processes

MDPI

Article

Research on Unmanned Aerial Vehicle Intelligent Maneuvering Method Based on Hierarchical Proximal Policy Optimization

Yao Wang [1], Yi Jiang [1], Huiqi Xu [1,*], Chuanliang Xiao [2] and Ke Zhao [2]

[1] Naval Aviation University, Yantai 264001, China; wy@cqu-edu.cn (Y.W.); jiangyi1989913@163.com (Y.J.)
[2] Electrical and Electronic Engineering College, Shandong University of Technology, Zibo 255000, China; xiaocl@sdut.edu.cn (C.X.); 23504040599@stumail.sdut.edu.cn (K.Z.)
* Correspondence: jy@cqu-edu.cn; Tel.: +86-15564506237

Abstract: Improving decision-making in the autonomous maneuvering of unmanned aerial vehicles (UAVs) is of great significance to improving flight safety, the mission execution rate, and environmental adaptability. The method of deep reinforcement learning makes the autonomous maneuvering decision of UAVs possible. However, the current algorithm is prone to low training efficiency and poor performance when dealing with complex continuous maneuvering problems. In order to further improve the autonomous maneuvering level of UAVs and explore safe and efficient maneuvering methods in complex environments, a maneuvering decision-making method based on hierarchical reinforcement learning and Proximal Policy Optimization (PPO) is proposed in this paper. By introducing the idea of hierarchical reinforcement learning into the PPO algorithm, the complex problem of UAV maneuvering and obstacle avoidance is separated into high-level macro-maneuver guidance and low-level micro-action execution, greatly simplifying the task of addressing complex maneuvering decisions using a single-layer PPO. In addition, by designing static/dynamic threat zones and varying their quantity, size, and location, the complexity of the environment is enhanced, thereby improving the algorithm's adaptability and robustness to different conditions. The experimental results indicate that when the number of threat targets is five, the success rate of the H-PPO algorithm for maneuvering to the designated target point is 80%, which is significantly higher than the 58% rate achieved by the original PPO algorithm. Additionally, both the average maneuvering distance and time are lower than those of the PPO, and the network computation time is only 1.64 s, which is shorter than the 2.46 s computation time of the PPO. Additionally, as the complexity of the environment increases, the H-PPO algorithm outperforms other compared networks, demonstrating the effectiveness of the algorithm constructed in this paper for guiding intelligent agents to autonomously maneuver and avoid obstacles in complex and time-varying environments. This provides a feasible technical approach and theoretical support for realizing autonomous maneuvering decisions in UAVs.

Keywords: autonomous maneuver; hierarchical reinforcement learning; UAV; proximal policy optimization; threat avoidance

check for updates

Academic Editors: Weicun Zhang and Quanmin Zhu

Received: 18 December 2024
Revised: 7 January 2025
Accepted: 9 January 2025
Published: 27 January 2025

Citation: Wang, Y.; Jiang, Y.; Xu, H.; Xiao, C.; Zhao, K. Research on Unmanned Aerial Vehicle Intelligent Maneuvering Method Based on Hierarchical Proximal Policy Optimization. *Processes* **2025**, *13*, 357. https://doi.org/10.3390/pr13020357

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the progress of technology, UAVs have been widely used in agriculture, rescue efforts, military endeavors, and other fields to undertake a variety of challenging tasks. Improving the efficient and safe maneuvering performance of UAVs is the basis and prerequisite for completing the established tasks to a higher standard. Improving the autonomous maneuvering level of UAVs is of great significance to enhance the safety

and efficiency of maneuvering, reduce human participation in the loop, and increase the efficiency and cost ratio. Traditional maneuvering decision-making methods include game theory [1], intuitive fuzzy [2,3], dynamic Bayesian network [4], genetic algorithm [5], approximate dynamic planning [6], influence graph [7], rough set theory [8], rolling time domain [9,10], and so on. Game theory [11], intuitive ambiguity [12], influence maps [13], and other decision-making methods can be used to build complete and clear models, but the operation procedure is often complicated and many complex problems are difficult to model; the methods of approximate dynamic planning and rough set theory have a large operation volume and low efficiency in solving problems online, and rough set theory requires huge information support; and dynamic Bayesian networks require a full understanding of tasks and have poor adaptability to unknown scenarios. In addition, the traditional maneuvering decision-making method needs to be combined with the attitude/position controller to complete the corresponding maneuver, which reduces the mobility autonomy of the equipment platform and makes it difficult to meet the needs of autonomous, efficient, and intelligent mobility for the UAV in a time-varying environment.

