Zheng *et al*. *Intell Robot* 2023;3(1):76-94
DOI: 10.20517/ir.2023.04

Intelligence & Robotics

Research Article

Open Access

UAV maneuver decision-making via deep reinforcement learning for short-range air combat

Zhiqiang Zheng, Haibin Duan

School of Automation Science and Electrical Engineering, Beihang University, Beijing 100083, China.

Correspondence to: Prof. Haibin Duan, School of Automation Science and Electrical Engineering, Beihang University, No. 37, Xueyuan Road, Haidian District, Beijing 100083, China. E-mail: hbduan@buaa.edu.cn; ORCID: 0000-0002-4926-3202

How to cite this article: Zheng Z, Duan H. UAV maneuver decision-making via deep reinforcement learning for short-range air combat. *Intell Robot* 2023;3(1):76-94. http://dx.doi.org/10.20517/ir.2023.04

Received: 6 Jan 2023 **First Decision:** 16 Feb 2023 **Revised:** 26 Feb 2023 **Accepted:** 9 Mar 2023 **Published:** 18 Mar 2023

Academic Editor: Simon Yang **Copy Editor:** Yin Han **Production Editor:** Yin Han

Abstract

The unmanned aerial vehicle (UAV) has been applied in unmanned air combat because of its flexibility and practicality. The short-range air combat situation is rapidly changing, and the UAV has to make the autonomous maneuver decision as quickly as possible. In this paper, a type of short-range air combat maneuver decision method based on deep reinforcement learning is proposed. Firstly, the combat environment, including UAV motion model and the position and velocity relationships, is described. On this basic, the combat process is established. Secondly, some improved points based on proximal policy optimization (PPO) are proposed to enhance the maneuver decision-making ability. The gate recurrent unit (GRU) can help PPO make decisions with continuous timestep data. The actor network's input is the observation of UAV, however, the input of the critic network, named state, includes the blood values which cannot be observed directly. In addition, the action space with 15 basic actions and well-designed reward function are proposed to combine the air combat environment and PPO. In particular, the reward function is divided into dense reward, event reward and end-game reward to ensure the training feasibility. The training process is composed of three phases to shorten the training time. Finally, the designed maneuver decision method is verified through the ablation study and confrontment tests. The results show that the UAV with the proposed maneuver decision method can obtain an effective action policy to make a more flexible decision in air combat.

Keywords: Short-range air combat, unmanned aerial vehicle, deep reinforcement learning, maneuver decision, proximal policy optimization, flight simulation

1. INTRODUCTION

The unmanned aerial vehicle (UAV) has been applied in many fields for its low cost and high efficiency, including the military domain[1,2]. As sensors, artificial intelligence (AI) and other related technologies are developed and applied to UAV, the serviceable range of UAV in the military has been significantly expanded[3]. Air combat is one of the fields where the UAV is utilized.

The air combat is incredibly complex due to the difficulty of predicting the various scenarios that may arise unpredictably. During the combat, especially in short-range air combat, the UAV performs violent maneuvers which make the combat scenarios change instantly. There are roughly three categories of methods, optimization methods, game theory methods and AI methods, to solve the short-range air combat maneuver decision-making problem[3,4]. For the optimization methods, the maneuver decision problem is turned into an optimization problem, and solved by the optimization theories, like optimization algorithms[4,5]. However, the optimization problem for air combat is a high-dimensional and large-scale problem that is usually so difficult and complex that most optimization-based decision-making algorithms cannot be executed in real-time and adapt to practical constraints. Game theory methods, especially differential games[6,7], are another popular method to solve air combat maneuver decision problems. Whereas, the mathematical models of game theory methods are difficult to establish and their solutions are always hard to prove the adequacy and necessity[4]. For the complex air combat problem, AI methods catch the scholars for their flexibility and operability. The expert system method[8] is one of the AI methods, which tries to map the human knowledge and experience into the flight rule library to complete the maneuver decision. However, the mapping process is complex because human knowledge and experience are always hard to generalize into rules and describe mathematically. On the other hand, once the rule library has been built, the maneuver policy is fixed and inflexible[3]. The methods based on reinforcement learning (RL) are popular for air combat problems recently.

RL is a type of machine learning method that improves its action policy with respect to the reward obtained by repeated trial and error in an interactive environment[9]. In recent years, the neural network has been combined with RL, which is called deep reinforcement learning (DRL). Many types of DRL algorithms have been proposed, like deep Q network (DQN), deep deterministic policy gradient (DDPG), proximal policy optimization (PPO), etc. DRL has been applied in UAV path control[10], quadrupedal locomotion control[11], autonomous platoon control[12], etc. At the same time, DRL has been used to improve the operational efficiency of air combat[9,13,14]. In ref.[3], the DQN is used to solve one-to-one short-range air combat with an evaluation model and maneuver decision model, and basic-confrontation training is presented because of the huge computation load. The PPO is used to learn the continuous 3-DoF short-range air combat strategy in ref.[15], and it can adapt to the combat environment to beat the enemy who uses the minmax strategy. With the DRL method, the UAV can adapt to the changing combat situation and make a reasonable maneuver decision. However, the huge computation load and slow training speed are still the main issues that needs to be addressed when combining DRL with air combat problems.

In this paper, the problem of one-to-one UAV short-range air combat maneuver decision-making is studied. The main contributions are summarized as follows.

(1) The air combat environment with the attacking and advantage areas is designed to describe the relationship between the UAVs. And to increase the confrontment difficulty, the attacking conditions, blood values and the enemy action policy consisting of prediction and decision are introduced.

(2) The GRU layer is applied to design the neural networks which are used as the PPO's actor and critic networks. On the other hand, the observation of the UAV for actor network and combat state information for critic network are designed to separate their roles.

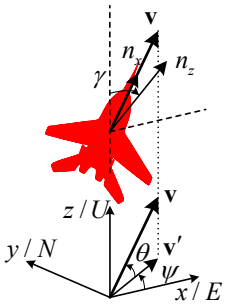

Figure 1. The UAV's motion model in the ground coordinate system. UAV: unmanned aerial vehicle

(3) To improve flexibility and intelligence during the confrontment, the reward function is divided into three parts: dense reward, event reward and end-game reward. Then, the phased training process is designed from easy to difficult to ensure the feasibility of training.

The remainder of this paper is organized as follows. In section II, the UAV motion model, air combat environment and its process are introduced. Then the method designed is explained in section III, which includes the PPO algorithm, some improved points and the design for state, action and reward function to combine the PPO algorithm with the air combat problem. The action policy for the enemy and training process is also introduced. Next, the training results and simulation analysis are presented in section IV. Finally, the conclusion is presented in section V.

2. PROBLEM FORMULATION

2.1. UAV motion model

The three-degree-of-freedom UAV model is considered because the main consideration in short-range air combat problem is the position and velocity relationship between the two sides[3]. The motion model is established in the ground coordinate system as East-North-Up (ENU) coordinate system, which is shown in Figure 1.

