there are no expectations related to actions. Therefore, compared with a stochastic policy, policy strategies needless learning data and exhibit high efficiency, especially in the case of action spaces of large dimensions.

The policy and value functions of DDPG are approximated with deep neural networks. Like the DQN algorithm [29], experience replay [30] and dual network structures are introduced in the DDPG to make the training process more stable, and the algorithm converges more. DDPG is, therefore, an off-policy algorithm, and the sample trajectories from the experience replay buffers stored throughout the training.

3.2. State-Adversarial Deep Deterministic Policy Gradient (SA-DDPG)

In SA-MDP, the optimal adversary $v^*_\pi(s)$ that minimizes the total expected reward for a given π, and define the optimal adversarial value and action-value functions:

$$V^\pi_{v^*}(s) = \min_v V^\pi_v(s) \tag{23}$$

$$Q^\pi_{v^*}(s,a) = \min_v Q^\pi_v(s,a) \tag{24}$$

The Bellman equation for optimal adversary $v^*_\pi(s)$ can be written as:

$$V^\pi_{v^*}(s) = \min_{s_v \in B(s)} \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s_v)) \left[\mathcal{R}(s, \pi(s_v), s') + \gamma V^\pi(s')\right] \tag{25}$$

According to Theorem 5 in [27], given a policy π for a standard MDP, the inequality relation between the value function of standard MDP and SA-MDP can be obtained:

$$\max_{s \in \mathcal{S}} \{V^\pi(s) - V^\pi_{v^*}(s)\} \leq \alpha \max_{s \in \mathcal{S}} \max_{\hat{s} \in B(s)} D_{TV}(\pi(\cdot|s), \pi(\cdot|\hat{s})) \tag{26}$$

where $D_{TV}(\pi(\cdot|s), \pi(\cdot|\hat{s}))$ is the total variation distance

$$D_{TV}(\pi(\cdot|s), \pi(\cdot|s')) = \sqrt{\frac{2}{\pi}} \frac{\|\pi(\cdot|s) - \pi(\cdot|\hat{s})\|_2}{\sigma} + O(\|\pi(\cdot|s) - \pi(\cdot|\hat{s})\|_2^3) \tag{27}$$

between $\pi(\cdot|s)$ and $\pi(\cdot|\hat{s})$, and

$$\alpha := 2\left[1 + \frac{\gamma}{(1-\gamma)^2}\right] \max_{(s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}} \left|\mathcal{R}(s,a,s')\right| \tag{28}$$

is a constant that does not depend on π.

Equation (26) shows that as long as $D_{TV}(\pi(\cdot|s), \pi(\cdot|\hat{s}))$ is not too large for any $\hat{s}$ close to s the performance gap between $V^\pi(s)$ and $V^\pi_{v^*}(s)$ can be bounded. Equation (27) shows that as long as we can penalize $\|\pi(\cdot|s) - \pi(\cdot|\hat{s})\|_2$ the total variation distance between the two smoothed distributions can be bounded. In DDPG, we parameterize the policy as a policy network π_θ and the critic as a critic network Q_θ. According to Equation (26), for each state we need to find $\max_{\hat{s} \in B(s)} D_{TV}(\pi(\cdot|s), \pi(\cdot|\hat{s}))$, and we use $\max_{\hat{s} \in B(s)} \|\pi(\cdot|s) - \pi(\cdot|\hat{s})\|_2$ as a surrogate. Note that the smoothing procedure can be done completely in test time, and during training time our goal is to keep $\max_{\hat{s} \in B(s)} \|\pi(\cdot|s) - \pi(\cdot|\hat{s})\|_2$ small.

For the set $B(s) = \{\hat{s} : s|s_l \leq \hat{s} \leq s_u\}$, we can use the perturbation analysis tools Interval Bound Propagation (IBP) mentioned in Section 3.3 to give the upper and lower bounds of $\pi_\theta(\hat{s})$:

$$l_\pi(s;\theta) \leq \pi_\theta(\hat{s}) \leq u_\pi(s;\theta), \quad \forall \hat{s} \in B(s) \tag{29}$$

where the bounds $l_\pi(s;\theta)$ and $u_\pi(s;\theta)$ are differentiable functions of θ. So we can obtain the upper bound of the norm of the difference as follows:

$$\max_{\hat{s} \in B(s)} \|\pi_\theta(s) - \pi_\theta(\hat{s})\|_2 \leq \|u_\pi(s;\theta) - l_\pi(s;\theta)\|_2 \tag{30}$$

The SA-DDPG training algorithm is shown in Algorithm 1. The main difference comparing to regular DDPG is the additional loss term $L_{SA}(\theta)$, which provides an upper bound on $\max_{\hat{s} \in B(s)} \|\pi_\theta(s) - \pi_\theta(\hat{s})\|_2$. If this term is small, we can bound the performance loss under adversary.

Algorithm 1 State-Adversarial Deep Deterministic Policy Gradient (SA-DDPG).

Randomly initialize critic network $Q(s, a; \theta^Q)$ and actor $\pi(s; \theta^\pi)$ with weights θ^Q and θ^π
Initialize target network Q' and π' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\pi'} \leftarrow \theta^\pi$
Initialize replay buffer R.
for episode = 1 to M **do**
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state s_1
    for t = 1 to T **do**
        Select action $a_t = \pi(s; \theta^\pi) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute action at and observe reward r_t and observe new state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in R
        Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \pi(s_{i+1}; \theta^{\pi'}); \theta^{Q'})$
        Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i | \theta^Q))^2$
        Obtain upper and lower bounds on $\pi(s_i; \theta^\pi)$ using IBP:

$$l_\pi(s; \theta^\pi) \leq \pi(s; \theta^\pi) \leq u_\pi(s; \theta^\pi), \quad \forall s \in B(s_i) \tag{31}$$

        Upper bound on $\mathcal{I}_2$ distance:

$$L_{SA}(\theta^\pi) = \frac{1}{N} \sum_i \| u_\pi(s_i; \theta^\pi) - l_\pi(s_i; \theta^\pi) \|_2 \tag{32}$$

        Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\pi} J(\theta^\pi) = \frac{1}{N} \sum_i \left[\nabla_a Q(s, a; \theta^Q) \mid_{s=s_i, a=\pi(s_i; \theta^\pi)} \nabla_{\theta^\pi} \pi(s; \theta^\pi) \mid_{s_i} - \kappa \nabla_{\theta^\pi} L_{SA}(\theta^\pi) \right] \tag{33}$$

        Update the target networks:

