drones

MDPI

Article

Hierarchical Maneuver Decision Method Based on PG-Option for UAV Pursuit-Evasion Game

Bo Li [1], Haohui Zhang [1], Pingkuan He [1], Geng Wang [1,*], Kaiqiang Yue [1] and Evgeny Neretin [2]

[1] School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China; libo803@nwpu.edu.cn (B.L.); zhanghaohui@mail.nwpu.edu.cn (H.Z.); npuhpk@163.com (P.H.); ykq15929955434@163.com (K.Y.)

[2] School of Robotic and Intelligent Systems, Moscow Aviation Institute, 125993 Moscow, Russia; e.s.neretin@mai.ru

* Correspondence: wanggeng@nwpu.edu.cn; Tel.: +86-133-8922-3600

Abstract: Aiming at the autonomous decision-making problem in an Unmanned aerial vehicle (UAV) pursuit-evasion game, this paper proposes a hierarchical maneuver decision method based on the PG-option. Firstly, considering various situations of the relationship of both sides comprehensively, this paper designs four maneuver decision options: advantage game, quick escape, situation change and quick pursuit, and the four options are trained by Soft Actor-Critic (SAC) to obtain the corresponding meta-policy. In addition, to avoid high dimensions in the state space in the hierarchical model, this paper combines the policy gradient (PG) algorithm with the traditional hierarchical reinforcement learning algorithm based on the option. The PG algorithm is used to train the policy selector as the top-level strategy. Finally, to solve the problem of frequent switching of meta-policies, this paper sets the delay selection of the policy selector and introduces the expert experience to design the termination function of the meta-policies, which improves the flexibility of switching policies. Simulation experiments show that the PG-option algorithm has a good effect on UAV pursuit-evasion game and adapts to various environments by switching corresponding meta-policies according to current situation.

Keywords: UAV pursuit-evasion game; hierarchical reinforcement learning; meta-policy; policy gradient

Citation: Li, B.; Zhang, H.; He, P.; Wang, G.; Yue, K.; Neretin, E. Hierarchical Maneuver Decision Method Based on PG-Option for UAV Pursuit-Evasion Game. *Drones* **2023**, *7*, 449. https://doi.org/10.3390/drones7070449

Academic Editor: Diego Gonzalez-Aguilera

Received: 23 April 2023
Revised: 30 June 2023
Accepted: 4 July 2023
Published: 6 July 2023

1. Introduction

Unmanned aerial vehicles (UAVs) [1–7] are used in many fields, such as intelligent confrontation [8], target rounding [9] and intelligent transportation [10], because of their characteristics of being unmanned, having good concealment and having no casualties. UAV pursuit-evasion [11] involves a game between two UAVs with competing interests. In the process of UAV pursuit-evasion, being able to make effective maneuvering decisions [12] to destroy the other side and capture the other side is the key to victory. Among these, the real-time intelligent maneuvering decision-making ability of UAV is the core of problem solving. The maneuvering decision-making mechanism reflects the intelligence level of a UAV in the pursuit-evasion game. Therefore, it is necessary to design an effective maneuvering policy in the process of the UAV pursuit-evasion game.

At present, decision algorithms in UAV pursuit-evasion mainly include differential game theory [13], influence graph method [14], heuristic search algorithm [15], etc. F. Yu et al. [13] take into account the impact of environmental impediments in the pursuit-evasion game between UAVs and UGVs, qualitatively assess the pursuit problem of the difference game and use the differential game in the pursuit-evasion game. Q. Pan et al. [14] propose a cooperative maneuver decision method for multiple unmanned aerial vehicles based on the influence graph theory. A state predicted influence diagram model is used to analyze elements, and an unscented Kalman filter model is used for belief state updating. Mikhail et al. [15] propose schemes to solve the pursuit-evasion problem using Apollonius

circles and UAVs in a non-deterministic environment. However, the disadvantages of traditional UAV pursuit-evasion decision methods are that the rule base is complicated and difficult to cover all pursuit-evasion situations, and it has poor flexibility.

Since the 21st century, the development of artificial intelligence [16] has reached its climax. Artificial intelligence methods such as deep learning [17] and reinforcement learning [18] have been well applied in many high-tech fields. The application of artificial intelligence to UAV pursuit-evasion has also achieved good results. Researchers regard UAV as an agent in reinforcement learning, enabling them to gradually acquire optimized polices in complex environments after a certain stage of trial-and-error learning in the environment [19]. Deep reinforcement learning combines the perceptual ability of deep learning with the decision-making ability of reinforcement learning in a general form, providing a way to solve the problem of UAV pursuit-evasion [20–24]. Zhang et al. [20] design a guiding reward function based on the Deep Deterministic Policy Gradient (DDPG) algorithm for the UAV pursuit-evasion task and introduce a soft update strategy based on a sliding average. A UAV swarm can successfully carry out pursuit missions. However, the game between multi-UAVs and one UAV results in a significant advantage gap between the opposing sides. Additionally, the two-dimensional simulation environment differs greatly from actual pursuit-evasion tasks. Zhang et al. [21] construct a multi-agent coronal bidirectional coordinated target prediction network (CBC-TP network) by vectorial extension of the multi-agent depth Deterministic strategy gradient formula, so as to realize the process of UAV pursuit-evasion. However, the pursuit-evasion game environment is singular, and this paper does not comprehensively consider various situations. Fu et al. [22] propose IL-DDPG, which combines the DDPG algorithm and imitation learning algorithm. At the same time, the proportional guidance law is introduced as a guidance strategy. Compared with the DDPG algorithm, it improves the search efficiency and realizes the fast tracking of a pursuing UAV to avoid a UAV. However, the two-dimensional simulation environment differs significantly from the real pursuit-evasion environment. Additionally, the trained UAV can only perform the task of pursuit, resulting in limited functionality. Sun et al. [23] study the pursuit-evasion game problem of multiple UAVs with multiple static obstacles in a two-dimensional bounded environment and propose a multi-agent deep deterministic policy gradient based on attention algorithm to solve the pursuit-evasion problem of multi-UAVs. However, the paper does not take into account the threat posed by an opponent UAV, and there is a lack of a game process between the opponent UAV and our UAV. Vlahov et al. [24] discuss a framework for the development of reactive strategies that can learn to exploit behaviors, apply the A3C algorithm to UAV pursuit decision-making and verify the effectiveness of learning strategies through Monte Carlo experiments. However, it does not consider the generalization of the algorithm. When the environment changes, the performance advantage of the algorithm is not manifested. Additionally, it also does not comprehensively consider multiple situational scenarios.

Although reinforcement learning has a good performance in the UAV pursuit-evasion game, it also faces a problem: in the training process, due to the large state space, it may lead to dimensional disaster, which in turn affects the convergence speed of training. In addition, the pursuit-evasion environment changes in real time. The above algorithms do not comprehensively consider the overall situation, and only one strategy is trained so that the trained UAV can only complete the task of pursuit or evasion, and its performance is relatively simple, which cannot realize the function of turning the defeat into victory.

At present, hierarchical reinforcement learning algorithms are widely used in many fields [25–28]. Hierarchical reinforcement learning is for the performance of multiple sub-tasks in a hierarchical manner, which improves the decision-making efficiency of the model. Based on the above motivations, this paper considers applying the hierarchical reinforcement learning algorithm to the intelligent decision-making of UAV pursuit-evasion. Based on the framework of hierarchical reinforcement learning based on option [29], this paper chooses to use a policy gradient (PG) algorithm to train the top-level policy selector and proposes a UAV hierarchical maneuver decision-making method based on the PG-

option. Four options of the advantage game (AG), quick escape (QE), situation change (SC) and quick pursuit (QP) are designed by considering various situations in the UAV pursuit-evasion game comprehensively, and the corresponding meta-policy is trained by the SAC algorithm. The meta-policy termination function is designed to improve the flexibility of meta-policy switching. The experimental results show that the UAV trained by the PG option can flexibly switch meta-policies in the pursuit-evasion game and can use different meta-policy combinations to deal with different complex scenarios, reflecting the superiority and robustness of hierarchical maneuver decision-making method based on the PG option in the future.

The main contributions of this paper are summarized as follows:

(1) Comprehensively considering the constraint information in the process of UAV pursuit-evasion, this paper conducts a flight control model for a UAV and introduces the concept of threat zone, which makes the simulation more realistic;
(2) Considering various situations in the process of UAV pursuit-evasion comprehensively, four meta-policies are designed for UAV flight decision-making, which not only enrich the maneuver library but also effectively improve the effectiveness of the algorithm;
(3) A hierarchical maneuvering decision-making method for the UAV is designed to ensure that the UAV can flexibly switch meta-polices under different situations and achieve victory.

2. Problem Formulation and Preliminaries

In this section, to simplify the UAV pursuit-evasion scenario and subtract some unnecessary influencing factors, some assumptions are made for the scenario. The UAV flight control model and threat zone model are established for subsequent experiments.

2.1. Scenario Description and Assumptions

In this paper, we focus on the problem of UAV maneuvering decisions in a one-to-one UAV pursuit-evasion game. In the process of pursuit-evasion, many factors may affect the flight of UAV, which can cause the model to be more complex. However, it is not necessary to take all factors into account. Therefore, to pay more attention to our research and simplify the complexity of scenario, this paper proposes the following assumptions:

(a) The UAV is assumed to be a rigid body, and the gravity acceleration is a unified value;
(b) This paper ignores the influence of earth curvature, earth rotation and earth revolution on the UAV flight;
(c) To simplify the complexity of the UAV flight, this paper only considers the kinematics model.

The scenario is defined as a three-dimensional pursuit-evasion scenario. Therefore, a three-dimensional kinematics model of a UAV is established, and the UAV is described by physical quantities such as position, speed and attitude. The OXYZ northeast coordinate system is established. The coordinate origin is O, which indicates the center point of the scenario. The X-axis points to the north direction; the Z-axis points to the east direction; and the Y-axis points to the vertical direction. The situation diagram of both sides in scenario is shown in Figure 1.