With the development of artificial intelligence technology, deep reinforcement learning algorithms make it possible for UAVs to maneuver autonomously. In 2013, a deep reinforcement learning method was proposed by the DeepMind team [14], which combines the RL method with decision-making ability, DL (deep learning), and environmental perception ability, providing a solution for handling the perceptual decision problem of complex systems [15]. At present, the method of deep reinforcement learning has been widely used in automatic control, information theory, and other fields. Some of the deep algorithms have general intelligence to deal with complex tasks and reach a similar level to human beings in dealing with practical problems.

The method based on deep reinforcement learning is studied to explore the intelligent autonomous maneuvering of UAVs, which is in line with the current intelligent development trend and the practical application needs of improving the maneuver efficiency of UAVs. Scholars from all walks of life have achieved fruitful results through their continuous exploration. In references [16–18], a maneuver intelligent autonomous decision method was constructed through RL algorithms such as DDPG (Deep Deterministic Policy Gradient). Reference [19] used the reinforcement learning method to accumulate decision-making experience and realized the intelligent action decision-making task in a virtual simulation experiment. Todd Hester's team [20], using the DQN (Deep Q-network) algorithm in the learning process of poor training, proposed the DQfD (Deep Q-learning from Demonstrations) algorithm, building upon the idea of supervised learning and accelerating the process of reinforcement learning through a small amount of prior data and pre-training methods. The DeepMind team [21] fused a variety of improved DQN algorithms and proposed the Rainbow algorithm, which performed better than other methods in dealing with decision-making problems in complex environments.

However, there are the following difficulties in using the current deep reinforcement learning method to improve the autonomous intelligent maneuvering obstacle avoidance of UAVs:

(1) The motion space of UAV maneuvering decisions is continuous, and the combination of maneuver strategies is diverse and complex; furthermore, the maneuver airspace is huge, and the detection performance of all directions is uneven. Problems may be encountered when using the deep reinforcement learning method directly, such as the large training space and difficulty in setting the distribution of targets; in addition, the training process does not converge easily.

(2) The ultimate goal of the maneuvering obstacle avoidance mission is to reach the designated location point. In terms of the maneuver decision, the UAV should not

only consider entering the established area successfully, but also whether the UAV has reached the target point and determine the outcome of the mission.

(3) In the process of maneuver, considering the influence of various unpredictable dynamic threat target factors, the threat area is time-changing and of different sizes, and the decision-making process is extremely uncertain, which further increases the difficulty of UAV maneuvering and obstacle avoidance.

In order to improve the autonomous maneuvering decision-making ability of UAVs in complex environments and improve the efficiency and safety of maneuvering, this paper introduces the idea of hierarchical reinforcement learning on the basis of PPO (Proximal Policy Optimization). The main focus of this work is as follows:

In view of the situation that the UAV action and state space are large and training is difficult to converge, the concept of HRL (hierarchical reinforcement learning) is adopted on the basis of PPO, and the UAV maneuvering obstacle avoidance task is divided into the low-level control task and high-level guidance task. The low-level control strategy is responsible for generating the action command of the UAV, controlling the angle, speed, and other actions of the UAV, and the high-level guidance strategy sets multiple intermediate sub-goal points between the starting and ending points and guides the UAV to maneuver along the specified path. The method of solving problems in a hierarchical manner can reduce the complexity of maneuvering and obstacle avoidance tasks, reduce decision space, and improve training efficiency and strategy quality.

For a UAV to meet the realistic requirements of specific tasks, the optimal path reward, the final task reward, maneuver guide reward, avoid threat reward, and four-reward function must be designed, and the final reward estimate must be used to replace the proximal strategy optimization of the original advantage estimation function to improve the network's training performance and efficiency.

In view of the situation with a strong unknown environment and variable threat information, random dynamic threat targets are set in the process of training agent maneuver, representing multiple types of threat targets with different threat areas so as to further verify the ability of the network to adapt to complex environments.