To simplify the problem, assume that the velocity direction is fixed with the -axis of the body coordinate system, and the UAV's motion model is shown as[14,16]

$$\begin{cases} \dot{x} = v\cos\theta\cos\psi \\ \dot{y} = v\cos\theta\sin\psi \\ \dot{z} = v\sin\theta \\ \dot{v} = g\left(n_x - \sin\theta\right) \\ \dot{\theta} = \dfrac{g}{v}\left(n_z\cos\gamma - \cos\theta\right) \\ \dot{\psi} = \dfrac{gn_z\sin\gamma}{v\cos\theta} \end{cases}, \tag{1}$$

where x, y and z are the UAV's position coordinate values and $\mathbf{p} = [x, y, z]^T$, $v = \|\mathbf{v}\|$ is the velocity, $\dot{x}$, $\dot{y}$ and $\dot{z}$ are the values of $\mathbf{v}$ on the ground coordinate axes, γ, θ and ψ represent the flight-path bank angle, flight-path angle and heading angle respectively, g is the acceleration of gravity, n_x is the overload in velocity direction, and n_z is the normal overload. Noting that the heading angle ψ is the angle between $\mathbf{v}$', the projection of $\mathbf{v}$ on the xoy plane, and ox axis. The basic control parameters n_x, n_z and γ in the motion model can be expressed as

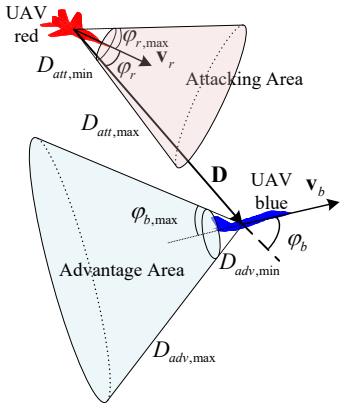

Figure 2. The relationship between the red and blue sides during the battle.

a control input vector $\mathbf{u} = [n_x, n_z, \gamma]^T \in \mathbb{R}^3$, which can be used to control the UAV's position and velocity in the air combat maneuver decision making[16]. At the same time, the UAV's motion must satisfy the constraints which are expressed as[3]

$$\begin{cases} v_{\min} \leqslant v \leqslant v_{\max} \\ \theta_{\min} \leqslant \theta \leqslant \theta_{\max} \\ -\pi < \gamma \leqslant \pi \\ 0 \leqslant \psi < 2\pi \\ n_{x\min} \leqslant n_x \leqslant n_{x\max} \\ n_{z\min} \leqslant n_z \leqslant n_{z\max} \end{cases}, \tag{2}$$

where subscript *min* and *max* mean the minimum and maximum values, and the parameters are set as $v_{\min} = 30m/s$, $v_{\max} = 150m/s$, $\theta_{\min} = -\pi/4$, $\theta_{\max} = \pi/4$, $n_{x\min} = -1$, $n_{x\max} = 2.5$, $n_{z\min} = -4$ and $n_{z\max} = 4$. These constraints are judged and processed when the control input vector $\mathbf{u}$ is input to the UAV motion model and the UAV state, including v, θ and ψ, is updated.

By giving the control parameters' values and UAV's state at the time step t, the UAV's state at the time step $t+1$ can be easily obtained by the Runge-Kutta method[14].

2.2. Air combat environment

In the one-to-one short-range air combat environment, there are two UAVs, divided into red and blue. The red aims to gain the advantage situation over the blue until the blue side is destroyed by its weapon, and the blue aims to do the opposite[4]. In this paper, the red UAV is controlled by the proposed decision method based on DRL algorithm. The relationship between the red and blue sides during the battle, which is shown in Figure 2, is mainly described by both sides' velocity vectors, the red's velocity $\mathbf{v}_r$ and the blue's velocity $\mathbf{v}_b$, and the relative position vector $\mathbf{D}$, which can be expressed as

$$\mathbf{D} = \mathbf{p}_b - \mathbf{p}_r, \tag{3}$$

where $\mathbf{p}_r$ and $\mathbf{p}_b$ are the red and blue's position vectors respectively.

The angle φ_r, named the attacking angle, is between $\mathbf{v}_r$ and $\mathbf{D}$, and the formula can be expressed as[16]

$$\varphi_r = \arccos \frac{\mathbf{v}_r \cdot \mathbf{D}}{\|\mathbf{v}_r\| \cdot \|\mathbf{D}\|}. \tag{4}$$

During the confrontation, the red has a chance to attack and deal damage to the blue only if the blue is in its attacking area. φ_r can be used to describe whether the blue in the red's attacking area and the conditions can be described as

$$\begin{cases} D_{att,\min} \leqslant \|\mathbf{D}\| \leqslant D_{att,\max} \\ \varphi_r \leqslant \varphi_{att,\max} \end{cases}, \tag{5}$$

where $D_{att,\min}$ and $D_{att,\max}$ are the minimum and maximum attacking distances, and $\varphi_{att,\max}$ is the maximum attacking angle that has a chance to deal damage.

The angle φ_b, named escaping angle, is between $\mathbf{v}_b$ and $\mathbf{D}$, and the formula is expressed as[16]

$$\varphi_b = \arccos \frac{\mathbf{v}_b \cdot \mathbf{D}}{\|\mathbf{v}_b\| \cdot \|\mathbf{D}\|}. \tag{6}$$

During the confrontation, the red not only needs to keep the blue in its attacking area but also tries to avoid being attacked by the blue. Thus, the advantage area for the red is defined behind the blue and can be described as

$$\begin{cases} D_{adv,\min} \leqslant \|\mathbf{D}\| \leqslant D_{adv,\max} \\ \varphi_b \leqslant \varphi_{esp,\max} \end{cases}, \tag{7}$$

where $D_{adv,\min}$ and $D_{adv,\max}$ are the minimum and maximum advantage distances, and $\varphi_{esp,\max}$ is the maximum escaping angle that the red is in its advantage area.

During air combat, the UAV has limited attacking resource and needs attack only under certain conditions. The enemy should be both within the red's attacking area and hard to escape. These conditions can be described as[15]

$$\begin{cases} D_{att,\min} \leqslant \|\mathbf{D}\| \leqslant D_{att,\max} \\ \varphi_r \leqslant \varphi_{att,\max} \\ \varphi_b \leqslant \varphi_{esp,att} \end{cases}, \tag{8}$$

where $\varphi_{esp,att}$ is the maximum escape angle for the enemy if the red wants to make a successful attack. On the other hand, not every attack could cause damage to the other side, and it will probably take more than an attack to destroy the other side. Thus, it is assumed that the UAVs have blood value B_0 when initialized, and every attack has the probability p_{att} to cause some damage ΔB, which is shown as

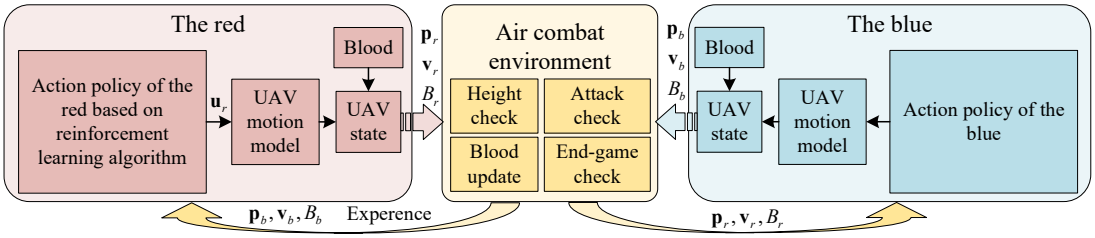

Figure 3. Short-range air combat framework.

$$\Delta B = \begin{cases} -B_1 & 0 \le p < p_{att1} \\ -B_2 & p_{att1} \le p < p_{att2} \\ -B_3 & p_{att2} \le p < p_{att} \\ 0 & p \ge p_{att} \end{cases}, \tag{9}$$

where $p \in [0, 1]$ is a random number, B_1, B_2, B_3 are the reduced blood values after an attack, and p_{att1} and p_{att2} are the threshold of probability.