The situation information of the red UAV is $\vec{R}_m = (X_m, Y_m, Z_m)$ and $\vec{V} = (v_{xm}, v_{ym}, v_{zm})$. The situation information of the blue UAV is $\vec{R}_t = (X_t, Y_t, Z_t)$ and $\vec{V}_t = (v_{xt}, v_{yt}, v_{zt})$.

The relative position vector of red UAV and the blue UAV is $\vec{D}$, and the distance scalar is d. The azimuth angle q represents the angle between the red UAV's velocity $\vec{V}$ and $\vec{D}$. The formula is as follows:

$$\vec{D} = (X_t - X_m, Y_t - Y_m, Z_t - Z_m) \tag{1}$$

$$q = \text{acos}\left(\frac{\vec{D} \cdot \vec{V}}{\left\|\vec{D}\right\| \cdot \left\|\vec{V}\right\|}\right) = \text{acos}\left(\frac{(X_t - X_m)v_{xm} + (Y_t - Y_m)v_{ym} + (Z_t - Z_m)v_{zm}}{d * \left(\sqrt{v_{xm}^2 + v_{ym}^2 + v_{zm}^2}\right)}\right) \quad (2)$$

$$d = \left\|\vec{D}\right\| \quad (3)$$

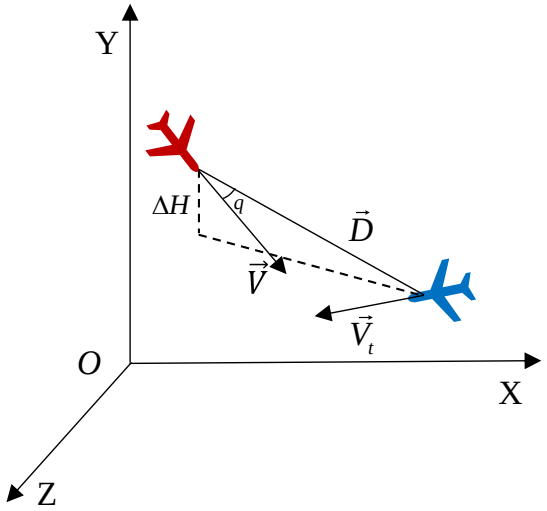


Figure 1. The situation in UAV pursuit-evasion.

2.2. UAV Model

It is assumed that our UAV can obtain the position and speed of the opponent through the ground command center. Our UAV needs to evaluate the current situation and make maneuvering decisions to effectively capture the opponent.

In the process of UAV kinematics modeling, it is considered a rigid body and regarded as a mass point. The kinematics equation of a three-degree-of-freedom UAV is as follows:

$$\begin{cases} X_{t+dT} = X_t + V_{t+dT} * \cos(\theta_{t+dT}) * \cos(\varphi_{t+dT}) * dT \\ Y_{t+dT} = Y_t + V_{t+dT} * \sin(\theta_{t+dT}) * dT \\ Z_{t+dT} = Z_t + V_{t+dT} * \cos(\theta_{t+dT}) * \sin(\varphi_{t+dT}) * dT \end{cases} \quad (4)$$

where φ denotes the heading angle; θ denotes the pitch angle; V is the speed of the UAV; $[X, Y, Z]$ denotes the component of the UAV in coordinate axes.

To make the model more real, this paper sets the threat range with the UAV as the center. The threat range is a conical area centered on the UAV and formed by the deviation of the UAV's speed direction from the $q_{\max}$ angle, which is called the threat zone in this paper. The threat zone contains three important indicators: maximum threat distance $D_{\max}$, minimum threat distance $D_{\min}$ and maximum threat angle $q_{\max}$. The two-dimensional diagram of the threat zone is shown in Figure 2.

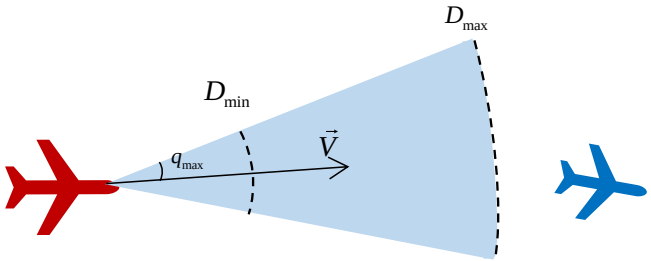


Figure 2. Threat zone in 2D.

When the distance between the two sides is less than the minimum threat distance $D_{\min}$, it is considered that the two sides collide and crash, and our UAV failed. When the continuous flight time of blue UAV in our threat zone that is defined as t_{in} exceeds the maximum time threshold $t_{\max}$, it is considered that the red UAV successfully captured the blue UAV. The formula is as follows:

$$\begin{cases} t_{in} > t_{\max} \\ q < q_{\max} \\ D_{\min} < \left|\vec{D}\right| < D_{\max} \end{cases} \tag{5}$$

3. Hierarchical Maneuver Decision Method for UAV Pursuit-Evasion Game

In this section, a hierarchical maneuver decision-making method for the UAV pursuit-evasion game based on the PG option is proposed. Considering the situation of both sides comprehensively, UAV maneuver decision-making be divided into the four meta-policies of advantage game (AG), quick escape (QE), situation change (SC) and quick pursuit (QP). Because different meta-polices perform different maneuvering characteristics, the corresponding reward function is designed and trained by the SAC algorithm. Then, the PG algorithm is integrated into the traditional hierarchical reinforcement learning to train the upper policy selector, and the expert experience is introduced to design meta-policy termination function. The structure of hierarchical decision-making is shown in Figure 3. The hierarchical decision-making model designed in this paper consists of two policy layers. There are four meta-polices at the bottom of the hierarchical decision model. The meta-policies are pre-trained so that they can perform corresponding actions according to the current pursuit-evasion situation when they are selected. There is a policy selector at the top of the hierarchical decision model, which is used to select the corresponding meta-policy according to the current situation.

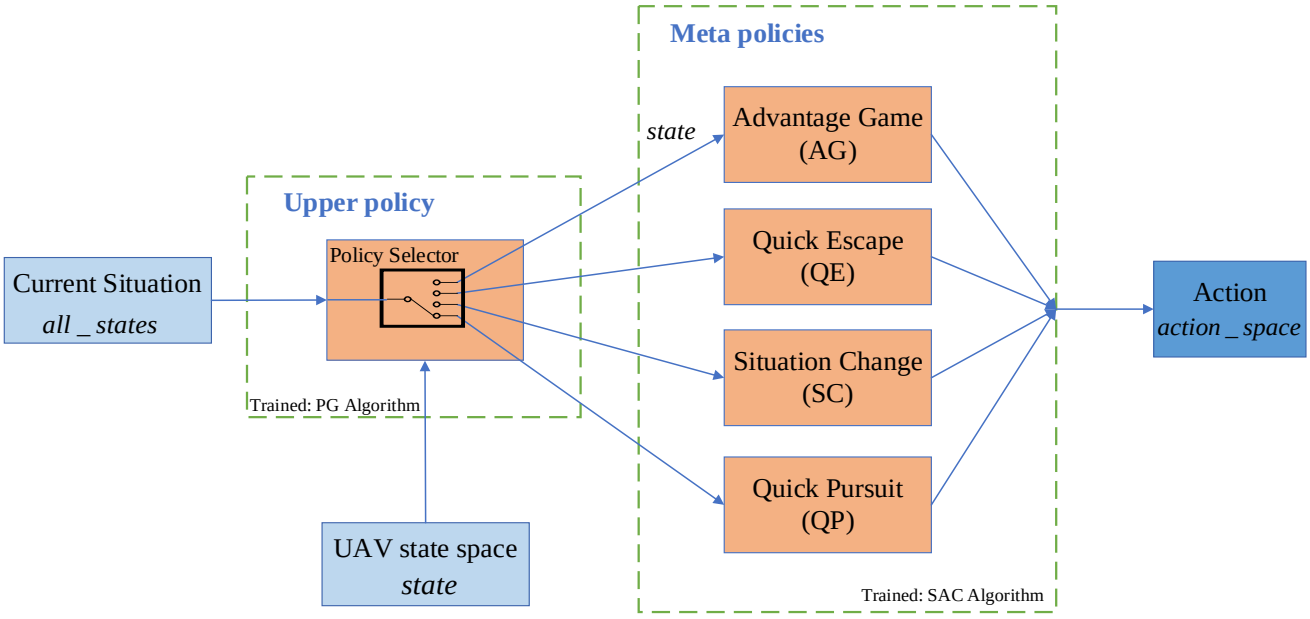


Figure 3. The structure of hierarchical decision.

3.1. Meta-Policy Model Training Algorithm Based on SAC

For the four meta-policies training of AG, QE, SC and QP, the paper uses the maximum entropy Soft Actor-Critic (SAC) algorithm to train them. SAC is a classical reinforcement learning algorithm. The SAC [30–33] algorithm is an algorithm based on the Actor-Critic framework, which can randomize the policy. At the same time, entropy is introduced to represent the randomness of the policy. After being trained, the agent will achieve a balance

between the reward value and the entropy, so that the agent can increase the exploration of the state space while maximizing the reward and achieving the purpose of accelerating the learning speed.

The four meta-policies all use the same state space, which is represented by nine tuples, as shown in Formula (6).

$$state = [X_m, Y_m, Z_m, v, \theta, \varphi, d, q_m, q_t] \tag{6}$$

where X_m, Y_m, Z_m represents the projection of UAV 's position on three coordinate axes; v is the speed of our UAV; θ denotes the pitch angle of our UAV; φ is the heading angle of the UAV; d denotes the distance between the opponent and our UAV; q_m is the relative azimuth of the opponent UAV related to our UAV; q_t is the relative azimuth of our UAV related to opponent UAV. q_t and q_m are shown in Figure 4, where the blue side is the opponent UAV and the red side is our UAV.