This paper is organized as follows:

Section 1 is the Introduction, which describes the development of maneuver decision methods, focusing on analyzing the important and difficult problems existing in the current deep reinforcement learning method in the realization of autonomous maneuvering of UAVs and the solutions adopted in this paper. Section 2 introduces the UAV maneuvering obstacle avoidance model, and a UAV motion model and path planning constraint model are constructed for the UAV maneuvering problem. This section also sets static and dynamic threat targets. Section 3 presents the analysis and introduction of the deep reinforcement learning algorithm, which elaborates on the principle of the PPO algorithm and hierarchical reinforcement learning algorithm and expounds the principle, composition, process, and training method of the H-PPO algorithm proposed in this paper. Section 4 is the experimental part, which verifies the effectiveness of the H-PPO algorithm to achieve the autonomous maneuvering obstacle avoidance task of UAVs in complex environments.

2. UAV Maneuvering Obstacle Avoidance Model

2.1. UAV Maneuvering Motion Model

The problem in UAV maneuvering path planning is exploring a maneuver path that can meet the task requirements, constraint conditions, and optimization indexes based on the coordinates of the position of the UAV and ship target, the relative movement situation,

the sudden threat, and the limitation condition of the UAV's mobility energy. Considering the UAV as a particle, its motion model is expressed as

$$\begin{cases} \frac{dx}{dt} = v\cos(\beta) \\ \frac{dy}{dt} = v\sin(\beta) \\ u = \frac{d\beta}{dt} = \frac{A}{v} \\ \frac{dv}{dt} = a \end{cases} \tag{1}$$

In Formula (1), (x, y) represents the position of the UAV, t represents the time, β represents the deviation angle of the maneuver path, u represents the control amount, A represents the normal acceleration, v represents the maneuver speed, and a represents the maneuver acceleration.

The motion model is shown in Figure 1. The dotted line represents the straight-line distance between the UAV's starting position and the target position, and the red arrow indicates the UAV's current direction of movement.

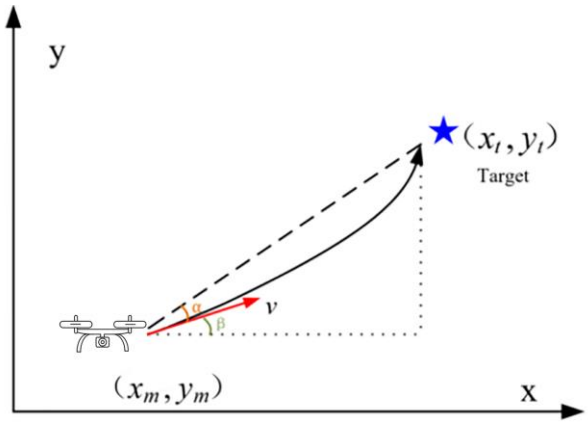


Figure 1. The UAV motion model.

(x_m, y_m) is set as the position coordinate of the UAV in space, (x_t, y_t) is set as the target coordinate, and the comparison expression is as follows:

$$\begin{cases} (\Delta x, \Delta y) = (x_m - x_t, y_m - y_t) \\ d = \sqrt{\Delta x^2 + \Delta y^2} \\ v' = v\cos\alpha \\ \alpha = \arcsin\frac{\Delta y}{d} - \beta \end{cases} \tag{2}$$

The equation of continuous motion is

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\alpha}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos\beta(t) \\ v(t)\sin\beta(t) \\ \omega(t) \\ a(t) \end{bmatrix} \tag{3}$$

$(\Delta x, \Delta y)$ is the difference between the coordinates of the UAV and the designated target position, d represents the linear distance between the UAV and the target, v' represents the velocity component of the position and the designated linear direction at a certain point in the UAV maneuver, and α represents the angle between the maneuver direction and the linear direction of the target.

The state of the agent at the interval t can be expressed as

$$\begin{cases} x(t) = x(t-1) + v(t-1)\Delta t \cos\beta(t-1) \\ y(t) = y(t-1) + v(t-1)\Delta t \sin\beta(t-1) \\ v(t) = [v(t-1) + a(t-1)\Delta t]_{V_{\min}}^{V_{\max}} \\ \beta(t) = \beta(t-1) + \omega(t-1)\Delta t \end{cases} \tag{4}$$

2.2. Path Planning Constraint Model

To explore the optimal path, various constraints should be considered, such as the maneuver time, start and end points, flight speed limit, threat size, threat area, no-fly zone, and energy consumption [22]. It is a decision control task that needs to consider the process and terminal constraints, and the constraint model of UAV path planning can be expressed as

$$\begin{aligned} &\min X = f(L) \\ &s.t. \begin{cases} c_i(L) = 0, (i = 1, \ldots, k') \\ \bar{c}_i(L) \geq 0, (i = k'+1, \ldots, k) \end{cases} \end{aligned} \tag{5}$$

$L = L(t)$ represents the maneuver route; it is a continuous curve, $c_i = (L)$, where $\bar{c}_i = (L)$ represents the constraint.