Note that Figure 1 is shown from the red perspective, and the relative relationship between the two sides can also be defined from the blue perspective in the same way. In the view of the blue, distance vector $\mathbf{D}'$, attacking angle φ'_b, and escaping angle φ'_r are defined like Equations (4-8), and the attacking conditions are defined as

$$\begin{cases} D_{att,\min} \leqslant \|\mathbf{D}'\| \leqslant D_{att,\max} \\ \varphi'_b \leqslant \varphi_{att,\max} \\ \varphi'_r \leqslant \varphi_{esp,\max} \end{cases}. \tag{10}$$

2.3. Air combat process

For an episode of one-to-one short-range air combat, the red and the blue are initiated, then confront in the combat environment until the conditions for the end of the episode are satisfied [17]. An episode ends when the red or the blue is damaged or the maximum decision step $t_{\max}$ is reached. If the UAV is out of the range of height or its blood value B is reduced to zero, this UAV will be judged as damaged. There are three types of results for the red, win, loss and draw, at the end of an episode. The red wins over the blue only when the blue is damaged before reaching the $t_{\max}$. Similarly, the red is lost when it is damaged before the blue within $t_{\max}$ time steps. The draw result means that both sides are still undamaged when the maximum decision step $t_{\max}$ is reached. The duration for each step is t_{step}. At every step in an episode, the red and the blue will make a maneuver decision by its action policy respectively to get the $\mathbf{u}_r$ and $\mathbf{u}_b$. During a step, the UAVs will continually execute $\mathbf{u}_r$ and $\mathbf{u}_b$ until another maneuver decision-making starts. The flag *done* shows when the episode ends. In this paper, the action policy of the red is actor network which is updated by the PPO algorithm, and the blue's action policy is described in section 3.6. The air combat framework is shown in Figure 3 and the process in an episode is shown in Algorithm 1.

3. AIR COMBAT DECISION METHOD DESIGN

3.1. PPO algorithm and improvement

The PPO algorithm is a type of DRL algorithm that has been used in many types of problems. In this part, the basic PPO algorithm and its usage in short-range air combat are introduced.

Algorithm 1 Air combat process in an episode

Input: position: $\mathbf{p}_r$, $\mathbf{p}_b$; velocity: $\mathbf{v}_r$, $\mathbf{v}_b$; blood: $B_r = B_b = B_0$; number of steps: $t_{\max}$; step: $t = 1$; damage: $dam_r = dam_b = False$; height range: $[0, z_{\max}]$

Output: confrontation result in the view of the red

for all $t \leqslant t_{\max}$ **do**
  set $done = False$
  use the red's action policy to get $\mathbf{u}_r$
  use the blue's action policy to get $\mathbf{u}_b$
  update the red's and the blue's positions and velocities by Equation (1)
  if $z_r \notin [0, z_{\max}]$ or $B_r \leqslant 0$ **then**
    set $B_r = 0$ and $dam_r = True$
  end if
  if $z_b \notin [0, z_{\max}]$ or $B_b \leqslant 0$ **then**
    set $B_b = 0$ and $dam_b = True$
  end if
  if dam_r is $False$ **then**
    calculate φ_r and φ_b in the view of the red by Equations (4) and (6)
    if satisfy Equation (8) **then**
      get a random number $p \in [0, 1]$
      calculate ΔB by Equation (9)
      set $B_b = B_b + \Delta B$
    end if
  end if
  if dam_b is $False$ **then**
    calculate φ'_r and φ'_b in the view of the blue
    if satisfy Equation (10) **then**
      get a random number $p \in [0, 1]$
      calculate ΔB by Equation (9)
      set $B_r = B_r + \Delta B$
    end if
  end if
  if dam_r is $False$ and dam_b is $True$ **then**
    set $done = True$
    return red win
  else if dam_r is $True$ and dam_b is $False$ **then**
    set $done = True$
    return red loss
  else if dam_r is $True$ and dam_b is $True$ **then**
    set $done = True$
    return tie
  else if $t = t_{\max}$ **then**
    set $done = True$
    return tie
  else
    set $t = t + 1$
  end if
end for

3.1.1. The PPO algorithm

The PPO algorithm is based on the actor-critic framework and the policy gradient method (PG), which can be applied to continuous or discrete motion space problems.[18] PG-based algorithms maximize the action policy's expected return by updating the action policy directl[19]. The PPO algorithm's main objective is[18]

$$L(\theta) = \mathbb{E}\left[L_{actor}(\theta) - c_1 L_{critic}(\theta) + c_2 S_\theta(o_t)\right], \tag{11}$$

where L_{actor} and L_{critic} are the loss function for actor network and critic network, θ is the parameters of networks, c_1 and c_2 are coefficients, S_θ is the entropy bonus which is used to ensure sufficient exploration, and o_t is the observation of the actor network[18]. By considering these terms, the loss calculated by Equation (11) is related to the parameters of actor and critic networks. L_{actor} is defined as

$$L_{actor}(\theta) = \mathbb{E}\left[\min\left(r(\theta)\hat{A}, \text{clip}\left(r(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}\right)\right], \tag{12}$$

where $r(\theta) = \pi_\theta(a_t|o_t)/\pi_{\theta old}(a_t|o_t)$ is the probability ratio. It is the probability of a_t with o_t under latest action policy π_θ and action policy before update $\pi_{\theta old}$, is the clipping function which is to ensure the policy from $\pi_{\theta old}$ to π_θ doesn't change too much. $\hat{A}$ is the estimator of the advantage function, and the generalized advantage estimation (GAE) is used to calculate it[20]. The L_{critic} is defined as[18]

$$L_{critic}(\theta) = \frac{1}{2}(V_\theta(s) - R(s))^2, \tag{13}$$

where V_θ is the state-value function, which means the critic network in the PPO algorithm, and R is the return.

In this paper, the red's action policy is based on the PPO algorithm. Thus, $\mathbf{u}_r$ is generated based on the actor network. The output of actor network is a probability distribution $dist$, which is the selection probability of each action for the red under the observation. Then, the action a is sampled from the $dist$, which means the index of selected action in the action space[21]. Finally, $\mathbf{u}_r$ can be generated by the designed action space. Noting that the $\log\_prob$ is the logarithm of $dist$. The usage of the PPO algorithm in short-range air combat is shown as Algorithm 2[21].

3.1.2. Improved points

To improve the training effect, some improvement points are adopted in this paper, and the framework of PPO algorithm for short-range air combat is shown in Figure 4.

The first improvement point is considering the historical combat data when making decision. During the confrontation, the red must gradually accumulate the situational advantages over the blue and finally beat it. Therefore, decision on current action should take into account the previous air combat situations. The link between the action decision and historical air combat experience is established by adding the gate recurrent unit (GRU)[22] to the neural network. The GRU is a type of recurrent neural network (RNN), which can adaptively capture the time's dependencies in different scales[22], similar to long short-term memory (LSTM). However, the GRU is easier to train and more efficient than LSTM. The GRU layer, the hidden layer using GRU, is used in both actor network and critic network. The networks' inputs, observation and state, are firstly processed by the fully connected network to extract the inputs' features. Then the features are fused with historical features by the GRU layer to obtain the integrated features considering the historical situational features. The specific process is shown in Figure 4.