Figure 4. Relative azimuth.

Maneuvering speed change rate, pitch angle change rate and heading angle change the rate control of our UAV. The action space of each meta-policy can be represented by triples in the form of Formula (7).

$$action\_space = [\dot{v}, \dot{\theta}, \dot{\varphi}] \tag{7}$$

where $\dot{v}$ denotes the acceleration; $\dot{\theta}$ denotes the change rate of pitch angle; $\dot{\varphi}$ denotes the change rate of heading angle.

When using a reinforcement learning algorithm to train agents, the design of the reward function is often very important for training results. A good reward function is often more intuitive and simple, which can make the agent move in the optimal direction. Different reward functions need to be set for different meta-policy to explore the corresponding optimal policy.

To facilitate the setting of different reward functions for the four meta-policy trainings, the following reward and penalty items are set:

- R_{mq1} is the continuous angle penalty value for our UAV, which will continuously penalize the agent in a round. The equation R_{mq1} is: $R_{mq1} = -q_m/180$;
- R_{mq2} is the sparse angle reward value for our UAV, which will reward the agent under certain conditions in a round. The equation R_{mq2} is as follows: $R_{mq2} = 1, if\ q_m < q_{\max}$;
- R_{tq1} is the continuous angle penalty value for opponent UAV, which will continuously penalize the agent in a round. The equation R_{tq1} is as follows: $R_{tq1} = -q_t/180$;
- R_{tq2} is the sparse angle reward value for opponent UAV, which will reward the agent under certain conditions in a round. The equation R_{tq2} is as follows: $R_{tq2} = 1, if\ q_t < q_{\max}$;
- R_{d1} will reward the agent when the opponent UAV is in the threat zone of our UAV. The equation R_{d1} is as follows: $R_{d1} = 1\ , if\ D_{\min} < d < D_{\max}$;
- R_{d2} will penalize the agent when the opponent UAV is out of the threat zone of our UAV. The equation R_{d2} is as follows: $R_{d2} = -3,\ if\ d \geq D_{\max}\ or\ d \leq D_{\min}$;
- R_{d3} denotes the ratio of the distance between the two sides and maximum threat distance. The equation R_{d3} is as follows: $R_{d3} = d/D_{\max}$.

Then, we describe each meta-policy and set different reward function combinations for different meta-policies:

- Advantage Game (AG): AG refers to two UAVs in the same scenario, and they are in each other's threat zone. In this situation, our UAV needs to escape from the opponent's threat zone as soon as possible and keep the distance between two sides within a small range to facilitate our UAV to capture the opponent. Therefore, for the AG, the total reward is R_{AG}. The equation R_{AG} is as follows: $R_{AG} = w_1 * (R_{mq1} + R_{mq2}) - w_2 * (R_{tq1} + R_{tq2}) + w_3 * (R_{d1} + R_{d2} + R_{d3}/5)$. The paper holds that the reward of both sides is equally important to the task of AG, so $w_1 = w_2 = 0.4$. Furthermore, the primary task of AG is to ensure that UAV escapes from the opponent's threat zone quickly. Therefore, the weight of the distance reward is small; that is $w_3 = 0.2$.
- Quick Escape (QE): QE means that in the same scenario, our UAV is located in the opponent threat zone, while the opponent UAV is not in our threat zone, and the opponent's advantage is greater than our advantage. The primary task of our UAV is to maneuver quickly to escape the opponent's threat zone. For the QE task, the total reward is R_{QE}. The equation R_{QE} is as follows: $R_{QE} = w_1 * (R_{tq1} + R_{tq2}) - w_2 * (R_{d1} + R_{d3}/5)$. In the QE task, angle reward and distance reward are equally important; that is $w_1 = w_2 = 0.5$.
- Situation Change (SC): SC means that in the same scenario, both sides are not in the other's threat zone, but the opponent's advantage is greater than our advantage. Our primary task is to maneuver quickly to change the situation so that our advantage is greater than the opponent's advantage. For the SC task, the total rewards are R_{SC}. The equation R_{SC} is as follows: $R_{SC} = w_1 * (R_{mq1} + R_{mq2}) - w_2 * (R_{tq1} + R_{tq2}) + w_3 * R_{d3}$. This paper believes that our angle reward and distance reward are equally important to the SC task; that is $w_1 = w_3 = 0.4$, $w_2 = 0.2$.
- Quick Pursuit (QP): QP means that in the same scenario, both the opponent and our UAV are not in the opponent's threat zone, and our UAV's advantage is greater than the opponent UAV's advantage. The primary task of our UAV is to capture the opponent UAV. Our UAV needs to maneuver quickly to make the opponent UAV fall into our threat zone. The total rewards are R_{QP}. The equation R_{QP} is $R_{QP} = w_1 * (R_{mq1} + R_{mq2}) + w_2 * (R_{d1} + R_{d2}/5)$. In the QP task, angle reward and distance reward are equally important; that is $w_1 = w_2 = 0.5$.

The meta-policy uses the maximum entropy SAC algorithm for training. The network structure set in this paper is shown in Figure 5.

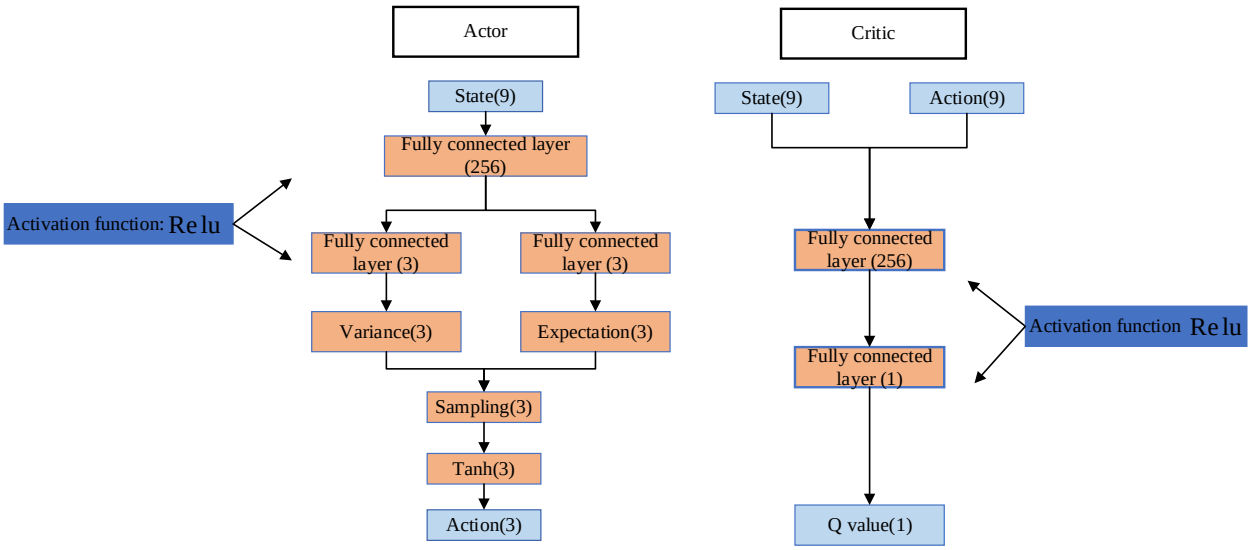


Figure 5. The structure of the Actor-Critic network.

The input of the Actor neural network is the state value of both sides, and the output is the speed change rate, pitch angle change rate and heading angle change rate of our UAV. The input of the Critic neural network is the state value and action value, and the output is the state action value (Q value). Experiments have shown that a wide and shallow fully connected neural network with the same number of neurons performs better than a narrow and deep network [34]. Therefore, both Actor and Critic networks are fully connected neural networks with two hidden layers, and the number of hidden layer neurons is 256. The activation function is Relu. In addition, the Actor neural network has two output layers, which are used to output the mean μ and variance σ, respectively, and the final action value can be obtained by sampling for agent execution.

Meta-policy training pseudo-code is shown in Algorithm 1.

Algorithm 1: Meta-Policy Training Algorithm

1. Randomly generated parameters: $\theta, \varphi_1, \varphi_2$
2. Initialize the policy network π_θ and two Soft-Q networks $Q_{\varphi_1}, Q_{\varphi_2}$
3. Initialize the target network of the Soft-Q network, and let $\varphi_1\prime \leftarrow \varphi_1, \varphi_2\prime \leftarrow \varphi_2$
4. **FOR** $t = 0, T$:
5. Get state s
6. **IF** $t <$ *start_size*:
7. $a = random()$
8. **ELSE**:
9. $\mu, \log\sigma = \pi_\theta(s), a = \tanh(\mu + \tau * \exp(\sigma))$
10. The agent performs actions a and gets a reward r and the next state $s\prime$
11. Store the array $< s, a, r, s\prime >$ in the experience pool
12. **IF** Experience storage in the experience pool > *batch_size*:
13. Sampling *batch_size* group data $< s, a, r, s\prime >$ from the experience pool
14. Update Q function network parameters: $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$
15. Update policy network's weights: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
16. Adjust the parameters of temperature: $\alpha \leftarrow \alpha - \lambda_\pi \hat{\nabla}_\alpha J(\alpha)$
17. Update the parameters of the target network: $\varphi_1{}' \leftarrow \varphi + (1-\tau)\varphi_1{}'$ $\varphi_2{}' \leftarrow \varphi + (1-\tau)\varphi_2{}'$
18. **END IF**
19. Jump to step 6, let $s \leftarrow s\prime$
20. **END IF**
21. **END FOR**

3.2. Hierarchical Decision Method Based on PG-Option

The traditional hierarchical reinforcement learning algorithm based on option divides the target task into multiple options. Each option has its policy, which is called the meta-policy. The meta-policy is selected by the top-level policy selector. In the traditional hierarchical reinforcement learning algorithm based on this option, the policy selector uses the ε-greedy algorithm to select the meta-policy with the highest reward return in the current state as the decision. The problem with ε-greedy algorithm is that it generally does not consider other possible situations from the whole. Each choice is only a local optimal solution, and it takes a long time to capture the opponent. However, in a complex scene, our UAV needs to remove danger as soon as possible to achieve victory.