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'} \\ \theta^{\pi'} &\leftarrow \tau \theta^\pi + (1 - \tau) \theta^{\pi'} \end{aligned} \tag{34}$$

    end for
end for

3.3. Interval Bound Propagation of Neural Network

The goal of IBP [31] is to find the lower and upper bounds in Equation (30). For clarity of presentation, we assume that the neural network is defined by a sequence of transformations h_k for each of its K layers. That is, for an input z_0, we have

$$z_k = h_k(z_{k-1}) \quad k = 1, ..., K \tag{35}$$

The output $z_K \in R^N$ has N logits. The simplest approach is to bound the activation z_k of each layer by an axis-aligned bounding box (i.e., $\underline{z}_k(\epsilon) \leq z_k \leq \overline{z}_k(\epsilon)$) using interval arithmetic. For $\mathcal{I}_\infty$ adversarial perturbations of size ϵ we have for each coordinate $z_{k,i}$ of z_k:

$$\begin{aligned} \underline{z}_{k,i}(\epsilon) &= \min_{\underline{z}_{k-1}(\epsilon) \leq z_{k-1} \leq \overline{z}_{k-1}(\epsilon)} e_i^T h_k(z_{k-1}) \\ \overline{z}_{k,i}(\epsilon) &= \max_{\underline{z}_{k-1}(\epsilon) \leq z_{k-1} \leq \overline{z}_{k-1}(\epsilon)} e_i^T h_k(z_{k-1}) \end{aligned} \tag{36}$$

where $\underline{z}_0(\epsilon) = x_0 - \epsilon$ and $\overline{z}_0(\epsilon) = x_0 + \epsilon$. For the fully connected layers that can be represented in the form $h_k(z_{k-1}) = Wz_{k-1} + b$, solving the optimization problems Equation (36) can be done efficiently with two matrix multiplies:

$$\begin{aligned} \mu_{k-1} &= \frac{\overline{z}_{k-1} + \underline{z}_{k-1}}{2} \\ r_{k-1} &= \frac{\overline{z}_{k-1} - \underline{z}_{k-1}}{2} \\ \mu_k &= W\mu_{k-1} + b \\ r_k &= |W| r_{k-1} \\ \underline{z}_k &= \mu_k - r_k \\ \overline{z}_k &= \mu_k + r_k \end{aligned} \tag{37}$$

where $|\cdot|$ is the element-wise absolute value operator. Propagating bounds through any element-wise monotonic activation function (e.g., ReLU, tanh, sigmoid) is trivial. Concretely, if h_k is an element-wise increasing function, we have:

$$\begin{aligned} \underline{z}_k &= h_k(\underline{z}_{k-1}) \\ \overline{z}_k &= h_k(\overline{z}_{k-1}) \end{aligned} \tag{38}$$

3.4. Maueuvering Strategy Generation Algorithm OutLine

The algorithm framework is composed of the aerial combat environment model, SA-DDPG, policy generation of the target, and reward shaping modules. The overall framework of the maneuvering strategy generation algorithm is shown in Figure 3.

In Figure 3, the inputs of the module of the aerial combat environment model are the actions of both the UCAVs, and the output is the next state and the reward value. The module of SA-DDPG contains an actor and a critic.

The actor's input is the state vector described in Section 2.2.1, and the output is the action vector described in Section 2.2.3. In order to improve the stability of the learning process, two neural networks are used in the actor, one as the target network and the other as the evaluation network. The structures of the two networks are the same, but the update methods are different. The evaluation network uses the deterministic policy gradient for updating, and the target network copies the online network's parameters through soft updates.

The critic's input is the state vector and the action vector, and the output is the $Q(s, a)$ value. Like the actor, there are two neural networks in the critic—one for the target network and the other for the evaluation network. The update method is also similar to that used in the case of the actor.

The input of the reward-shaping module includes the state vector, the action vector, and the next state. The shaping reward is a designed reward that is received from the aerial combat environment. The specific method of designing the shaping reward using MaxEnt IRL is explained in Section 5.

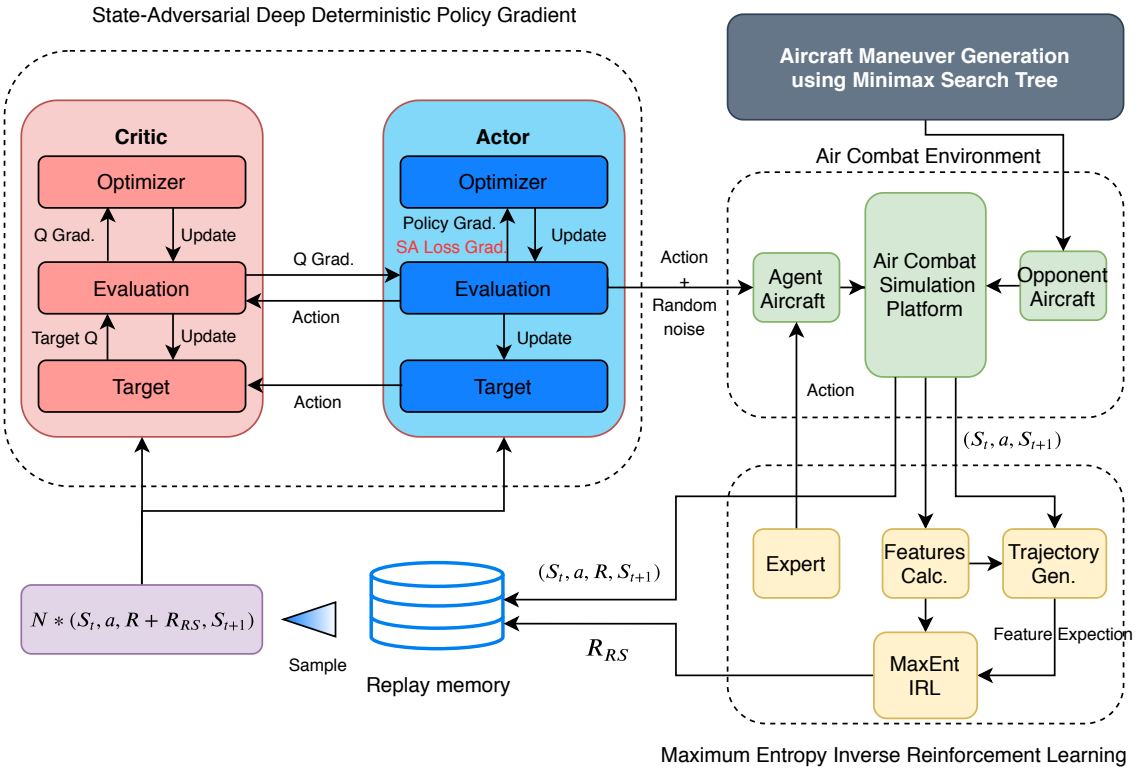


Figure 3. Aerial combat maueuvering strategy generation algorithm using SA-DDPG.

4. Reward Shaping Using Inverse Reinforcement Learning

4.1. Reward Shaping

Due to the learning efficiency of deep reinforcement learning, reinforcement learning has been restricting its practical application. Therefore, shortening the training time has become an important issue.

Reward shaping is a common technique to improve learning performance in reinforcement learning tasks [32]. The concept of reward shaping is to provide complementary incentives to agents to encourage them to move to a higher-paid status in the environment. These rewards and punishments, if applied arbitrarily, may divert agents from their intended objectives. In this case, agents converge to the optimal strategy in the case of reward formation, but they are suboptimal in the primary task.

Reward shaping accelerates the convergence rate of reinforcement learning by introducing additional rewards in an attempt to obtain more accurate rewards than those in the original environment. In general, the shaping reward function is expressed as follows:

$$\mathcal{R}_P(s) = \mathcal{R}_{RS}(s) + \mathcal{R}(s) \tag{39}$$

where $\mathcal{R}(s)$ denotes the binary reward function of the aerial combat environment in which the primary task owns. $\mathcal{R}_{RS}(s)$ denotes the shaping reward function.