Algorithm 2 The usage of the PPO algorithm in short-range air combat

initialize the PPO's hyperparameters, including epoch K, the number of the minimum experience for training N, etc.
initialize the air combat environment including the UAVs' positions, velocities, blood values, etc.
initialize the number of experiences in experience buffer $i = 0$
for all $episode \in training\_episodes$ **do**
    execute the process shown in Algorithm 1, including updating the UAVs' positions, velocities, blood values and damage flags. For each time step, o, state and reward are generated, and the experiences are generated, then set $i = i + 1$
    if $i \geq N$ **then**
        calculate the return for every step and normalize the advantage
        set $k = 1$
        for all $k \leq K$ **do**
            sample from the experience buffer based on the batch size
            calculate the loss of each batch by Equation (11)
            update the networks' parameters by Adam optimizer
            set $k = k + 1$
        end for
        set $i = 0$ and clear the experience buffer
    end if
end for

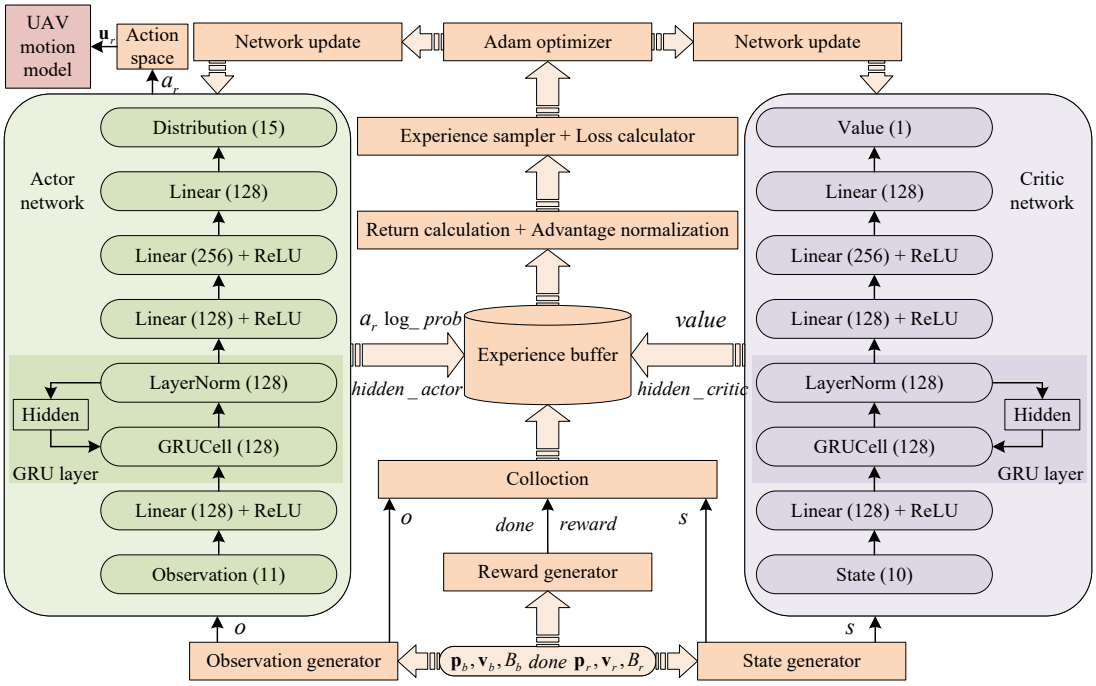

Figure 4. The PPO algorithm framework for short-range air combat.

The second improvement point is to differentiate the neural network inputs. For the basic PPO algorithm, the critic network uses the same input as the actor network, which is named state space[20]. However, the actor and critic networks play different roles in the algorithm. The actor network's state space is in the view of the red because the actor network's input is the red's observation of the air combat environment. On the other hand,

the critic network is used to evaluate the output of the actor network by its output, the critic value, based on the current air combat situation[23]. Thus, the input of critic network can be more objective and include some information that cannot be observed by the red. In this vein, the observation o from the observation generator and the state s from the state generator are designed for actor network and critic network respectively, as shown in Figure 4.

The third improvement point is a variable-sized experience buffer. For each time step, the experiences are stored in the experience buffer in order. During the training, it is divided into thousands of episodes, and each episode is over if it satisfies the ending condition. Thus, the length of each episode may be different. To make the training more general, the network parameters are updated until enough experiences are stored. However, the return calculation should be calculated on the basis of the complete experiences. Therefore, when the networks are updated, the numbers of experiences, which are larger than the number of the minimum experience for training N, are different to make sure that the same episode's experiences are stored in the buffer.

The fourth improvement point is the phased training process, which is discussed in section 3.6.

3.2. State space design

The state space of air combat should contain the information of the red and the blue UAVs, and a suitable designed state space can speed up the convergence of training. In this part, two state spaces designed for actor network and critic network are introduced.

3.2.1. The actor network's state space

The designed state space consists of two parts, position information s_{pos} and velocity information s_{vel} which is expressed as

$$s_{actor} = \left[s_{pos}^T, s_{vel}^T\right]^T. \tag{14}$$

The s_{pos} is the position relationship between the UAVs, which is defined as[24]

$$s_{pos} = [x_r - x_b, y_r - y_b, z_r, D, \theta_D, \psi_D]^T, \tag{15}$$

where the subscript r and b represent the information for the red and blue, $D = \|\mathbf{D}\|$, and θ_D and ψ_D are the flight-path angle and heading angle for $\mathbf{D}$ respectively, which are similar to the θ and ψ for $\mathbf{v}$ in Figure 2. The θ_D and ψ_D are described as[14]

$$\begin{aligned} \theta_D &= \arcsin\frac{z_b - z_r}{D}, \\ \psi_D &= \begin{cases} \operatorname{atan2}(y_b - y_r, x_b - x_r) & \operatorname{atan2}(y_b - y_r, x_b - x_r) > 0 \\ 2\pi + \operatorname{atan2}(y_b - y_r, x_b - x_r) & \operatorname{atan2}(y_b - y_r, x_b - x_r) < 0 \end{cases}, \end{aligned} \tag{16}$$

where $\operatorname{atan2}(y, x) \in [-\pi, \pi]$ returns the angle between $[x, y]^T$ and x axis. The s_{vel} is the velocity relationship between the UAVs, which is defined as[14]

$$s_{vel} = [\dot{x}_r - \dot{x}_b, \dot{y}_r - \dot{y}_b, \dot{z}_r - \dot{z}_b, \varphi_b, \varphi_r]^T. \tag{17}$$

At the same time, to avoid the difference between the values of each state variable from being too large to affect the learning efficiency of the network, the normalization for every state variable is adopted. The threshold vector δ_{actor} for state variables is selected as

$$mathbf{\delta}_{actor} = [D_{th}, D_{th}, H_{th}, D_{th}, \pi, 2\pi, v_{\max} - v_{\min}, v_{\max} - v_{\min}, v_{\max} - v_{\min}, \pi, \pi]^T, \tag{18}$$

where D_{th} and H_{th} are the threshold values for distance and height. Noting that the elements of δ_{actor} and s_{actor} correspond one to one. Then, the normalization is executed by[3]

$$o_i = \frac{s_{actor,i}}{\delta_{actor,i}} \cdot a - b \quad i = 1, 2, \cdots, 11, \tag{19}$$

where o is the normalized state vector, which is usually called the observation of the actor network. a and b are constants and satisfy $a = 2b$.

3.2.2. The critic network's state space

The actor network gets action according to the relationship in the view of the red, but the critic network gets the evaluation value based on the state of the air combat environment, which can include the information that cannot be observed. Thus, the input for the critic network s_{critic} is defined as[17]

$$s_{critic} = [D, \varphi_r, \varphi_b, z_b - z_r, \psi_D, \psi_{br}, \theta_{br}, v_r - v_b, B_r, B_{rb}]^T, \tag{20}$$

where subscript rb means the red's value minus the blue's and br is on the contrary, and B_r is the red's residual blood. The critic network's state variables are also normalized and the threshold vector δ_{critic} is selected as

$$\delta_{critic} = [D_{th}, \pi, \pi, H_{th}, 2\pi, \pi, \pi, v_{\max} - v_{\min}, B_0, B_0/2]^T. \tag{21}$$

Then, the s_{critic} is normalized like Equation (19), and the normalized result is s.