To better solve the problem of the complex UAV pursuit-evasion game, the paper designs a hierarchical decision model based on the traditional hierarchical reinforcement learning algorithm framework and the policy gradient (PG) algorithm. Instead of the ε-greedy policy, the neural network trained by the PG algorithm is used as the policy selector. Compared with the ε-greedy algorithm, PG is easier to converge the training results and can handle high-dimensional continuous state data.

Since the policy selector evaluates the overall situation, the state space of the upper policy selector adds the position $[X_t, Y_t, Z_t]$ and speed v_t of the opponent UAV based on the meta-policy state space, which is shown in Formula (8).

$$all\_states = [X_m, Y_m, Z_m, X_t, Y_t, Z_t, v, v_t, \theta, \varphi, d, q_m, q_t] \tag{8}$$

The action space outputs the probability of four meta-policies. If a meta-policy is selected, the final reward value will increase, which will increase the probability of this probability, and vice versa.

When the policy selector model is trained, the reward function is shown in Formula (9).

$$R = \begin{cases} r(\tau)/total\_step & if \text{ Success} \\ -3 & if \text{ Failed} \end{cases} \tag{9}$$

where $r(\tau)$ is the sum of the rewards at the end of each round; $total\_step$ is the all simulation step size in each round.

The policy selector can only be rewarded at the end of that moment. The success mark is to capture the opponent UAV, and the failure mark is that the opponent UAV captures our UAV or our UAV fails to capture opponent UAV within the maximum number of rounds.

In the process of training, the upper policy selector model is optimized by maximizing the reward value, and the network parameter of the whole hierarchical model is set as $\theta = (\phi, \varphi)$. ϕ denotes the network parameters of the policy selector, and φ denotes the network parameters of the meta-policy. The expected reward of the pursuit-evasion process is shown in Formula (10).

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] = \int_{\tau \sim \pi_\theta(\tau)} \pi_\theta(\tau) r(\tau) \tag{10}$$

Next, we take the derivative of Formula (10) which is shown in Formula (11):

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)] \tag{11}$$

where $\pi_\theta(\tau)$ is as follows:

$$\pi_\theta(\tau) = p(s_{0,0}) \prod_{i=0}^{p-1} \pi_\phi(g_i | s_{i,0}) \prod_{j=0}^{T_i} \pi_\varphi(a_{i,j} | s_{i,g}, g_i) p(s_{i,j+1} | s_{i,j}, a_{i.j}) \tag{12}$$

Each round is divided into p stages. $s_{0,0}$ denotes the initial state of each round in UAV pursuit-evasion game, and $s_{i,0}$ denotes the initial state of the UAV in the first stage. At each stage, the UAV performs T_i steps, and $\pi_\phi(g_i | s_{i,0})$ denotes the probability that the meta-strategy g_i is selected by the policy selector. $\pi_\varphi(a_{i,j} | s_{i,j}, g_i)$ denotes the probability of the current action. $p(s_{i,j+1} | s_{i,j}, a_{i,j})$ denotes the state transition probability. In Section 2.1, the meta-policy is trained to converge, so the state transition probability is 1. Formula (12) can be rewritten:

$$\pi_\theta(\tau) = p(s_{0,0}) \prod_{i=0}^{p-1} \pi_\phi(g_i | s_{i,0}) \tag{13}$$

Then, we take the derivative of Formula (13) which is shown in Formula (14):

$$\nabla_\theta \log \pi_\theta(\tau) = \sum_{i=0}^{p-1} \nabla_\phi \log \pi_\phi(g_i | s_{i,0}) \tag{14}$$

Therefore, Formula (11) can be written as:

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\sum_{i=0}^{p-1} \nabla_\phi \log \pi_\phi(g_i | s_{i,0}) r(\tau)] \tag{15}$$

In the process of model training, the optimization goal is to maximize the gradient of Formula (15), to achieve the convergence of the policy selector model.

In addition, in the traditional hierarchical reinforcement learning algorithm based on this option, for each time the agent performs an action and obtains a new state, the policy will reselect the meta-policy according to the current state. However, in this way, when the situation is complex, the switching of meta-policy is too frequent, which makes it difficult for the UAV to meet such high-frequency switching, thus making it difficult to complete the task.

To solve the above problems, the policy selector performs periodic state evaluation at a frequency of 10 Hz and introduces expert experience to design meta-policies' termination functions to obtain the signs of meta-policy success and failure which are shown in Table 1. After the end, the next meta-policy is selected. The framework of the model training is shown in Figure 6.

Table 1. The signs of meta-policy success and failure.

Meta-Policy	Success	Failure
AG	Out of the opponent UAV's threat zone 1000 m	Failed to break away from the opponent threat zone within 500 steps
QE	Out of the opponent UAV's threat zone 2000 m	Failed to break away from the opponent threat zone within 500 steps
SC	The relative azimuth of the opponent is less than 25°	Our relative azimuth is not bigger than the opponent's relative azimuth within 500 steps
QP	Successfully capturing the opponent	The opponent failed to exist our threat zone within 500 steps

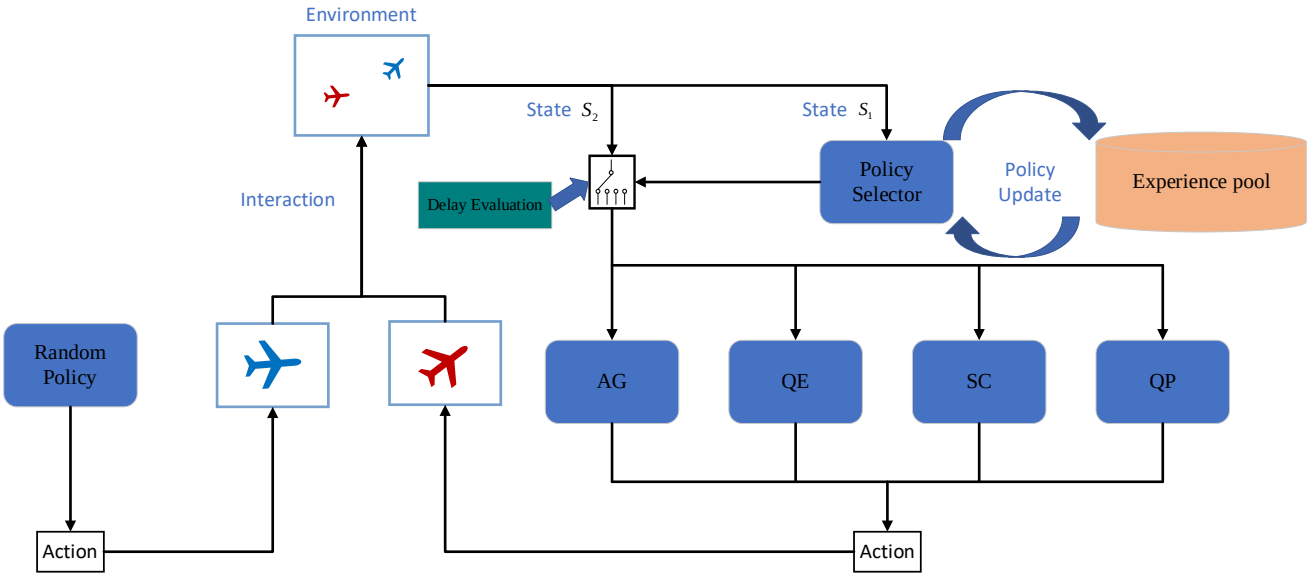


Figure 6. Training framework of hierarchical decision model.

The policy selector training pseudo-code is shown in Algorithm 2.

Algorithm 2: Hierarchical Maneuver Decision Algorithm Based on PG-Option

1. Randomly generate the policy selector network's parameters ϕ; Initialize the experience pool D
2. Initialize state variables $s_{Drevious}$, Initialize the initial state of the agent s_0, $s_0 = s_{Drevious}$
3. **FOR** $m = 1, M$:
4. Initialization counting stage: $p = 0$

Periodic selection of meta-policy g_π according to 10 Hz frequency
FOR t = 1, T:
  Generate actions a_t and execute actions; Get reward value r_t and new state s_{0+t}
  IF the current meta-policy execution ends:
  Save $(s_{Dreviously}, g_\pi, r_t)$ into the experience pool
  $s_{Dreviously} = s_{0+t}, p = p + l$
  END IF
  IF the end of the current round:
    IF capturing the opponent UAV:
      Update the reward value $r_t = r(\tau)/t$
    ELSE:
      Update the reward value $r_t = -3$
    END IF
    Update the network parameters of the policy selector ϕ, Clear the experience pool D
    Reinitialize the environment to obtain the initial state of the agent
  END IF
END FOR
END FOR

4. Experiments and Results

In this section, the proposed hierarchical maneuver decision method will be numerically simulated in many scenes. Firstly, the training parameters and hardware conditions of the simulation experiment are given. After the convergence of the four meta-policy trainings, the policy selector is trained and tested in the experimental scenario, which shows the feasibility of the method. Finally, the algorithm is tested by multiple UAV pursuit-evasion situations, and the simulation analysis is carried out to illustrate the generalization of the method.