4.2. Nonparameterized Features of Aerial Combat

The design of the shaping reward function is difficult because the behavior of pilots in aerial combat is difficult to describe. The shaping reward function also varies from pilot to pilot and may even change over time. A common way to design a shaping reward function is to represent it manually chosen non parameterized features. The nonparameterized features $\Phi(s)$ are functions of the states of aerial combat. We consider shaping reward functions as a linear combination of features in the following form:

$$\mathcal{R}_{RS}(s;w) = w^T \Phi(s) \tag{40}$$

where w is the weight vector, and $\Phi(s)$ is the feature vector with each component representing a nonparameterized feature of aerial combat. In order to improve the efficiency of the SA-DDPG algorithm, the nonparametric features of air combat selected in this paper are all continuous scalar functions sensitive to air combat state. In this work, we define the nonparameterized features in Table 2.

Table 2. The nonparameterized features of aerial combat.

Nonparameterized Feature	Description
$\|\rho\|$	The relative distance between blue and red aircrafts
$\|AA\|$	The absolute value of aspect angle
$\|ATA\|$	The absolute value of antenna train angle
$\|v_a - v_o\|$	The relative velocity between blue and red aircrafts

One can design the weight vector w to encourage or penalize certain features using the given reward function, and then use reinforcement learning to learn the corresponding optimal policy by maximizing the total reward.

4.3. Shaping Reward Modeling Using Maximum Entropy Inverse Reinforcement Learning

However, the weight vector of Equation (40) of the features set by hand is hard, and the weight vector of a bad setting may affect the optimality of the aerial combat maneuvering strategy. Therefore, this paper uses another method; it learned from the experience of aerial combat expert weight vector w based on inverse reinforcement learning (IRL) algorithm [33]. IRL can be defined as a reverse procedure of RL problems, which assumes that the demonstrations from experts are generated from the optimal policy. In the IRL problem, one is given a number of time histories of the agent's behaviors consisting of past states and actions. These past states and actions are often referred to as demonstrations. Ziebart [34] first applies the maximum entropy principle to solve the inverse reinforcement learning problem, for cases where the reward function depends only on the current state and represented via a linear combination of feature functions, namely in Equation (40). Formally, IRL considers the case in MDP, where the reward function is unknown. Correspondingly, there is a demonstration set $D = \{\xi_1, \xi_2, ..., \xi_N\}$, which is composed of expert demonstration trajectories. Each demonstration trajectory includes a set of state-action pairs, which are $\xi_i = \{s_0, a_0, s_1, a_1, ..., s_T, a_T\}$. Thus, we define an MDP/R process with no reward function, defined as tuple S, A, T, γ, D. Inverse reinforcement learning aims to learn the potential reward function $\mathcal{R}$.

Jaynes first proposed the principle of maximum entropy, and since then, it has been used in many areas of computer science and statistical learning [35]. In the basic maximum entropy formula, we give a set of samples of the target distribution and a set of constraints on the distribution, and then we estimate the distribution using the maximum entropy distribution that satisfies these constraints. The probability of a demonstration ξ over all paths of duration T is calculated:

$$\mathcal{P}(\xi|w) = \frac{1}{Z(w)} e^{\sum_{s \in \xi} w^T \Phi(s)} \tag{41}$$

where the partition function $Z(w)$ is a normalization constant, and it provides the solutions corresponding to a deterministic MDP, where the future state can be uniquely determined with the given action at the present state.

The goal of IRL is to find the optimal weight of w^*, such that the likelihood of the observed demonstrations is maximal under the distribution in Equation (41). The entropy of the path distribution under the constraint of maximizing the features of the observed data means the possibility of maximizing the observed data under the maximum entropy distribution we deduced above.

$$w^* = \arg\max L(w) = \arg\max \log \mathcal{P}(\xi|w) \tag{42}$$

This function is a convex function of deterministic MDPs, and the optimal solution can be obtained using the gradient optimization method. The gradient is the difference between expected empirical feature counts and the learner's expected feature counts, which can be expressed by the expected state visitation frequencies D_{s_i},

$$\nabla L(w) = \tilde{\Phi}_{\xi} - \sum_{s_i} D_{s_i} \Phi(s_i) \tag{43}$$

where $\tilde{\Phi}_{\xi} := \frac{1}{N} \sum_{\xi \in D} \Phi_{\xi}$ is the expected empirical feature count and $\Phi_{\xi} = \sum_{s \in \xi} \Phi(s)$. From Equation (43), when the expected state frequencies D_{s_i} is given, the gradient can easily be computed Equation (43) for optimization. The algorithm to approximate the state frequencies is proposed in [34]. The algorithm approximates the state frequencies for the infinite time horizon using a large fixed time horizon. It recursively "backs up" from each possible terminal state and computes the probability mass associated with each branch along the way by computing the partition function for Equation (41) at each action and state. These branching values yield local action probabilities, from which state frequencies in each timestep can be computed and summed for the total state frequency counts. The algorithm of approximating expected empirical feature count is shown in Algorithm 2.

Algorithm 2 Approximate expected empirical feature count.

$Z_{s_i,0} \rightarrow 1$
for j = 1 to N **do**
    $Z_{a_{i,j}} \rightarrow \sum_k P(s_k|s_i, a_{i,j}) e^{R_{RS}(s_i;w)} Z_{s_k}$
    $Z_{s_i} \rightarrow \sum_{a_{i,j}}$
end for
$P(a_{i,j}|s_i) \rightarrow \frac{Z_{a_{i,j}}}{Z_{s_i}}$
$D_{s_k,1} = P(S_i = S_{initial})$
for t = 1 to N **do**
    $D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$
end for
$D_{s_i} = \sum_t D_{s_i,t}$

Since the state space and action space in the aerial combat model are continuous, it is impossible to use Algorithm 2 to estimate the expected empirical feature count. Therefore, in this paper, the state space and action space in the aerial combat model are discretized, and each parameter is divided into multiple discrete values.

5. Simulation and Analysis

5.1. Platform Setting

5.1.1. Aerial Combat Simulation Platform Construction

The aerial combat simulation platform was built using the HARFANG 3D framework [36], which is a software framework used for the development of modern multimedia applications and for modeling the shapes of aircraft. The HARFANG framework can manage and display complex 3D scenes, play sounds and music, and virtual reality (VR) devices. It is a new multimedia application development framework, which is highly suitable for the development of games. Nowadays, ground controllers use VR equipment to control the UAVs; hence, we adopted the HARFANG 3D framework for our proposed aerial combat simulation platform.

The platform was able to simulate the aerial combat in certain airspaces, use the dynamic equation refer in Section 3.2 to simulate the flight characteristics of aircraft, and set the performance parameters. The platform was also able to reserve the aerial combat strategy interface of the two aircraft, thereby obtaining the aircraft control signals from the external environment. The platform could also support human control, where the interface received input from a keyboard or hands-on throttle and stick system (HOTS). The graph user interface of the aerial combat simulation platform is shown in Figure 4.