3.3. Action space design

In the air combat problem, the UAV's actions are always summed up as a maneuver library, which consists of a series of tactical actions, such as high yo-yo, cobra maneuvering and so on[3]. Pilots can choose from the library according to the combat situation. However, the establishment of the library is difficult and complex, and these tactical actions can be disassembled into basic actions. Thus, fifteen basic actions $a_1, a_2, \cdots, a_{15}$ are adopted to form the action space A, which includes five types of directions, forward, upward, downward, left turn and right turn, and three types of speed control, maintenance, acceleration and deceleration[25]. Every basic action in the designed library is a set of values of control parameters, $[n_x, n_z, \gamma]^T$, for the UAV motion model. The designed maneuver library is shown in Table 1. Therefore, the action space is discrete and its dimension is 15.

3.4. Reward function design

The aim of RL is to maximize the cumulative reward obtained from the environment. Therefore, the reward function is the bridge to communicate the training result requirements to the DRL algorithm and its design is extremely important[25]. In this paper, the reward function is well-designed and divided into three parts:

Table 1. The basic action's values in the designed action space

No.	Action	Values for $[n_x, n_z, \gamma]^T$	No.	Action	Values for $[n_x, n_z, \gamma]^T$
1	Forward, maintain	0; 1; 0	2	Forward, accelerate	2; 1; 0
3	Forward, decelerate	-1; 1; 0	4	Upward, maintain	0; 3.5; 0
5	Upward, accelerate	2; 3.5; 0	6	Upward, decelerate	-1; 3.5; 0
7	Downward, maintain	0; -3.5; 0	8	Downward, accelerate	2; -3.5; 0
9	Downward, decelerate	-1; -3.5; 0	10	Left turn, maintain	0; 3.5; arccos (2/7)
11	Left turn, accelerate	2; 3.5; arccos (2/7)	12	Left turn, decelerate	-1; 3.5; arccos (2/7)
13	Right turn, maintain	0; 3.5; - arccos (2/7)	14	Right turn, accelerate	2; 3.5; - arccos (2/7)
15	Right turn, decelerate	-1; 3.5; - arccos (2/7)			

dense reward, event reward and end-game reward. Different types of rewards are triggered under different conditions to transmit different expectations. In this part, the reward function designed for short-range air combat is introduced.

3.4.1. Dense reward

The red receives a dense reward from the air combat environment after completing the action for every step. Thus, a properly designed dense reward can improve the red's exploration efficiency and speed up the training process. The dense reward is based on the air combat situatio[16] after the execution of the red's and blue's actions and can be considered as the immediate situation value for the maneuvering decision-making. The dense reward r_{dense} is defined as

$$r_{dense} = (w_a r_a + w_d r_d + w_h r_h + w_v r_v - 1) \cdot w_{dense}, \tag{22}$$

where r_a, r_d, r_h and r_v are the angle reward, distance reward, height reward and velocity reward respectively, and w_a, w_d, w_h, w_v and w_{dense} are the weights. By giving a negative reward as a penalty term at every step, the red will try to find ways to reduce the penalty by trying its best to increase r_a, r_d, r_h and r_v as quickly as possible. Thereby, the red can be trained more efficiently. r_a represents the evaluation value of the azimuth relationship between the red and blue, and is defined as

$$r_a = \frac{\pi - \varphi_r}{\pi} \cdot \frac{\pi - \varphi_b}{\pi}. \tag{23}$$

r_d represents how good the distance relationship is, which consists of two parts, r_{d1} and r_{d2}, and satisfies $r_d = r_{d1} + r_{d2}$. r_{d1} is defined as

$$r_{d1} = \begin{cases} 0.25 & \Delta D < 0, D > D_{mid} \\ 0 & other \end{cases}, \tag{24}$$

where ΔD is the distance difference from the previous time, and $D_{mid} = \left(D_{att,\min} + D_{att,\max}\right)/2$ is the desired distance during air combat. This means the red can receive the reward r_{d1} only when the distance D is larger than D_{mid} and the red is closer to the blue than the previous time. r_{d2} is defined as[16]

$$r_{d2} = \begin{cases} 0.25 \cdot \left(a_{D1}\left(D - D_{adv,\max}\right)^2 + 1\right) & D_{adv,\max} < D \leqslant D_{th} \\ 0.25 + 0.25 \cdot \left(a_{D2}\left(D - D_{att,\max}\right)^2 + 1\right) & D_{att,\max} < D \leqslant D_{adv,\max} \\ 0.5 + 0.25 \cdot a_{D3}\left(D - D_{att,\min}\right)\left(D - D_{att,\max}\right) & D_{att,\min} < D \leqslant D_{att,\max} \\ 0 & other \end{cases}, \tag{25}$$

where a_{D1}, a_{D2} and a_{D3} are the parameters and defined as

$$\begin{aligned} a_{D1} &= -1 \Big/ \left(D_{th} - D_{adv,\max}\right)^2 \\ a_{D2} &= -1 \Big/ \left(D_{adv,\max} - D_{att,\max}\right)^2 \\ a_{D3} &= 1 \Big/ \left(\left(D_{mid} - D_{att,\min}\right)\left(D_{mid} - D_{att,\max}\right)\right) \end{aligned} . \tag{26}$$

r_h represents the height advantage and is defined as

$$r_h = \begin{cases} 0.1 & H_{\max} < z_r - z_b < D_{att,\max} \\ h_1(z_r - z_b - H_{adv})^2 + 1 & H_{adv} < z_r - z_b \leqslant H_{\max} \\ 1 & H_{att} < z_r - z_b \leqslant H_{adv} \\ h_2(z_r - z_b - H_{att})^2 + 1 & H_{\min} < z_r - z_b \leqslant H_{att} \\ 0 & other \end{cases} , \tag{27}$$

where $H_{\max}$, H_{adv}, H_{att} and $H_{\min}$ are maximum desired height, desired advantage height, desired attacking height and minimum desired height during the combat, and they satisfy $H_{\max} > H_{adv} > H_{att} > H_{\min}$. h_1 and h_2 are parameters that are defined as

$$\begin{aligned} h_1 &= -0.9 \Big/ \left(H_{\max} - H_{adv}\right)^2 \\ h_2 &= -1 \Big/ \left(H_{\min} - H_{att}\right)^2 \end{aligned} . \tag{28}$$

As for r_v, it is the evaluation of the velocity for both sides and is defined as

$$r_v = \begin{cases} 0.1 & v_r/v_b > 1.5 \\ 1 & 1.0 \leqslant v_r/v_b < 1.5 \\ 5 \cdot v_r/v_b - 4 & 0.8 \leqslant v_r/v_b < 1.0 \\ 0 & other \end{cases} . \tag{29}$$

3.4.2. Event reward

During air combat, there are many types of events [24,26], such as attacking successfully, reaching the advantage area, making the enemy in the attacking area and so on. By continuously triggering these events, the red will beat the blue finally. Thus, the event reward is necessary to make the red consciously trigger these events to maintain the advantage. This paper designs two types of event rewards: advantage area reward r_{adv} and attacking reward r_{att}. r_{adv} encourages the red for getting in the advantage area, and is defined as

$$r_{adv} = w_{adv} \cdot \left(0.6 \cdot \frac{D_{adv,\max} - D}{D_{adv,\max} - D_{adv,\min}} + 0.4 \cdot \frac{\pi - \varphi_r}{\pi}\right), \tag{30}$$

where w_{adv} is the weight. By giving a changeable r_{adv}, the red is encouraged to keep in the advantage area and get closer to the blue. Analogously, the blue can also be in the blue's advantage area, which is harmful to the red. In this situation, the penalty r'_{adv}, which satisfies $r'_{adv} < 0$ and $|r'_{adv}| > w_{adv}$, will be given to the red to

encourage it to turn over the situation as soon as possible. When the red succeeds to attack the blue, r_{att} will reward it. r_{att} is a positive const number. On the other hand, if the red is attacked by the blue, it will get the penalty r'_{att} which satisfies $r'_{att} < 0$ and $|r'_{att}| > r_{att}$.