4.1. Parameters and Hardware

In this paper, to avoid the instability of simulation results due to the large performance advantage gap between two UAVs, it is assumed that parameters of two UAVs are the same, as shown in Table 2. The heading angle variation is the range of the maximum allowable charge of the heading angle in the single-step execution of the UAV. The pitch angle variation is the range of the maximum allowable change of the pitch angle in the single-step execution of the UAV. The maximum speed of the UAV is the upper limit of the speed of the UAV. The maximum time threshold when the UAV falls into the other's threat zone is 2 s. Each step takes 0.1 s.

Table 2. The parameters of UAVs.

Parameter	Value
Our UAV pitch angle range	[0°, 4°]
Our UAV heading angle range	[0°, 10°]
Our UAV change range in speed	[0 m/s, 10 m/s]
Our UAV threat distance range	[1 km, 3 km]
Our UAV maximum speed	200 (m/s)
Our UAV Maximum threat angle	20 (°)
Our UAV Maximum time threshold	2 (s)
Opponent UAV pitch angle range	[0°, 4°]
Opponent UAV heading angle range	[0°, 10°]
Opponent UAV change range in speed	[0 m/s, 10 m/s]
Opponent UAV threat distance range	[1 km, 3 km]
Opponent UAV maximum speed	200 (m/s)
Opponent UAV Maximum threat angle	20 (°)
Opponent UAV Maximum time threshold	2 (s)

The four meta-policies are trained using the SAC algorithm, and the hyperparameters during training are as follows in Table 3. Batch_size is the number of samples extracted by the UAV from the experience pool during each training. The discount rate denotes the future reward discount when calculating the loss value. The optimization algorithm is the Adam algorithm. The regularization coefficient of entropy is initialized to 1. The target entropy is -3.

Table 3. The hyperparameters of SAC.

Hyperparameter	Value	Hyperparameter	Value
Actor-network learning rate	3×10^{-4}	Total number of rounds	10,000
Critic network learning rate	3×10^{-4}	Max steps in one round	500
Experience pool size	100,000	Soft update parameters	0.005
Batch_size	64	Discount rate	0.99
Target entropy value	-3	Regularization coefficient of entropy	1

When training the model, the Pytorch (v1.2.1) framework is used to design the neural network and optimizer. The CPU is an Intel i5-10400F, and the GPU is an Nvidia RTX 3060Ti which can accelerate the training process of the deep neural network. The GPU memory size is 12 G, and the computer memory is 32 G.

4.2. Results of Train and Simulation

In this paper, to train the PG-option hierarchical decision algorithm, the four meta-policies must firstly be trained to converge. According to Figure 7, it can be seen that at the beginning of training, our UAV adopts a random policy. After a period of training, the reward value convergence tends to be stable, and our UAV has been able to complete the corresponding tasks stably.

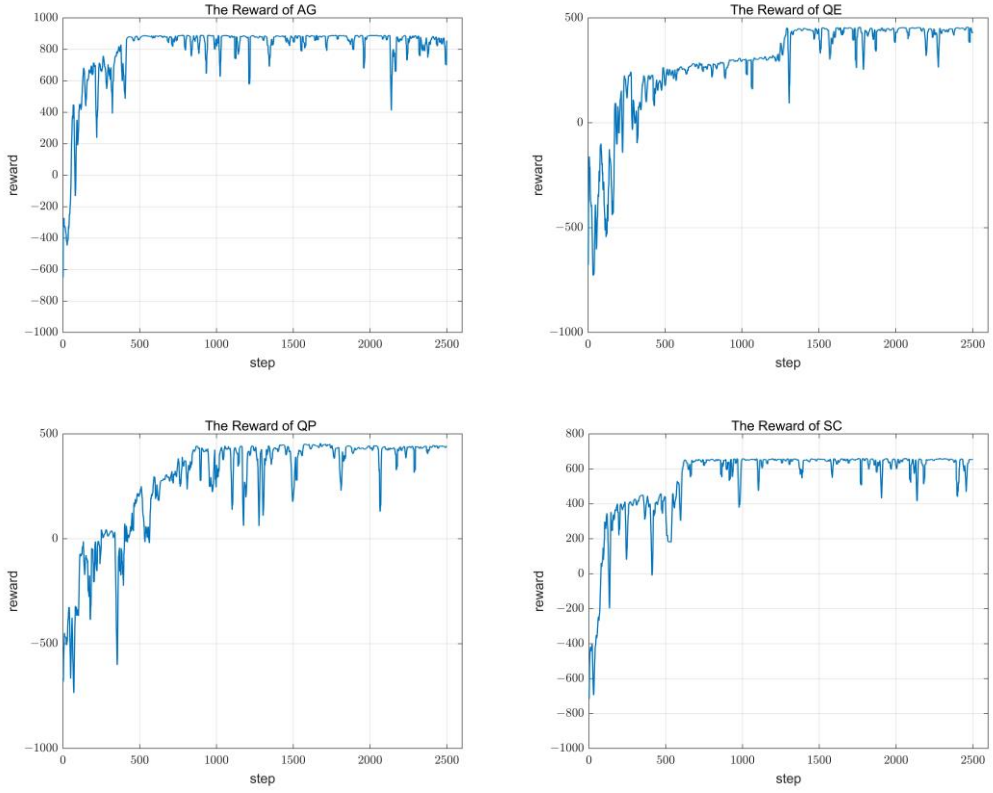


Figure 7. The reward function of meta-policies.

In this paper, to train the hierarchical decision algorithm based on the PG option, in the initial environment, the positions of both sides are randomly initialized within a certain range, and the opponent UAV is added to the disturbance in the original maneuvering mode.

Our UAV maneuvers use hierarchical decisions based on the PG-option algorithm. The opponent UAV maneuvers according to the predetermined trajectory and will threaten our UAV.

In this paper, the success rate is used to judge whether the policy selector network converges or not. The specific method is to count the success rate of the UAV in the training round. If the final success rate can converge within a certain range, it shows that the network of policy selectors has been trained and converged.

Training is performed in the initial environment using Table 4. The random value range is taken $[-1000, 1000]$, and the success rate is counted every 50 rounds.

Table 4. Initialization information in the training scene.

	X (km)	Y (km)	Z (km)	Pitch (°)	Yaw (°)	Speed (m/s)	Distance (km)	Azimuth (°)
Our UAV	2	3.5	−3	2	50	70	7.457	97.3
Opponent UAV	−3.5	3	2	1	−40	75		

It can be seen from Figure 8 that the final success rate is stable at about 82% under the above training, indicating that the policy selector training has converged after about 7200 rounds.

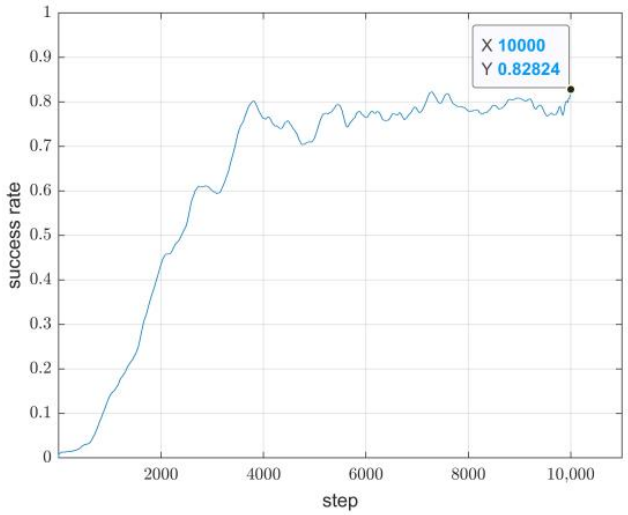


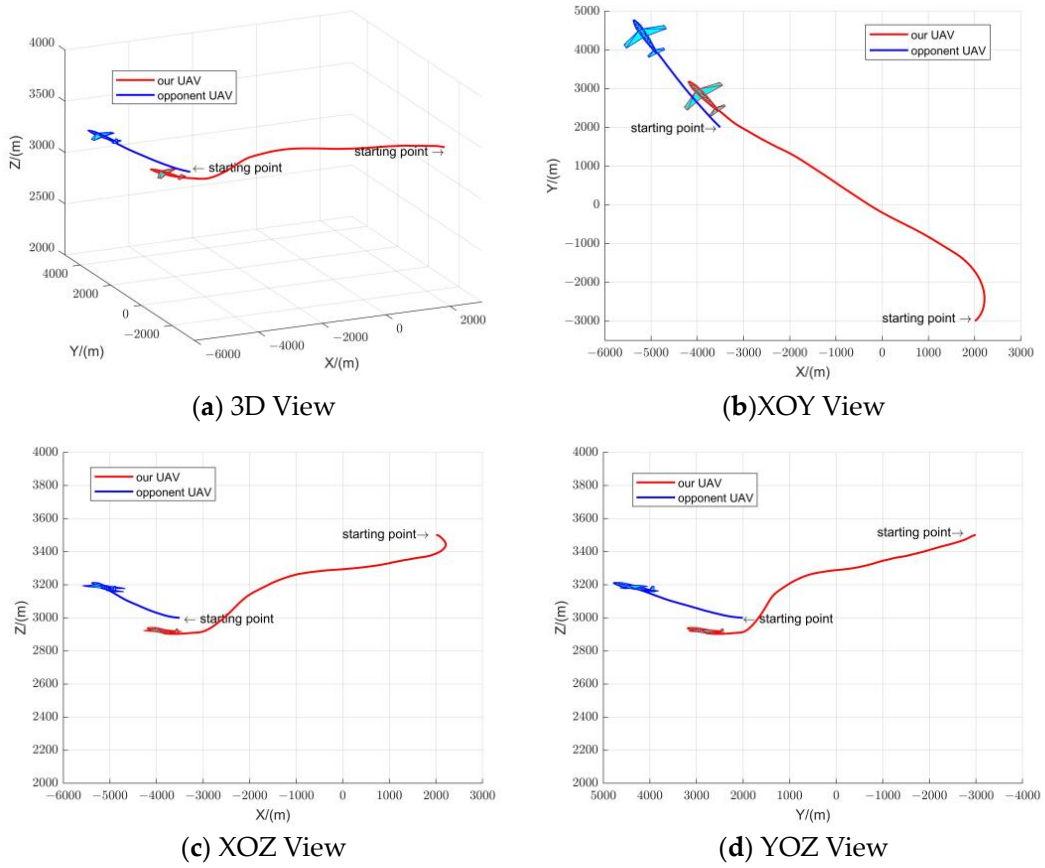
Figure 8. The curve of success rate in the training scene.