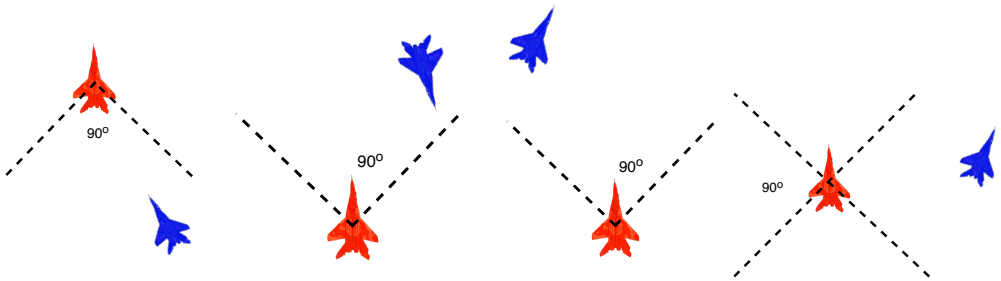
(**a**) Main view (**b**) Top view

Figure 4. The graph user interface of the aerial combat simulation platform.

The main interface of the aerial combat simulation platform is depicted in Figure 4. The interface presents a real-time combat situation of the aircraft, and the five positions on the screen display the health value, altitude and speed, radar image, attitude angle, and all the speed components.

5.1.2. Initial Setting for 1-vs-1 WVR Aerial Combat Engagement

As in Reference [37], Figure 5 shows the four different initial situation of the WVR aerial combat engagement based on the initial relative positions and orientations of the UCAVs. The initial position of the attacker concerning target in the horizontal direction is randomly chosen to be between 350 m and 1050 m in the offensive, defensive, and neutral cases. The two aircraft are at the same height initially.



(**a**) Offensive situation (**b**) Opposite situation (**c**) Defensive situation (**d**) Neutral situation

Figure 5. The four different initial situation of the WVR aerial combat engagement.

5.1.3. Aircraft Performance Parameters Setting

For experimental comparison, two different aircraft performance parameters were considered in this paper. Variation of the performance capabilities focuses on the following parameters: maximum thrust and maximum angular rates, as presented in Table 3. The dominant aircraft performance

parameters are called the "advantage" parameters, and the non-dominant aircraft performance parameters are called the "disadvantage" parameters. The masses of both the aircraft are the same and equal to 10,000 kg each and the speed range of both the aircraft is $[50, 250]$ meters per second.

Table 3. The aircraft performance parameters setting.

Parameter	Advantage Value	Disadvantage Value
Roll maximum rate (deg/s)	140	110
Maximum thrust (N)	100,000	80,000

5.1.4. Evaluation Metrics of Aerial Combat Strategy

The most direct method to evaluate an aerial combat strategy is to conduct an aerial combat confrontation with other aerial combat strategies and then determine the winner. We can also determine if the flight track generated by an aerial combat strategy is reasonable by observing the aircraft's flight tracks from both sides of the combat.

To quantitatively and accurately analyze the intelligence degree of an aerial combat strategy, this study proposes four metrics: intercept time T_I, defensive time T_D, offensive time T_O, and winning probability P_W. Intercept time is measured from the aerial combat simulation's beginning until an aircraft has a position of advantage. The position of advantage is the termination condition of the aerial combat, expressed in Equation (21). Defensive time is when a UCAV is at $|AA| > 90^\circ$ during aerial combat. Offensive time is when a UCAV is at $|ATA| < 90^\circ$ during aerial combat. The winning probability is the ratio of aerial combat simulations and the total number of aerial combat simulations.

5.1.5. Opponent Maueuvering Strategy

In this paper, the maneuver strategy of the target is designed using the Minimax Search Tree method (MST). The MST method simplifies and models the combat engagement between two players as a chess-type tree, a model-based method. See the schematic plot in Figure 6. The MST idea models the aircraft to following certain predefined candidate maneuvers. The candidate maneuvers include seven actions: max load factor left turn, max load factor right turn, max long acceleration, steady flight, max long deceleration, max load factor pull up, and max load factor pull over.

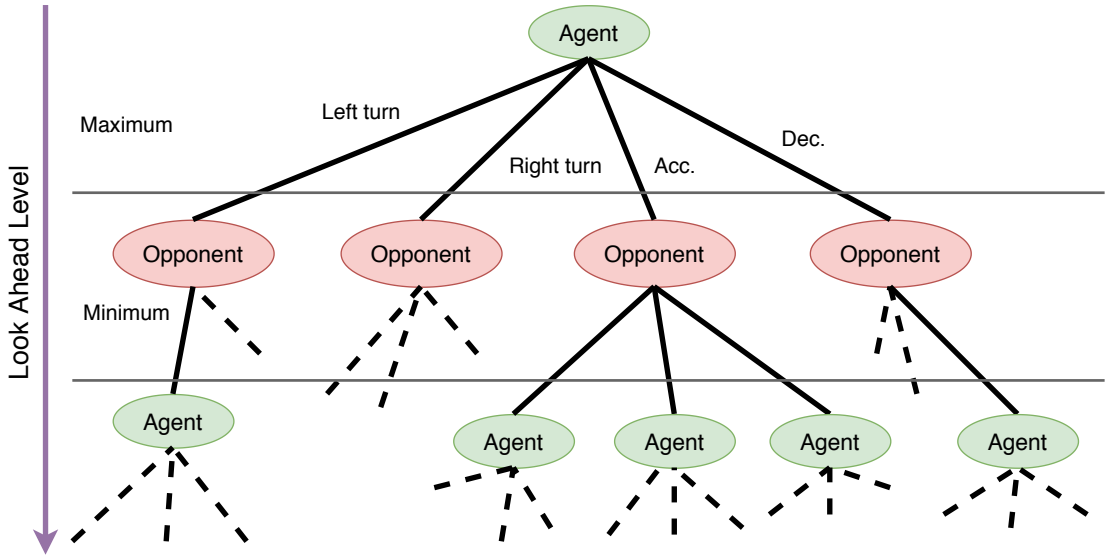


Figure 6. A schematic plot of the MST. Agent and Opponent indicate two aircraft.

At each decision point, both aircraft are assumed to take actions that are prespecified as the seven elemental maneuvers. The end states of both aircraft are obtained by numerically integrating the

motion equations. Given the seven potential options from each player, the terminating condition consists of 49 possibilities, represented by a seven-by-seven square matrix. Based on the terminate orientation, range, velocity, and terrain, a scoring function is defined for each index in the matrix. One player thus is to take actions that maximize the terminate score, while the other to minimize it.

The scoring function is composed of contributions consisting of an orientation score, a relative range score, and a velocity score. The contributions and the method of combining them were developed by incorporating various candidate functions into the program, performing numerous computer simulations, and studying the aircraft performance. The scoring function is defined in Equation (44).

$$Score(S) = \left[1 - \frac{|AA|/\pi + |ATA|/\pi}{2}\right] e^{-(|\rho| - R_D)/(k\pi)} \tag{44}$$

where R_D is the expected attacking range of weapons, and k is a coefficient adjusting the influence of $Score(S)$ in the total score.

5.2. Shaping Reward Experiment

5.2.1. Experiment Settings

Before using the SA-DDPG algorithm to obtain aerial combat maneuver strategy, the MaxEnt IRL algorithm is required to obtain shaping reward in order to provide the SA-DDPG algorithm. To complete the experiment, we used the aerial combat simulation platform to collect 500 different aerial combat counter trajectories manually. The initial states of these aerial combat trajectories are generated randomly, and the target uses the MST method for confrontation, and its look ahead level is set to 4.