3.4.3. End-game reward

During the training, the draw result is regarded as a case in which the red loses and is used to motivate the red to beat the blue. When the flag $done = True$, which means an episode is over, the end-game reward will be generated, and is defined as [25]

$$r_{end} = \begin{cases} r_0 + r_1 \cdot \dfrac{t_{\max} - t}{t_{\max}} + r_2 \cdot \dfrac{B_r}{B_0} & \text{win} \\ -r_{loss} & \text{loss} \end{cases}. \tag{31}$$

where r_0, r_1, r_2 and r_{loss} are positive numbers, and satisfy $r_0 + r_1 + r_2 \leqslant r_{loss}$.

3.5. Action policy for the Blue

During the training, a policy is adopted as the blue's action policy, which consists of prediction and decision. In the prediction step, the blue will predict which action the red will do at the next time step and then estimate the red's position and velocity at the next time step based on the predicted action. At the decision step, the blue will find which action it should take to confront the red. To find which action is better, the thread function T is defined [27], which consists of angle thread T_φ, velocity thread T_v, distance thread T_d and height thread T_h. Hence, T is calculated by

$$T = 0.41 \cdot T_\varphi + 0.26 \cdot T_v + 0.19 \cdot T_d + 0.14 \cdot T_h. \tag{32}$$

Noting that the definitions of T_φ, T_v, T_d and T_h are the same as in ref. [27]. The blue's action policy is described as Algorithm 3.

3.6. Training process

In the confrontation training, the red and the blue confront each other for thousands of episodes in the air combat environment. Every episode works as Algorithm 1 and for every step the blue makes the maneuver decision as Algorithm 3. To train the red's action policy with the DRL algorithm, the experience for every step is stored. When satisfying the training conditions, the experiences will be used to update the red's action policy. To make sure the training is successful and to obtain satisfactory results, the training is divided into three phases, basic, dominant and balanced [3]. The initial states for these phases are shown in Table 2.

The three phases constitute a progressive relationship, which means the later training is based on the training results of the former training. The red's actor network and critic network are loaded with the former trained networks' parameters before starting the training.

4. RESULTS

4.1. Parameters setting

The hyperparameters setting for the DRL algorithm is shown in Table 3 [18].

The parameters of the designed air combat environment are set as follows [15]. For the attacking distance, it is set as $D_{att,\min} = 40m$ and $D_{att,\max} = 900m$. The maximum attacking angle $\varphi_{att,\max} = \pi/6$ and the maximum escape

Algorithm 3 Action policy of the blue

Input: position: $\mathbf{p}_r$, $\mathbf{p}_b$; velocity: $\mathbf{v}_r$, $\mathbf{v}_b$; action space: A; prediction time: t_{pred}
Output: the blue's action $\mathbf{u}_b$

initialize the thread set Ξ
for all action a in A **do**
  calculate the red's $\mathbf{p'}_r$ and $\mathbf{v'}_r$ after t_{pred} with action a by Equation (1)
  calculate thread value by Equation (32) in the view of the red with $\mathbf{p'}_r$, $\mathbf{v'}_r$, $\mathbf{p}_b$ and $\mathbf{v}_b$
  append the thread value to Ξ
end for
set ind equal to the index of the maximum value in Ξ
set $\mathbf{u'}_r$ equal to ind-th element in A
calculate the red's $\mathbf{p'}_r$ and $\mathbf{v'}_r$ after t_{pred} with action $\mathbf{u'}_r$ by Equation (1)
initialize the thread set Ξ
for all action a in A **do**
  calculate the blue's $\mathbf{p'}_b$ and $\mathbf{v'}_b$ after t_{pred} with action a by Equation (1)
  calculate thread value by Equation (32) in the view of the blue with $\mathbf{p'}_r$, $\mathbf{v'}_r$, $\mathbf{p'}_b$ and $\mathbf{v'}_b$
  append the thread value to Ξ
end for
set ind equal to the index of the maximum value in Ξ
set $\mathbf{u}_b$ equal to ind-th element in A
return $\mathbf{u}_b$

Table 2. The initial states of the UAVs in three training phases

Phases \ Initial states	Camp	x (m)	y (m)	z (m)	v (m/s)	θ (deg)	ψ (deg)
Basic	Red	[200, 1800]	[1500, 3500]	[700, 2500]	60	0	$[-0.5\pi, 0.5\pi]$
	Blue	[1500, 2200]	[1500, 3500]	[700, 2500]	60	0	$[-0.3\pi, 0.3\pi]$
Dominant	Red	[200, 1800]	[1500, 3500]	[700, 2500]	60	0	$[-0.5\pi, 0.5\pi]$
	Blue	[1500, 2200]	[1500, 3500]	[700, 2500]	60	0	$[-0.3\pi, 0.3\pi]$
Balanced	Red	[800, 4800]	[0, 4500]	[800, 3500]	60	0	$[0, 2\pi]$
	Blue	[800, 4800]	[0, 4500]	[800, 3500]	60	0	$[0, 2\pi]$

Table 3. The hyperparameters setting for the DRL algorithm

Hyperparameter	Value	Hyperparameter	Value
Learning rate	0.00025	GAE paramerter	0.95
Discount	0.99	Minimum buffer size	$N = 8192$
Number of batches	4	Epoch	$K = 5$
Clip parameter	0.1	Main objective's coefficients	$c_1 = 0.5; c_2 = 0.01$

angle when attacking $\varphi_{esp,att} = \pi/3$. For the advantage area, it is set as $D_{adv,\min} = 40m$, $D_{adv,\max} = 1300m$ and $\varphi_{esp,\max} = \pi/3$. For the blood, the probabilities are set as $p_{att1} = 0.1$, $p_{att2} = 0.4$ and $p_{att} = 0.8$, and the damage values are set as $B_0 = 300$, $B_1 = 51$, $B_2 = 21$ and $B_3 = 11$. For an episode, the maximum decision step $t_{\max} = 400s$ and the time step $t_{step} = 0.5s$. The threshold values are set as $D_{th} = H_{th} = 5000m$. For the reward function, the weights are set as $r_a = 0.15$, $r_d = 0.6$, $r_v = 0.1$, $r_h = 0.15$, $w_{dense} = 0.05$ and $w_{adv} = 0.05$, the parameters about height are set as $H_{\max} = 500m$, $H_{adv} = 300m$, $H_{att} = 100m$ and $H_{\min} = -300m$, and the parameters in end-game reward are set as $r_0 = 5$, $r_1 = 3$, $r_2 = 6$ and $r_{loss} = 15$.