The trained model is tested in the training scene. Figure 9 shows the flight trajectory of the opponent and our UAV in the training scene.

At the initial time, the two UAVs are far away. It can be seen from the XOZ view that the height of the opponent UAV is lower than that of our UAV when the simulation begins. After that, our UAV changes the pitch angle, raises the height and shortens the distance between the two sides. From the XOY view, our UAV can reduce the opponent's azimuth angle against us, fly to the opponent's tail and fix the opponent UAV in the threat zone. Finally, the relative distance between the two sides is gradually shortened to meet the conditions of capturing.

From Figure 10a, it can be seen that before 80 steps, our UAV reduces the relative azimuth of the opponent UAV by rapidly rotating the heading angle and the pitch angle. After that, the relative azimuth of our UAV has always been kept in a small range. From

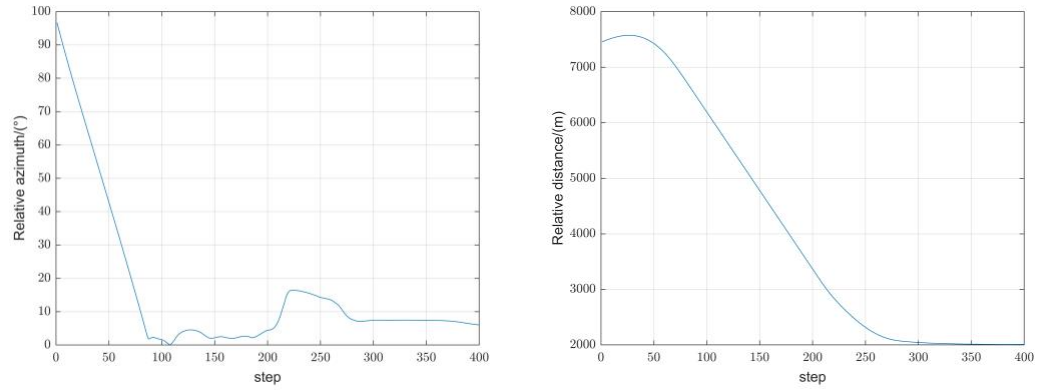
Figure 10b, it can be seen that our UAV can approach the target faster. After meeting the distance conditions of the threat zone, our UAV can converge stably at about 2000 m.



(**a**) 3D View (**b**)XOY View

(**c**) XOZ View (**d**) YOZ View

Figure 9. The flight trajectory in the training scene.



(**a**) Relative Azimuth (**b**) Relative Distance

Figure 10. Relative situation between opponent and our UAV in the training scene.

The selection of meta-policy is shown in Figure 11. At the beginning, the UAV chooses SC for situation conversion. After 220 steps, our UAV chooses the QP method to approach the opponent to capture it, and the selected meta-policy is more in line with the current situation.

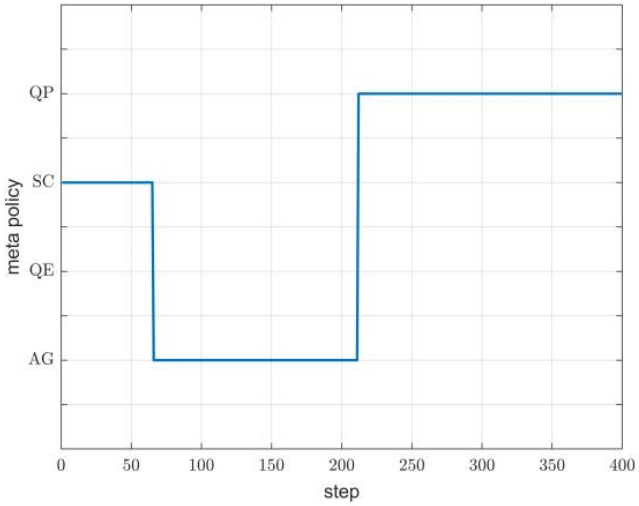


Figure 11. Meta-policy selection in the training scene.

4.3. Generalization Simulation

To test whether the PG-option hierarchical decision algorithm can realize the intelligent decision-making task under different initial situations, our UAV is placed in the pursued situation and the active capture situation, respectively, and its flight trajectory is analyzed.

4.3.1. Active Capture Situation

The active capture situation means that the relative azimuth angle of both sides is small at the beginning of the operation, and the overall situation is that the opponent UAV and our UAV are maneuvering in opposite directions. A simulation will be carried out in this situation. The initial state is shown in Table 5.

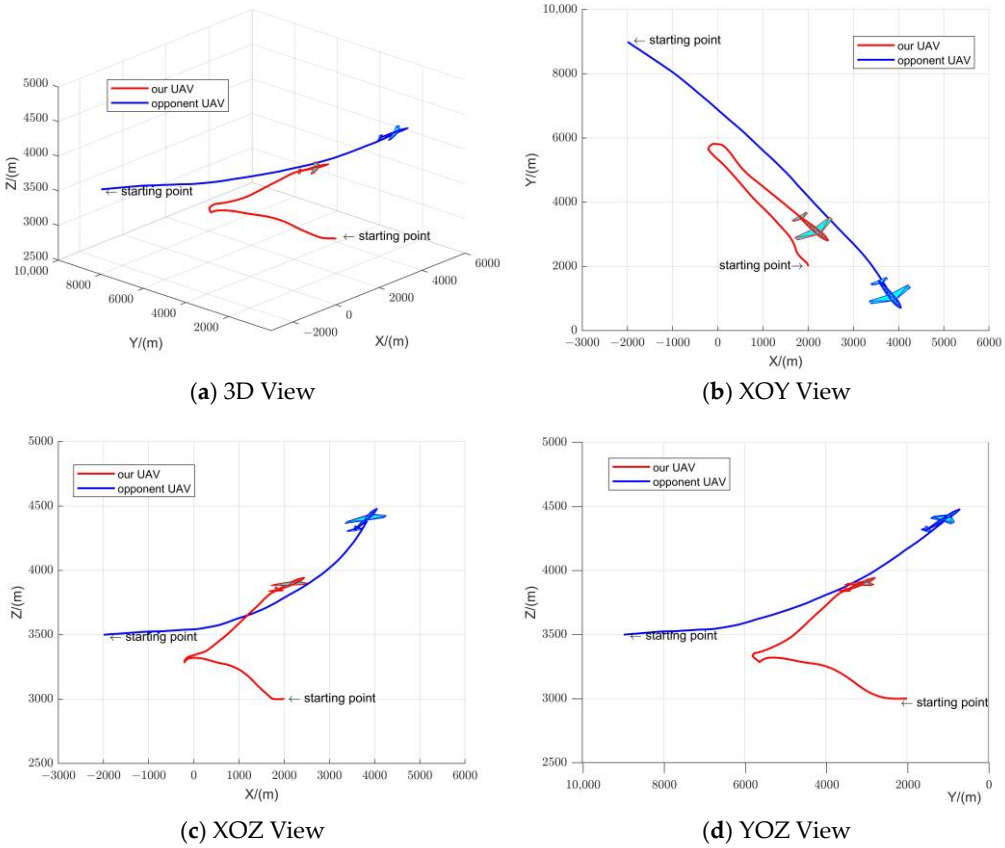
Table 5. Initialization information in active capture situation.

	X (km)	**Y (km)**	**Z (km)**	**Pitch (°)**	**Yaw (°)**	**Speed (m/s)**	**Distance (km)**	**Azimuth (°)**
Our UAV	2	2	3	2	50	70	7.991	29.6
Opponent UAV	−2	9	3.5	1	−40	80		

The flight trajectory is shown in Figure 12. From Figure 12a, it can be seen that at the beginning of the simulation, the height of the opponent UAV is higher than that of our UAV. Our UAV adjusts its pitch angle and shortens the distance between the opponent and our UAV. From Figure 12b, it can be seen that at the beginning of the simulation, both sides of the opponent UAV and our UAV are maneuvering toward each other. Our UAV continues to maneuver and threat around the tail of the opponent UAV.

It can be seen from Figure 13a that after the simulation, our UAV maneuvers towards the opponent and reduces the relative azimuth of the opponent. The relative azimuth meets the requirements of the threat zone from beginning to end. After our UAV escapes the opponent threat area, our UAV keeps the opponent within its threat area.

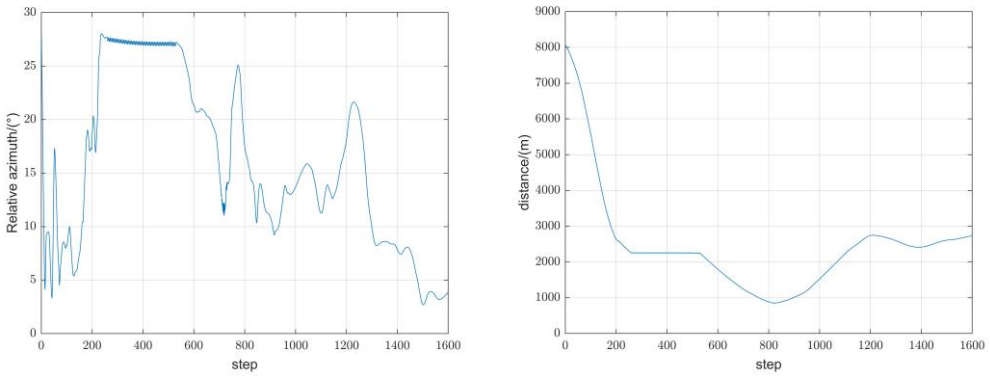
From Figure 13b, it can be seen that after the simulation begins, the two sides maneuver in opposite directions, and the distance between the opponent UAV and our UAV decreases rapidly. When the distance between the opponent UAV and our UAV meets the requirements of the threat zone, our UAV continues to maneuver and keeps the distance between the opponent and our UAV in this range.