Each component of the aerial combat features defined in Table 2 is divided into 20 parts on average, and each component of the action space is divided into five parts on average. Gradient descent method is used to solve $w*$ in Equation (42), and the learning rate is 0.01. The initial weight is set to -0.25.

5.2.2. Experiment Result and Analysis

The curve of weight w of the linear combination of features is shown in Figure 7. As can be seen from the figure, the curve showed a trend of convergence, and the weights all converge to some negative value after 6000 steps of learning. The optimal weight $w*$ are shown in Table 4.

In Table 4, the weight components are all negative, which indicates that the principle of aerial combat strategy used by aerial combat experts is to reduce $|\rho|$, $|AA|$, $|ATA|$ and $|v_a - v_o|$ as much as possible to complete the aerial combat, which is intuitively reasonable.

Table 4. The optimal weight $w*$ of the linear combination of aerial combat features.

Feature	$\|\rho\|$	$\|AA\|$	$\|ATA\|$	$\|v_a - v_o\|$
Optimal Weight	-0.046	-0.416	-0.515	-0.023

In order to quantitatively determine the effects of the four features of the aerial combat on the total shaping reward, the color map of the shaping reward using $|AA|$ and $|ATA|$ features and the color map of the shaping reward using $|\rho|$ and $|v_a - v_o|$ features are respectively drawn in Figure 8a. In Figure 8b, the influence of $|AA|$ and $|ATA|$ on shaping reward is far greater than that of $|\rho|$ and $|v_a - v_o|$ on shaping reward. Therefore, it is more important to have the advantage of a relative angle than the advantage of relative distance and speed in aerial combat, especially in short-range aerial combat.

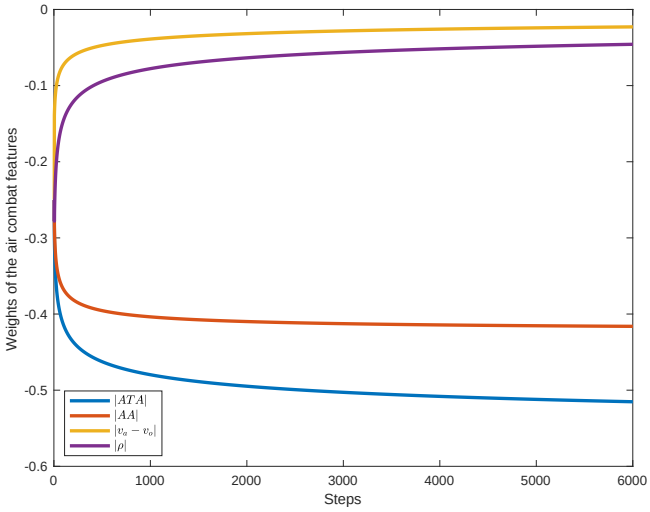
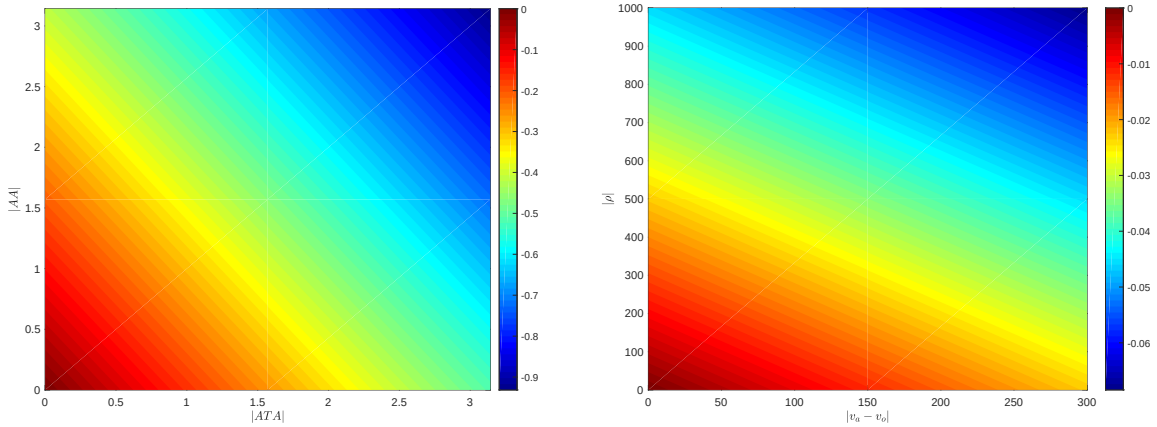


Figure 7. The convergence curve of weight w of the linear combination of aerial combat features.



(**a**) The color map of the shaping reward using $|AA|$ and $|ATA|$ features

(**b**) The color map of the shaping reward using $|\rho|$ and $|v_a - v_o|$ features

Figure 8. The effects of the 4 features of the aerial combat on the shaping reward.

5.3. Performance of Training Process

5.3.1. Experiment Settings

In this section, a comparative analysis of the training methods is performed. The five training algorithms are the DDPG algorithm without reward shaping (DDPG), the DDPG algorithm with reward shaping (RS-DDPG), the SA-DDPG algorithm with reward shaping, and the learning rate κ in Equation (33) is set to 0.1 (RS-SA-DDPG-0.1), the SA-DDPG algorithm with reward shaping and the learning rate κ is set to 1 (RS-SA-DDPG-1), and the SA-DDPG algorithm with reward shaping and the learning rate κ is set to 10 (RS-SA-DDPG-10) to training the aerial combat maneuver strategy.

When training the aerial combat strategies, the initial states of the two aircraft are randomly generated according to the four situations described in Section 5.1.2. At the same time, to prevent the two aircraft from falling into the "boring" Nash equilibrium of "biting the tail" during training, the performance parameters of the two aircraft to be different from one another: the attacker uses the "advantage" performance parameters, and the target uses the "disadvantage" performance parameters. The target uses the MST method for confrontation, and its look ahead level is set to 4.

The length of the actor network's input vector is 14, and that of the output vector is 3. The length of the critic network's input vector is 17, and that of the output vector is 1. The online actor-network, target actor-network, online critic network, and target critic network are constructed using three layers

fully connected neural network, and the nodes number of the hidden layer is 64. The output layer of the actor-network and the critic network have no activation function; furthermore, the input and hidden layers are all ReLU layers. The learning rate of the network is 0.01, and the discount factor γ is 0.9. The soft update factor of the target network is 0.01.

Additionally, the weight of the initialized neural network can be adjusted to be using the Xavier initializer. The batch size of the updating network is 1024, and the size of the experience replay memory is set to 10^6. The perturbation range ϵ for all normalized state components is 0.1.

5.3.2. Experiment Result and Analysis

The curve of the average reward of the aircraft shown in Figure 9. This curve is the average reward of all steps in every 500 episodes. Besides DDPG, which has only the original reward signal, the rewards of the other four algorithms all contain the shaping reward signal.