4.2. Training results in the phases

The four cases are trained with the hyperparameters in Table 3 and initial states in Table 2. Four cases are compared in this paper, which are case I PPO, case II PPO with GRU, case III PPO with GRU and state input

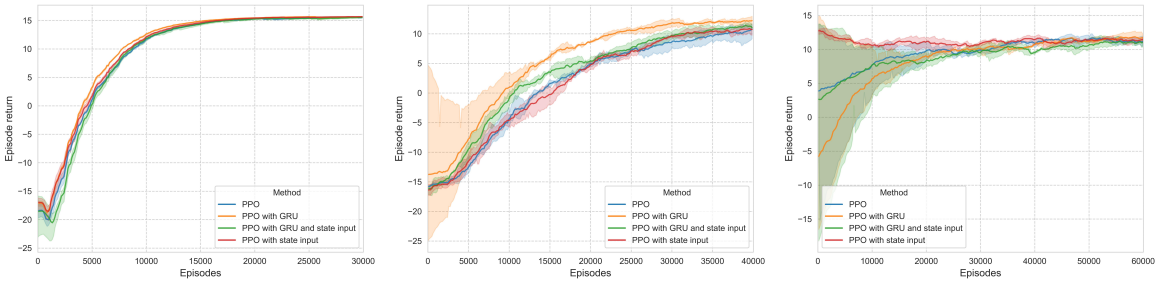

Figure 5. The episode returns of every episode in the three phases while training. A: the episode returns in the basic phase; B: the episode returns in the dominant phase; C: the episode returns in the balanced phase.

and case IV PPO with state input. The state input means the inputs of actor and critic networks are different, and if there is no GRU, the GRU layer in Figure 4 is replaced with linear layer with 128 units and ReLU.

In the basic phase, the UAVs' states are shown in the first line of Table 2. The blue always performs the forward and maintain action a_0 and the red is initialized behind the blue. Four cases are trained with 30000 episodes respectively. In the dominant phase, the UAVs' states are shown in the second line of Table 2. The blue uses the policy as Algorithm 2 and red is also initialized behind the blue. Four cases are trained with 40000 episodes respectively, and every case trains based on their own training result in the basic phase. In the balanced phase, the UAVs' states are shown in the third line of Table 2, and the initial states of the red and the blue are the same. Four cases are trained with 60000 episodes respectively, and every case trains based on their own training result in the dominant phase. The returns of every episode in the three phases are shown in Figure 5.

It can be seen that the PPO algorithm can go to converge faster when it is combined with GRU. In the basic phase, a relatively simple scenario, all four cases can easily find a better action policy to enclose the blue and beat it. And in the more complex scenario, it can find a better policy faster with GRU. But there are also larger episode reward variations in its infancy. Therefore, the state input is introduced to reduce the episode reward variations, which will synchronize to slightly reduce the final episode reward.

In addition, to test the final training result for the four cases, the training results in the balanced phase are reloaded and the initial states of the UAVs are set as

$$\begin{aligned} &x_r = 1000m, y_r = 2500m, z_r = 1200m, v_r = 90m/s, \theta_r = 0, \psi_r = 0 \\ &x_b = 4200m, y_b = 2500m, z_b = 1200m, v_b = 90m/s, \theta_b = 0, \psi_b = 0 \end{aligned}. \tag{33}$$

And the maneuvering trajectories are shown in Figure 6. The steps for the four cases are 146, 90, 123 and 193. For Figure 6B, it is shown that the blue and the red collide. It is obvious that the red can beat the blue in a smaller space range when trained with PPO with GRU, and by adding state input, the red can be more flexible to avoid collision. But the cost is the increase in time steps, which explains the decrease of final episode reward. In case III, the red uses hover and altitude variations to lure the blue closer and gain an advantage situation, instead of pursuing the blue.

4.3. Confrontation tests

To compare the final training result of four cases, the confrontation tests for the four cases are conducted in this part when all of the training is finish. To accelerate the confrontation tests, the B_0 is set as 100 during the test, and the UAVs' initial states are the same as they are in the balanced phases. The tie air combat result is

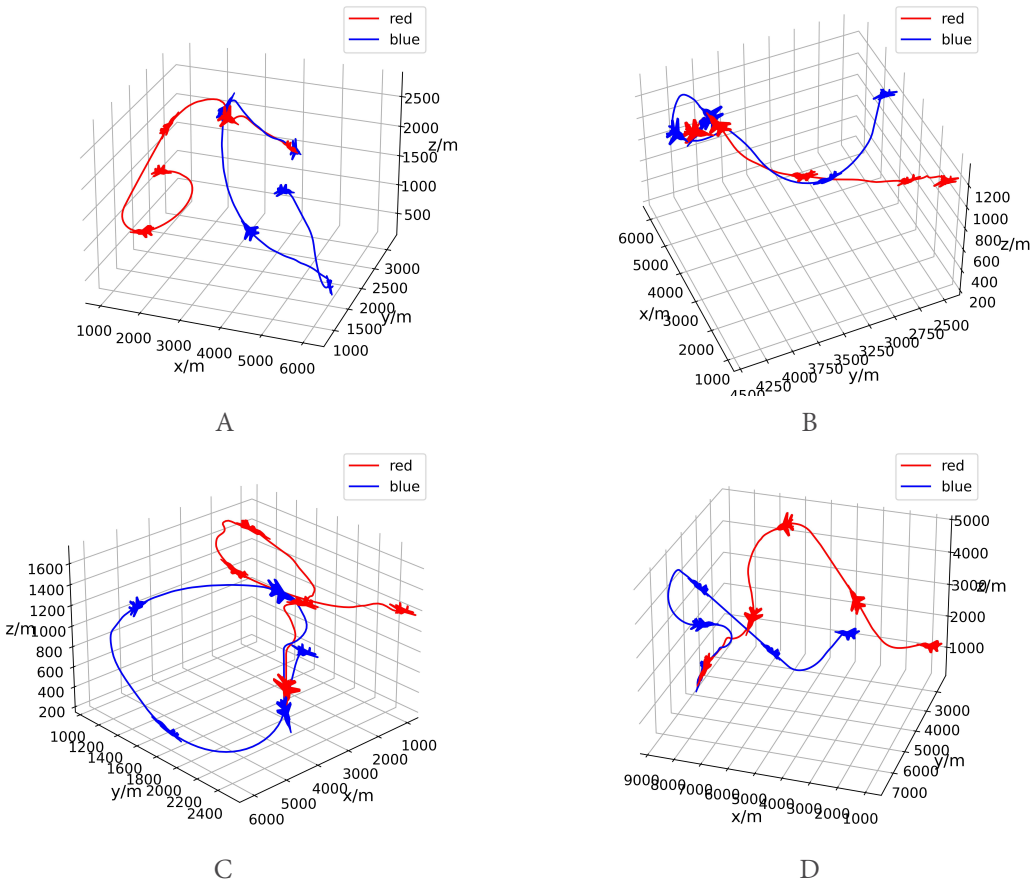

Figure 6. The maneuvering trajectories in the training test for four cases. A: the maneuvering trajectories for the case I PPO; B: maneuvering trajectories for case II PPO with GRU; C: maneuvering trajectories for case III PPO with GRU and state input; D: maneuvering trajectories for case IV PPO with state input.

Table 4. The confrontment test results between four cases

Algorithm case	Results (100 episodes)			
	Win rate	Loss rate	Tie rate	
			Tie win rate	Tie loss rate
PPO with GRU and state input *vs.* PPO	74%	12%	9%	5%
PPO with GRU and state input *vs.* PPO with GRU	57%	1%	17%	25%
PPO with GRU and state input *vs.* PPO with state input	52%	14%	14%	20%

divided into two types, tie win if $B_r > B_b$ and tie loss if $B_r \leqslant B_b$. All of the tests are conducted in the air combat environment for 100 episodes, and the results are shown in Table 4.

It can be seen that by combining PPO with GRU and state input, the UAV can get a more flexible and intelligent action policy even though the training process is the same. It is proved that training the action policy by the PPO with proposed improve points can help the UAV gain an advantage situation more quickly and greater operational capability in short-range air combat confrontation, and the action policy can be more intelligent to adapt to the blue's uncertain policy.