(**a**) 3D View (**b**) XOY View

(**c**) XOZ View (**d**) YOZ View

Figure 12. The flight trajectory in active capture situation.



(**a**) Relative Azimuth (**b**) Relative Distance

Figure 13. Relative situation between opponent and our UAV in active capture situation.

The selection of meta-policy in active capture situation is shown in Figure 14. Under the situation, our UAV begins to choose QP. At about 200 steps, because our UAV falls into the opponent's threat zone in the process of pursuing the opponent, it chooses SC. At about 830 steps, our UAV chooses QP and finally wins.

4.3.2. Pursued Situation

The pursued situation means that at the beginning of the operation, our relative azimuth is small, and the opponent's relative azimuth is large. The overall situation is that the opponent UAV is at the tail of our UAV. A simulation will be performed in this situation. The initial state of the pursued situation is shown in Table 6.

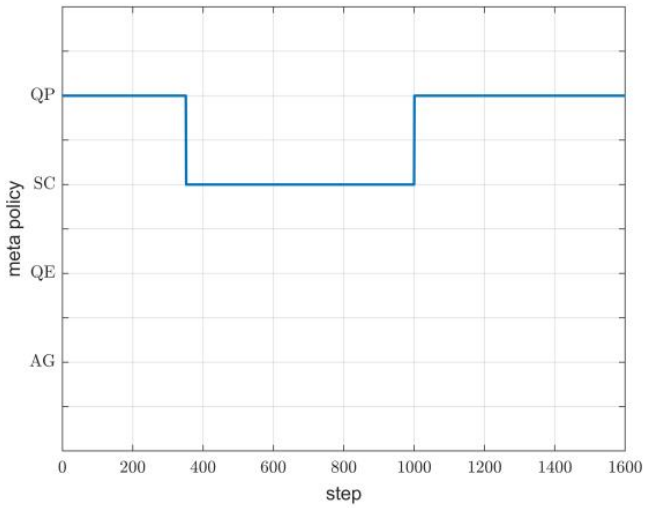


Figure 14. Meta-policy selection in active capture situation.

Table 6. Initialization information in pursued situation.

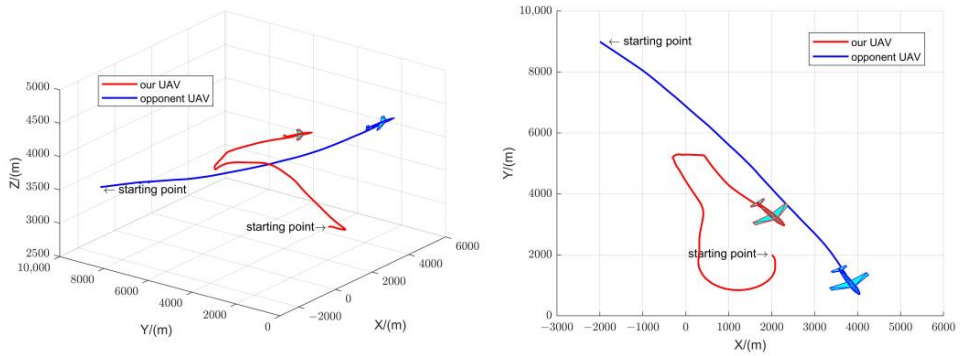
	X (km)	Y (km)	Z (km)	Pitch (°)	Yaw (°)	Speed (m/s)	Distance (km)	Azimuth (°)
Our UAV	2.1	2.9	3.0	2	50	70	8.01	165.1
Opponent UAV	−1.8	8.7	3.6	1	70	90		

The trajectory in pursued situation is shown in Figure 15. From Figure 15a, it can be seen that after the simulation begins, our UAV increases its pitch angle and reduces the distance between the opponent and us. When approaching the height of the opponent, our UAV gradually reduces its pitch angle to prevent our UAV from being higher than the height of the opponent. It can be seen from Figure 15b that after the simulation begins, our UAV quickly adjusts the heading angle to reduce the relative azimuth of the opponent. After adjusting the azimuth, our UAV maneuvers toward the opponent and shortens the distance between the two sides. To further reduce the relative azimuth of the opponent, our UAV begins to rotate the heading angle and maintain the tail-chasing situation of our UAV against the opponent.

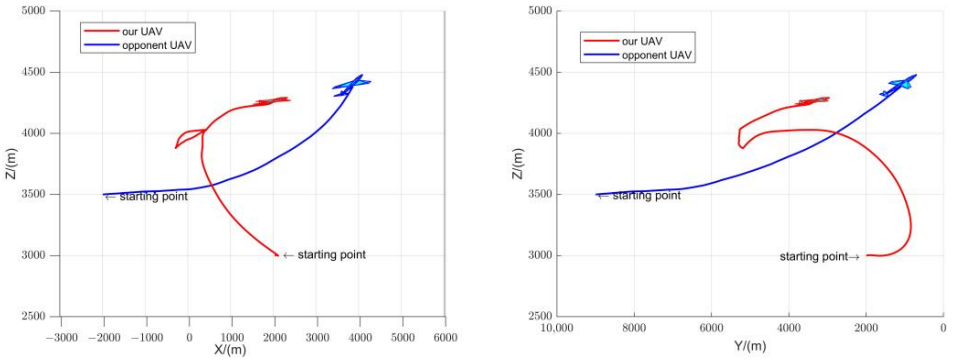
From Figure 16a, we can see that at the beginning of the simulation, the relative azimuth is very large, and our UAV is in a state of being pursued. After that, our UAV quickly reduces the relative azimuth and changes its unfavorable situation by maneuvering. After meeting the angle requirements of the threat zone, the simulation continues, and the azimuth has been meeting this requirement ever since.

The distance curve is shown in Figure 16b. It can be seen that at the beginning of the simulation because our UAV is pursued by the opponent, our initial speed is faster. Therefore, the distance between the opponent and our UAV will increase at the beginning. After that, our UAV quickly changes the heading angle and maneuvers toward the opponent. At the same time, the distance between the opponent and our UAV can always be maintained in this range after meeting the distance requirements of the threat zone.

The selection of meta-policy in the pursued situation is shown in Figure 17. Under the pursued situation, our UAV begins to choose the SC. At about 220 steps, our UAV chooses the AG to occupy a favorable position. At about 1100 steps, our UAV chooses the QP and successfully captures the opponent.

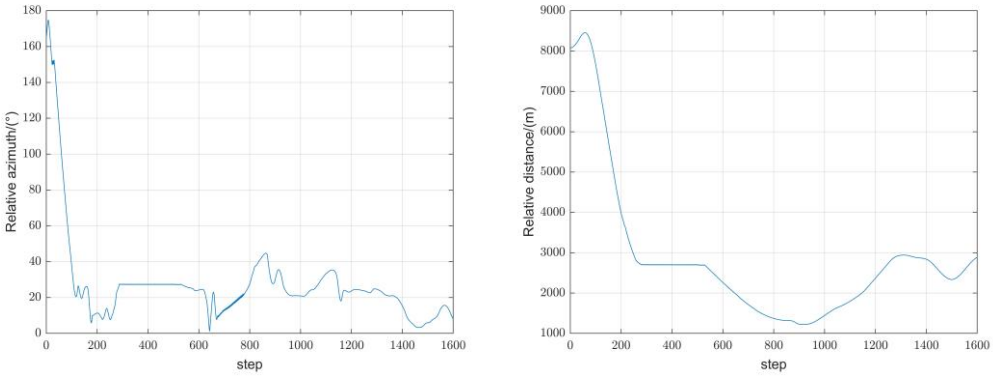


(**a**) 3D View (**b**) XOY View



(**c**) XOZ View (**d**) YOZ View

Figure 15. The flight trajectory in pursued situation.



(**a**) Relative Azimuth (**b**) Relative Distance

Figure 16. Relative situation between opponent and our UAV in pursued situation.

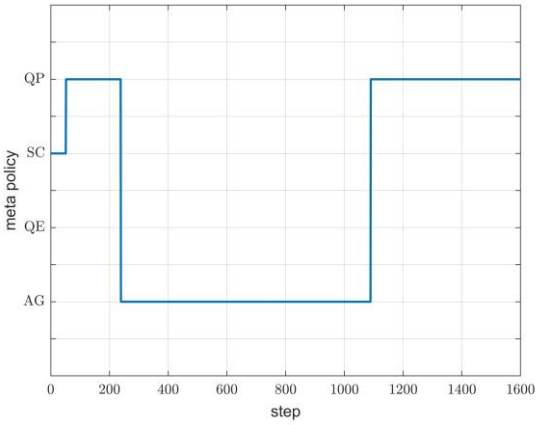


Figure 17. Meta-policy selection in pursued situation.

5. Conclusions

Based on the background of intelligent decision-making in the UAV pursuit-evasion game, this paper proposes a hierarchical maneuver decision-making method based on a PG option. The framework of maneuver decision-making is divided into two parts: the bottom is the four meta-policies of advantage game (AG), quick escape (QE), situation change (SC) and quick pursuit (QP), and the top is the policy selector trained by the PG algorithm. Delay evaluation and expert experience are introduced to effectively solve the problem of frequent meta-policy switching. The simulation and experimental results have shown that the method can choose reasonable meta-policies to control UAV maneuvering under different pursuit-evasion situations to capture the opponent. Some future work includes adding stationary or moving obstacles to simulation scenarios to make simulation more complex.