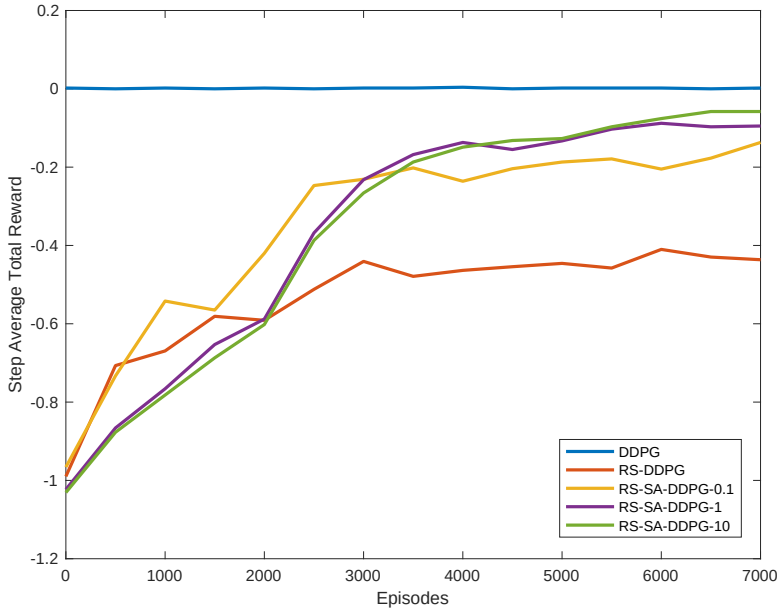


Figure 9. The convergence curve of weight w of the linear combination of aerial combat features.

As can be seen from Figure 9, Except DDPG, the other four algorithms tend to converge. This indicated that after the introduction of the shaping reward, the algorithm's efficiency in learning aerial combat maneuver strategy was significantly improved. By comparing the RS-DDPG algorithm with the 4 RS-SA-DDPG algorithms, it can be determined that the training convergence speed of the RS-SA-DDPG algorithms is slightly better than that of the RS-DDPG algorithm, and the final convergence result of RS-SA-DDPG algorithms is better than the RS-DDPG algorithm. By comparing the convergence curves of three RS-SA-DDPG algorithms with different κ parameters, the convergence speed will be slightly slower with the increase of κ, and it can converge to a better result, to obtain a better aerial combat maneuver strategy.

5.4. Testing Aerial Combat Maneuver Strategy

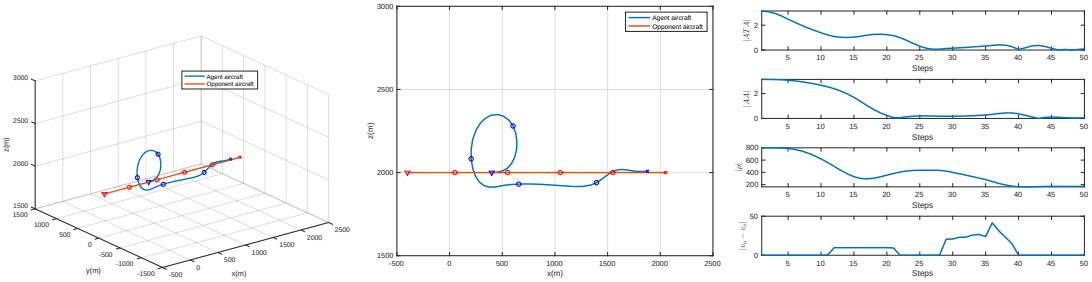
5.4.1. Experiment Settings

In this section, the intelligence degree of aerial combat maneuver strategy obtained using the RS-SA-DDPG-1 algorithm is evaluated against the target using the uniform linear motion maneuver strategy and four look-ahead MST maneuver strategy. The performance parameters of aircraft on both sides are kept at "Advantage" to ensure that the performance of the two aircraft is the same. The perturbation range ϵ is 0.1. Meanwhile, the initial situation of the attacker is set to "Defensive" when the attacker conducts aerial combat with the target with the uniform linear motion maneuver

strategy; the initial situation of the attacker is set to "Opposite" when the attacker conducts aerial combat with the target with the MST maneuver strategy.

5.4.2. Experiment Result and Analysis

The results of the simulation scenario against the uniform linear motion maneuver strategy are presented in Figure 10. The trajectories are depicted at intervals of 1 second. The downward-pointing triangle points at the trajectory represent the initial position. The cross points at the trajectory represent the end position of the aerial combat and, the circular points are marked every 10 s. To gain the situational advantage in aerial combat, the attacker performs a somersault maneuver to allow the target to overshoot it for about 30 s quickly. Then, the attacker speeds up to catch up with the opponent and, finally, slows down to prevent overshoot and steadily tracks the target. As a result, the target was successfully destroyed in 50 s by gunshot. It can also be seen from Figure 10 that the overall features of the attacker relative to the target increases from the initial positive value to zero, and there are some oscillations in the middle, but in the end, it can be stably maintained at zero, i.e., in an advantageous situation.



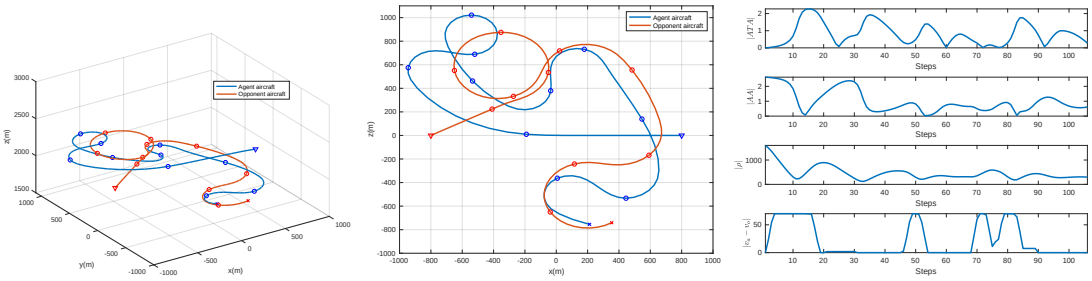
(**a**) The trajectory of aerial combat

(**b**) The projection of the trajectory on X-Z plane

(**c**) The result of the features in aerial combat

Figure 10. The simulation results of the aerial combat maneuver strategy using the RS-SA-DDPG-1algorithm against the uniform linear motion maneuver strategy.

The results of the simulation scenario against the four look-ahead MST maneuver strategy are presented in Figure 11. The attacker first accelerates to approach the target and overshoot quickly for about 20 s, and the target takes an advantageous situation. Then at 30 s, the attacker makes the left turn to bring the aircraft into the opposite situation. After 50 s, the attacker takes an advantageous position. Although the target adopted a series of maneuvers of acceleration or turning to try to get rid of it, the agent aircraft is still in an advantageous situation and gradually stabilized tracking.