5. CONCLUSIONS

In this article, a maneuver decision method for UAV air combat is proposed based on the PPO algorithm. To enhance the PPO's performance, the GRU layer and different compositions of networks' inputs are adopted. At the same time, to accelerate training, some designs are applied. The action space is discretized into 15 basic actions, and the reward function is well-designed with three parts. Further, the training process is divided into several progressively more complex phases. To illustrate the advantages of the designed method, ablation experiments and UAV air combat tests are conducted in this paper. The episode rewards and confrontment test results show that the designed maneuver decision method can generate a more intelligent action policy for the UAV to win short-range air combat. By combining the PPO with the improved points, the training feasibility is improved and convergence is more efficient. The proposed maneuver decision-making method is always is able to achieve a win rate of more than 50% and a loss rate of less than 15%.

In future, the more complex six-degree-of-freedom UAV motion model and tighter UAV performance constraints could be introduced to improve accuracy. On the other hand, the multiple-to-multiple air combat problem, including multi-UAV coordinated attacking and tactical decisions, is the focus of future research.

DECLARATIONS

Authors' contributions

Made substantial contributions to the research, idea generation, software development, and conduct the DRL experiments. Wrote and edited the original draft: Zheng Z

Performed process guidance responsibility for the planning and execution of the study, as well as the evolution of overarching research aims, critical review, and material support. Review and revise the original draft: Duan H

Availability of data and materials

Not applicable.

Financial support and sponsorship

This work was partially supported by National Natural Science Foundation of China under grant #U20B2071, #91948204, #T2121003 and #U1913602.

Conflicts of interest

All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Ayass T, Coqueiro T, Carvalho T, Jailton J, Araújo J, Francês R. Unmanned aerial vehicle with handover management fuzzy system for 5G networks: challenges and perspectives. *Intell Robot* 2022;2:20-6. DOI
2. Zhang JD, Yang QM, Shi GQ, Lu Y, Wu Y. UAV cooperative air combat maneuver decision based on multi-agent reinforcement learning. *J Syst Eng Electron* 2021;6:1421–88. DOI

3. Yang QM, Zhang JD, Shi GQ, Hu JW, Wu Y. Maneuver decision of UAV in short-range air combat based on deep reinforcement learning. *IEEE Access* 2020;8:363–78. DOI
4. Ruan WY, Duan HB, Deng YM. Autonomous maneuver decisions via transfer learning pigeon-inspired optimization for UCAVs in dogfight engagements. *IEEE/CAA J Autom Sinica* 2022;9:1639–57. DOI
5. Yang Z, Zhou DY, Piao HY, Zhang K, Kong WR, Pan Q. Evasive maneuver strategy for UCAV in beyond-visual-range air combat based on hierarchical multi-objective evolutionary algorithm. *IEEE Access* 2020;8:46605–23. DOI
6. Xu GY, Liu Q, Zhang HM. The application of situation function in differential game problem of the air combat. 2018 Chinese Automation Congress (CAC); 2018 Nov 30-Dec 2; Xi'an, China. IEEE; 2019. pp. 1190–5. DOI
7. Başpınar B, Koyuncu E. Differential flatness-based optimal air combat maneuver strategy generation. AIAA Scitech 2019 Forum; 2019 Jan 7-11; San Diego, CA, USA. AIAA; 2019. pp. 1–10. DOI
8. Wang D, Zu W, Chang HX, Zhang J. Research on automatic decision making of UAV based on Plan Goal Graph. 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO); 2016 Dec 3-7; Qingdao, China. IEEE; 2016. pp. 1245–9. DOI
9. Özbek MM, Koyuncu E. Reinforcement learning based air combat maneuver generation; 2022. Available from: http://arxiv.org/abs/2201.05528. [Last accessed on 15 Mar 2023] DOI
10. Zhang YT, Zhang YM, Yu ZQ. Path following control for UAV using deep reinforcement learning approach. *Guid Navigat Control* 2021;1:2150005. DOI
11. Zhang H, He L, Wang D. Deep reinforcement learning for real-world quadrupedal locomotion: a comprehensive review. *Intell Robot* 2022;2:275–97. DOI
12. Boin C, Lei L, Yang SX. AVDDPG - Federated reinforcement learning applied to autonomous platoon control. *Intell Robot* 2022;2:145–67. DOI
13. Li YF, Shi JP, Jiang W, Zhang WG, Lyu YX. Autonomous maneuver decision-making for a UCAV in short-range aerial combat based on an MS-DDQN algorithm. *Def Technol* 2022;9:1697–714. DOI
14. Li Y, Han W, Wang YG. Deep reinforcement learning with application to air confrontation intelligent decision-making of manned/unmanned aerial vehicle cooperative system. *IEEE Access* 2020;8:67887–98. DOI
15. Li LT, Zhou ZM, Chai JJ, Liu Z, Zhu YH, Yi JQ. Learning continuous 3-DoF air-to-air close-in combat strategy using proximal policy optimization. 2022 IEEE Conference on Games (CoG); 2022 Aug 21-24; Beijing, China. IEEE; 2022. pp. 616–9. DOI
16. Kang YM, Liu Z, Pu ZQ, Yi JQ, Zu W. Beyond-visual-range tactical game strategy for multiple UAVs. 2019 Chinese Automation Congress (CAC); 2019 Nov 22-24; Hangzhou, China. IEEE; 2019. pp. 5231–6. DOI
17. Ma XT, Xia L, Zhao QC. Air-combat strategy using deep Q-learning. 2018 Chinese Automation Congress (CAC); 2018 Nov 30-Dec 2; Xi'an, China. IEEE; 2019. pp. 3952–7. DOI
18. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms; 2017. Available from: http://arxiv.org/abs/1707.06347. [Last accessed on 15 Mar 2023] DOI
19. Wu JT, Li HY. Deep ensemble reinforcement learning with multiple deep deterministic policy gradient algorithm. *Math Probl Eng* 2020;2020:1–12. DOI
20. Yuksek B, Demirezen MU, Inalhan G, Tsourdos A. Cooperative planning for an unmanned combat aerial vehicle fleet using reinforcement learning. *J Aerosp Inform Syst* 2021;18:739–50. DOI
21. Xing JW. RLCodebase: PyTorch codebase for deep reinforcement learning algorithms; 2020. Available from: https://github.com/KarlXing/RLCodebase. [Last accessed on 15 Mar 2023]
22. Chung JY, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling; 2014. Available from: http://arxiv.org/abs/1412.3555. [Last accessed on 15 Mar 2023] DOI
23. Pope AP, Ide JS, Micovic D, et al. Hierarchical reinforcement learning for air-to-air combat; 2021. Available from: https://arxiv.org/abs/2105.00990. [Last accessed on 15 Mar 2023] DOI
24. Sun ZX, Piao HY, Yang Z, et al. Multi-agent hierarchical policy gradient for air combat tactics emergence via self-play. *Eng Appl Artif Intel* 2021;98:104112. DOI
25. Hu JW, Wang LH, Hu TM, Guo CB, Wang YX. Autonomous maneuver decision making of dual-UAV cooperative air combat based on deep reinforcement learning. *Electronics* 2022;11:467. DOI
26. Jing XY, Hou MY, Wu GL, Ma ZC, Tao ZX. Research on maneuvering decision algorithm based on improved deep deterministic policy gradient. *IEEE Access* 2022;10:92426-45. DOI
27. Yang AW, Li ZW, Li B, Xi ZF, Gao CQ. Air combat situation assessment based on dynamic variable weight. *Acta Armamentarii* 2021;42:1553-63. (in Chinese) DOI