Author Contributions: Funding acquisition, B.L.; Methodology, H.Z.; Resources, B.L. and G.W.; Validation, H.Z., P.H. and K.Y.; Writing—original draft, B.L. and H.Z.; Writing—review and editing, G.W. and E.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by Project supported by the National Nature Science Foundation of China under grant no. 62003267, the Fundamental Research Funds for the Central Universities under grant no. G2022KY0602, the Technology on Electromagnetic Space Operations and Applications Laboratory under grant no. 2022ZX0090, the Key Research and Development Program of Shaanxi Province under grant no. 2023-GHZD-33, and the key core technology research plan of Xi'an under grant no. 21RGZN0016.

Data Availability Statement: Data sharing not applicable.

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

References

1. Chen, B. Research on AI Application in the Field of Quadcopter UAVs. In Proceedings of the 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT), Weihai, China, 14–16 October 2020; pp. 569–571.
2. Li, B.; Gan, Z.; Chen, D.; Sergey Aleksandrovich, D. UAV Maneuvering Target Tracking in Uncertain Environments Based on Deep Reinforcement Learning and Meta-Learning. *Remote Sens.* **2020**, *12*, 3789. [CrossRef]
3. Li, B.; Song, C.; Bai, S.; Huang, J.; Ma, R.; Wan, K.; Neretin, E. Multi-UAV Trajectory Planning during Cooperative Tracking Based on a Fusion Algorithm Integrating MPC and Standoff. *Drones* **2023**, *7*, 196. [CrossRef]
4. Liu, X.; Su, Y.; Wu, Y.; Guo, Y. Multi-Conflict-Based Optimal Algorithm for Multi-UAV Cooperative Path Planning. *Drones* **2023**, *7*, 217. [CrossRef]
5. Li, S.; Wu, Q.; Du, B.; Wang, Y.; Chen, M. Autonomous Maneuver Decision-Making of UCAV with Incomplete Information in Human-Computer Gaming. *Drones* **2023**, *7*, 157. [CrossRef]
6. Zhang, H.; He, P.; Zhang, M.; Chen, D.; Neretin, E.; Li, B. UAV Target Tracking Method Based on Deep Reinforcement Learning. In Proceedings of the 2022 International Conference on Cyber-Physical Social Intelligence (ICCSI), Nanjing, China, 18–21 November 2022; pp. 274–277.
7. Alanezi, M.A.; Haruna, Z.; Sha'aban, Y.A.; Bouchekara, H.R.E.H.; Nahas, M.; Shahriar, M.S. Obstacle Avoidance-Based Autonomous Navigation of a Quadrotor System. *Drones* **2022**, *6*, 288. [CrossRef]
8. Shahid, S.; Zhen, Z.; Javaid, U.; Wen, L. Offense-Defense Distributed Decision Making for Swarm vs. Swarm Confrontation While Attacking the Aircraft Carriers. *Drones* **2022**, *6*, 271. [CrossRef]
9. Awheda, M.D.; Schwartz, H.M. A fuzzy reinforcement learning algorithm using a predictor for pursuit-evasion games. In Proceedings of the 2016 Annual IEEE Systems Conference (SysCon), Orlando, FL, USA, 18–21 April 2016; pp. 1–8.
10. Gao, K.; Han, F.; Dong, P.; Xiong, N.; Du, R. Connected Vehicle as a Mobile Sensor for Real Time Queue Length at Signalized Intersections. *Sensors* **2019**, *19*, 2059. [CrossRef] [PubMed]
11. Alexopoulos, A.; Kirsch, B.; Badreddin, E. Realization of pursuit-evasion games with unmanned aerial vehicles. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13–16 June 2017; pp. 797–805.
12. Gan, Z.; Li, B.; Neretin, E.; Sergey Aleksandrovich, D. UAV Maneuvering Target Tracking based on Deep Reinforcement Learning. *J. Phys.* **2021**, *1958*, 12015. [CrossRef]
13. Yu, F.; Zhang, X.; Li, Q. Determination of The Barrier in The Qualitatively Pursuit-evasion Differential Game. In Proceedings of the 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC), Xiamen, China, 10–12 August 2018; pp. 1–6.
14. Pan, Q.; Zhou, D.; Huang, J.; Lv, X.; Yang, Z.; Zhang, K.; Li, X. Maneuver decision for cooperative close-range air combat based on state predicted influence diagram. In Proceedings of the 2017 IEEE International Conference on Information and Automation (ICIA), Macao, China, 18–20 July 2017; pp. 726–731.

15. Mikhail, K.; Vyacheslav, K. Notes on the pursuit-evasion games between unmanned aerial vehicles operating in uncertain environments. In Proceedings of the 2021 International Conference Engineering and Telecommunication (En&T), Dolgoprudny, Russia, 24–25 November 2021; pp. 1–5.
16. Han, Z. The Application of Artificial Intelligence in Computer Network Technology. In Proceedings of the 2021 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), Shanghai, China, 15–17 October 2021; pp. 632–635.
17. Zhu, X.; Wang, Z.; Li, C.; Sun, X. Research on Artificial Intelligence Network Based on Deep Learning. In Proceedings of the 2021 2nd International Conference on Information Science and Education (ICISE-IE), Chongqing, China, 26–28 November 2021; pp. 613–617.
18. Lyu, L.; Shen, Y.; Zhang, S. The Advance of Reinforcement Learning and Deep Reinforcement Learning. In Proceedings of the 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 25–27 February 2022; pp. 644–648.
19. Li, W.; Wu, J.; Chen, J.; Lia, K.; Cai, X.; Wang, C.; Guo, Y.; Jia, S.; Chen, W.; Luo, F.; et al. UAV countermeasure maneuver decision based on deep reinforcement learning. In Proceedings of the 2022 37th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Beijing, China, 19–20 November 2022; pp. 92–96.
20. Zhang, Y.Z.; Xu, J.L.; Yao, K.J.; Liu, J.L. Pursuit missions for UAV swarms based on DDPG algorithm. *Acta Aeronaut. Astronaut. Sin.* **2020**, *41*, 314–326.
21. Zhang, R.; Zong, Q.; Zhang, X.; Dou, L.; Tian, B. Game of Drones: Multi-UAV Pursuit-Evasion Game With Online Motion Planning by Deep Reinforcement Learning. *IEEE Trans. Ind. Neural Netw. Learn. Syst.* **2022**. [CrossRef] [PubMed]
22. Fu, X.; Zhu, J.; Wei, Z.; Wang, H.; Li, S. A uav pursuit-evasion strategy based on ddpg and imitation learning. *Int. J. Aerosp. Eng.* **2022**, *2022*, 1–14. [CrossRef]
23. Sun, Y.; Yan, C.; Lan, Z.; Lin, B.; Zhou, H.; Xiang, X. A Scalable Deep Reinforcement Learning Algorithm for Partially Observable Pursuit-Evasion Game. In Proceedings of the 2022 International Conference on Machine Learning, Cloud Computing and Intelligent Mining (MLCCIM), Xiamen, China, 5–7 August 2022; pp. 370–376.
24. Vlahov, B.; Squires, E.; Strickland, L.; Pippin, C. On Developing a UAV Pursuit-Evasion Policy Using Reinforcement Learning. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications(ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 859–864.
25. Li, Z. A Hierarchical Autonomous Driving Framework Combining Reinforcement Learning and Imitation Learning. In Proceedings of the 2021 International Conference on Computer Engineering and Application (ICCEA), Kunming, China, 25–27 June 2021; pp. 395–400.
26. Cheng, Y.; Wei, C.; Sun, S.; You, B.; Zhao, Y. An LEO Constellation Early Warning System Decision-Making Method Based on Hierarchical Reinforcement Learning. *Sensors* **2023**, *23*, 2225. [CrossRef] [PubMed]
27. Qiu, Z.; Wei, W.; Liu, X. Adaptive Gait Generation for Hexapod Robots Based on Reinforcement Learning and Hierarchical Framework. *Actuators* **2023**, *12*, 75. [CrossRef]
28. Li, Q.; Jiang, W.; Liu, C.; He, J. The Constructing Method of Hierarchical Decision-Making Model in Air Combat. In Proceedings of the 2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 22–23 August 2020; pp. 122–125.
29. Bacon, L.; Harb, J.; Precup, D. The option-critic architecture. In Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 1726–1734.
30. Wu, Y.; Sun, G.; Xia, X.; Xing, M.; Bao, Z. An Improved SAC Algorithm Based on the Range-Keystone Transform for Doppler Rate Estimation. *IEEE Geosci. Remote Sens. Lett.* **2013**, *10*, 741–745.
31. Gao, M.; Chang, D. Autonomous Driving Based on Modified SAC Algorithm through Imitation Learning Pretraining. In Proceedings of the 2021 21st International Conference on Control, Automation and Systems (ICCAS), Jeju, Republic of Korea, 12–15 October 2021; pp. 1360–1364.
32. Xiao, T.; Qi, Y.; Shen, T.; Feng, Y.; Huang, L. Intelligent Task Offloading Method for Vehicular Edge Computing Based on Improved-SAC. In Proceedings of the 2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 16–18 December 2022; pp. 1720–1725.
33. Zhu, Q.; Su, S.; Tang, T.; Xiao, X. Energy-efficient train control method based on soft actor-critic algorithm. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; pp. 2423–2428.
34. Ota, K.; Jha, D.K.; Kanezaki, A. Training larger networks for deep reinforcement learning. *arXiv* **2021**, arXiv:2102.07920v1.