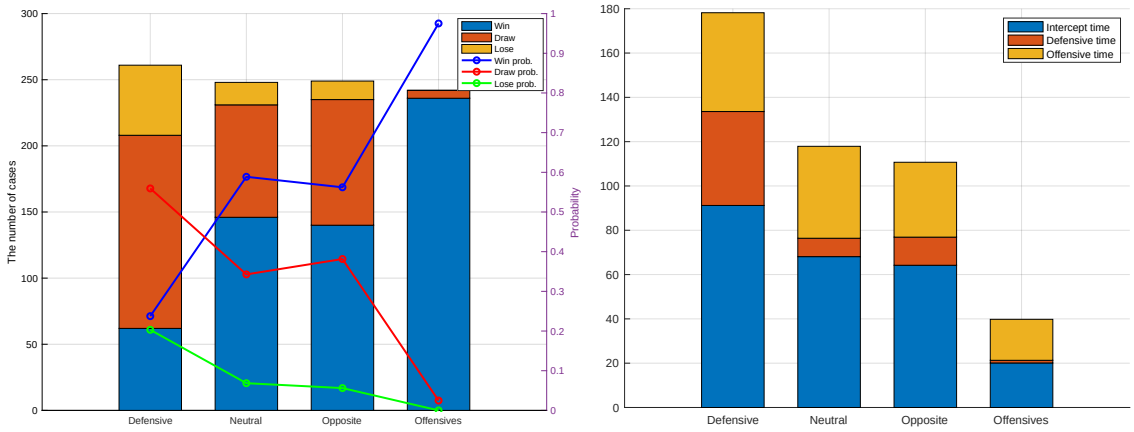
(**a**) The trajectory of aerial combat

(**b**) The projection of the trajectory on X-Z plane

(**c**) The result of the features in aerial combat

Figure 11. The simulation results of the aerial combat maneuver strategy using the RS-SA-DDPG-1 algorithm against the uniform linear motion maneuver strategy.

From the above two simulation results, it can be concluded that the aerial combat maneuver strategy using the RS-SA-DDPG-1 algorithm is effective. Besides, 1000 times Monte-Carlo simulations with the randomly initial situations were performed to ensure the universal validity of the performance in air combats between the strategies of the RS-SA-DDPG-1 algorithm and the four look-ahead MST method. The results are presented in Figure 12. As can be seen from Figure 12, regardless of the initial aerial combat situations, RS-SA-DDPG-1 strategy wins the air combats with a high probability, and the total probability of winning reaches 63.01%, and the average intercept time is 60.09 s.



(**a**) Winning probability of the aerial combat simulations

(**b**) Intercept time, defensive time, and offensive time of the aerial combat simulations

Figure 12. The results of the 1000 times aerial combat simulations.

5.5. Robustness Evaluation of the Aerial Combat Maneuver Strategy

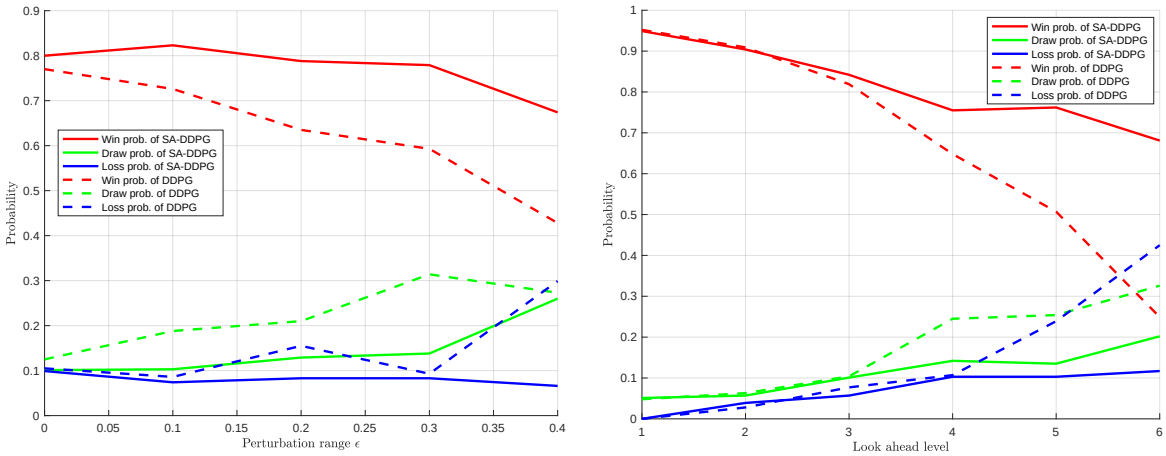
5.5.1. Experiment Settings

In this section, in order to test the robustness of the aerial combat strategy, two experiments are designed. First, we modified the perturbation range ϵ to evaluate the impact on the effectiveness of aerial combat strategies obtained by RS-SA-DDPG-1 and RS-DDPG algorithms. The perturbation range ϵ is set to $\{0, 0.1, 0.2, 0.3, 0.4\}$. For each perturbation range, we will conduct 1000 aerial combat simulations, and the initial aerial combat states are randomly generated to record the result of each aerial combat. Then, we modified the look ahead level of the aerial combat strategy of the target using the MST method to evaluate the impact on the effectiveness of aerial combat strategies obtained by RS-SA-DDPG-1 and RS-DDPG algorithms. The look-ahead level is set from 1 to 6. For each look ahead level, we will conduct 1000 aerial combat simulations, and the initial aerial combat states are randomly generated to record the result of each aerial combat.

5.5.2. Experiment Result and Analysis

The result of the experiment is shown in Figure 13. Firstly, the robustness of the airial combat strategy on the perturbation range of observation error is analyzed. In Figure 13a, when the perturbation range ϵ is 0, the winning rate of aerial combat strategies obtained by RS-SA-DDPG-1 and RS-DDPG algorithms are relatively high, about 0.8. This shows that the introduction of the regularization of robustness will not affect the performance of the airial combat strategy network. With the increase of the perturbation range ϵ, the winning rate of aerial combat strategies obtained by RS-SA-DDPG-1 algorithm remained relatively stable, and the winning rate of aerial combat strategies obtained by RS-DDPG algorithm gradually declined to about 40%. This shows that the robustness of the policy network with regularization is better than that without regularization. Then, the robustness of the airial combat strategy on the target intelligence level is analyzed. In Figure 13b, with the

increase of the look ahead level of target, the winning rate of aerial combat strategies obtained by RS-SA-DDPG-1 and RS-DDPG algorithm gradually decrease simultaneously, but the rate of decline of RS-SA-DDPG-1 is less than the rate of decline of RS-DDPG. Through these two experiments, it can be found that the robustness of the aerial combat strategies obtained by the RS-SA-DDPG-1 algorithm is better than that of the aerial combat strategies obtained by the RS-DDPG algorithm no matter changing the perturbation range ϵ or the intelligence degree of the target.



(**a**) Winning probability of the aerial combat simulations

(**b**) Intercept time, defensive time, and offensive time of the aerial combat simulations

Figure 13. The results for robustness evaluation of the aerial combat maneuver strategy.

6. Conclusions

In order to generate intelligent and robust aerial combat maneuver strategy, an algorithm for generating the aircraft's autonomous maneuver strategy based on the SA-DDPG algorithm and inverse reinforcement learning algorithm is proposed in this paper. In order to model the measurement errors of the sensors of aircraft in aerial combat, the process of aerial combat is modeled as SA-MDP. As can be seen from the experimental results above, the shaping reward obtained through the inverse reinforcement learning algorithm can effectively improve the speed of the aerial combat strategy learning. Meanwhile, through the comparison of aerial combat simulation results of the aerial combat strategies obtained by DDPG, SA-DDPG algorithm, it can be found that the aerial combat strategies obtained by the SA-DDPG algorithm have strong robustness. In this paper, only the decision of aircraft maneuver is studied, but the control strategy of weapon and sensor is not discussed, which needs study in the future.

Author Contributions: Conceptualization, W.K.; Methodology, W.K.; Data curation, Z.Y., and Y.Z.; Software, W.K.; Formal analysis, Z.Y. ; Project administration, D.Z.; Writing—original draft, W.K.; Writing—review and editing, Y.Z. and K.Z.; Funding acquisition, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 61603299 and Grant 61612385, and in part by the Fundamental Research Funds for the Central Universities under Grant 3102019ZX016.

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

References

1. Skjervold, E. Autonomous, Cooperative UAV Operations using COTS Consumer Drones and Custom Ground Control Station. In Proceedings of the MILCOM 2018—2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; pp. 1–6.
2. Gupta, S.G.; Ghonge, D.; Jawandhiya, P.M. Review of unmanned aircraft system (UAS). *Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET) Vol.* **2013**, *2*, 1646–1658.
3. Hebert, A.J. Fighter generations. *Air Force Mag.* **2008**, *91*, 1.
4. Ardema, M.; Rajan, N. An approach to three-dimensional aircraft pursuit–evasion. In *Pursuit-Evasion Differential Games*; Elsevier: Amsterdam, The Netherlands, 1987; pp. 97–110.
5. 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.
6. Jarmark, B.; Hillberg, C. Pursuit-evasion between two realistic aircraft. *J. Guid. Control Dyn.* **1984**, *7*, 690–694.
7. Greenwood, N. A differential game in three dimensions: The aerial dogfight scenario. *Dyn. Control* **1992**, *2*, 161–200.
8. 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.
9. Shin, H.; Lee, J.; Kim, H.; Shim, D.H. An autonomous aerial combat framework for two-on-two engagements based on basic fighter maneuvers. *Aerosp. Sci. Technol.* **2018**, *72*, 305–315.
10. Burgin, G.H.; Owens, A. *An Adaptive Maneuvering Logic Computer Program for the Simulation of One-to-One Air-to-Air Combat. Volume 2: Program Description*; Technical Report; NASA: Washington, DC, USA, 1975.
11. Chappell, A.R. Knowledge-based reasoning in the Paladin tactical decision generation system. In Proceedings of the IEEE/AIAA 11th Digital Avionics Systems Conference, Seattle, WA, USA, 5–8 October 1992; pp. 155–160.
12. Chappell, A.; Mcmanus, J.; Goodrich, K. Trial maneuver generation and selection in the PALADIN tactical decision generation system. In Proceedings of the Astrodynamics Conference, Hilton Head, SC, USA, 10–12 August 1992; p. 4541.
13. Virtanen, K.; Raivio, T.; Hämäläinen, R.P. An influence diagram approach to one-on-one air combat. In Proceedings of the 10th International Symposium on Differential Games and Applications, St. Petersburg, Russia, 8–11 July 2002; Volume 2, pp. 859–864.
14. Virtanen, K.; Karelahti, J.; Raivio, T. Modeling air combat by a moving horizon influence diagram game. *J. Guid. Control Dyn.* **2006**, *29*, 1080–1091.
15. McMahon, D.C. A neural network trained to select aircraft maneuvers during air combat: a comparison of network and rule based performance. In Proceedings of the 1990 IJCNN International Joint Conference on Neural Networks, San Diego, CA, USA, 17–21 June 1990; pp. 107–112.
16. 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.
17. Bo, L.; Zheng, Q.; Liping, S.; Youbing, G.; Rui, W. Air Combat Decision Making for Coordinated Multiple Target Attack Using Collective Intelligence. *Acta Aeronaut. Astronaut. Sin.* **2009**, *9*, E926.
18. Yang, Z.; Zhou, D.; Piao, H.; Zhang, K.; Kong, W.; Pan, Q. Evasive Maneuver Strategy for UCAV in Beyond-Visual-Range Air Combat Based on Hierarchical Multi-Objective Evolutionary Algorithm. *IEEE Access* **2020**, *8*, 46605–46623.
19. Yang, Q.; Zhu, Y.; Zhang, J.; Qiao, S.; Liu, J. UAV Air Combat Autonomous Maneuver Decision Based on DDPG Algorithm. In Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA), Edinburgh, UK, 16–19 July 2019; pp. 37–42.
20. Zuo, Y.; Deng, K.; Yang, Y.; Huang, T. Flight Attitude Simulator Control System Design based on Model-free Reinforcement Learning Method. In Proceedings of the 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 11–13 October 2019; pp. 355–361.
21. Radac, M.B.; Lala, T. Learning Output Reference Model Tracking for Higher-Order Nonlinear Systems with Unknown Dynamics. *Algorithms* **2019**, *12*, 121.

22. Qi, H.; Hu, Z.; Huang, H.; Wen, X.; Lu, Z. Energy Efficient 3-D UAV Control for Persistent Communication Service and Fairness: A Deep Reinforcement Learning Approach. *IEEE Access* **2020**, *36*, 53172–53184.
23. Song, H.; Liu, C.C.; Lawarrée, J.; Dahlgren, R.W. Optimal electricity supply bidding by Markov decision process. *IEEE Trans. Power Syst.* **2000**, *15*, 618–624.
24. Bernstein, D.S.; Givan, R.; Immerman, N.; Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* **2002**, *27*, 819–840.
25. Bellman, R. A Markovian decision process. *J. Math. Mech.* **1957**, *6*, 679–684.
26. Bellman, R. The Theory of Dynamic Programming. *Bull. Amer. Math. Soc.* **1954**, *60*, 503–515.
27. Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Boning, D.; Hsieh, C.J. Robust Deep Reinforcement Learning against Adversarial Perturbations on Observations. *arXiv* **2020**, arXiv:2003.08938.
28. 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.
29. 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.
30. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
31. Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.; Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv* **2018**, arXiv:1810.12715.
32. Laud, A.D. *Theory and Application of Reward Shaping in Reinforcement Learning*; Technical Report; 2004. Available online: https://www.ideals.illinois.edu/handle/2142/10797 (accessed on 8 July 2020).
33. Ng, A.Y.; Russell, S.J. Algorithms for inverse reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML), Stanford, CA, USA, 29 June–2 July 2000; Volume 1, p. 2.
34. Ziebart, B.D.; Maas, A.L.; Bagnell, J.A.; Dey, A.K. Maximum entropy inverse reinforcement learning. In Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), Chicago, IL, USA, 13–17 July 2008; Volume 8, pp. 1433–1438.
35. Jaynes, E.T. On the rationale of maximum-entropy methods. *Proc. IEEE* **1982**, *70*, 939–952.
36. Nichols, R.; Ryan, J.; Mumm, H.; Lonstein, W.; Carter, C.; Hood, J. Africa–World's First Busiest Drone Operational Proving Ground. In *Unmanned Aircraft Systems in the Cyber Domain*; New Prairie Press: Manhattan, KS, USA, 2019.
37. Ramírez López, N.; Żbikowski, R. Effectiveness of autonomous decision making for unmanned combat aerial vehicles in dogfight engagements. *J. Guid. Control Dyn.* **2018**, *41*, 1021–1024.