In the maneuver process, the UAV consumes less energy, but when the angle change occurs, the engine needs to provide more energy to balance the normal acceleration, and the angle of attack increases the induced resistance of the UAV [23,24].

Therefore, when the UAV moves to avoid threats, it will increase energy consumption and shorten the maneuver range. The constraint indicator functions are established as follows:

$$\begin{aligned} &F = \eta_1 f_1 + \eta_2 f_2 + \eta_3 f_3 | \textstyle\sum_i \eta_i = 1 \\ &\begin{cases} f_1 = \sum_{i=1}^{m} l_i, l_i = \|(x_m, y_m) \rightarrow (x_n, y_n)\|_2 \\ f_2 = T \\ f_3 = \int_0^T n^2(t)dt \approx \sum_{i=1}^{m} n_i^2 \end{cases} \end{aligned} \tag{6}$$

f_1–f_3 represents the voyage, time, and energy consumption, and it describes energy consumption through overload. η is the conditional coefficient; m is the number of route segments; l_i is the range of segment i, represented by two norms; n_i is the overload of segment i; and T is the total maneuver time.

Among them are the following:

(1) In terms of voyage constraints, the longest distance of the UAV maneuver needs to be less than its maximum maneuver range, that is, $f_1 \leq L_{\max}$, where $L_{\max}$ is the maximum maneuver range.

(2) For the speed constraint, the UAV maneuvering speed should be between the minimum cruising speed $V_{\min}$ and the maximum maneuver speed $V_{\max}$, i.e., $V_{\min} \leq v \leq V_{\max}$.

(3) For the maneuver overload limit, $n(t) \leq n_{\max}$, where $n_{\max}$ indicates the maximum overload, and $n = v\beta$.

(4) For the threat area, no-fly zone, and terrain obstacle mobility constraints, in the process of UAV maneuver, the no-fly zone and terrain obstacle are static, and the aircraft, drones, birds, and other threat areas are dynamic; the no-fly zone and terrain obstacle constraints are set as B_{jt}, the dynamic threat area constraints are set as B_{dt}, and the UAV in the process of any position is represented by $M(x, y)$ and cannot be in the two areas. This is shown by the following formula:

$$M(x, y) \big| (x_t, y_t) \notin \left(B_{jt} \cup B_{dt}\right), t \in (0, T) \tag{7}$$

2.3. Threat Zone Setting

2.3.1. Static Threat Area Setting

At a certain point, the position coordinate of the UAV in space is (x_m, y_m), the static threat area is represented by the circle, the central coordinate is $(x_j, y_j)(j = 1, 2, \ldots, n)$, and the radius of the threat area is R_J. A static threat area can be represented as

$$J = \left\{ (x, y) \in R_2 \middle| (x - x_j)^2 + (y - y_j)^2 \leq R_J \right\} \tag{8}$$

where J is the static circular region, (x, y) represents any coordinate point of the two-dimensional plane, and R_2 is the two-dimensional real number space, that is, the plane containing all possible coordinate points.

The distance between the UAV and the threat center point is

$$d_{dj} = \sqrt{(x_j - x_m)^2 + (y_j - y_m)^2} \tag{9}$$

The UAV needs a certain reaction time to detect the threat area and take maneuvering measures. Within this time, the UAV will continue to maneuver, so a margin value d_{safe} needs to be set to ensure that the UAV is kept at a sufficient distance from the threat area to take effective evasive measures. The threat avoidance condition is $d_{dj} \geq R_J + d_{safe}$.

2.3.2. Dynamic Threat Region Setting

The dynamic threat movement should be set at a constant speed; the speed is v_{D_i}, the initial coordinate of the central point of the threat area is $D_i(x_0, y_0)(i = 1, \ldots, n)$, the angle between the movement direction of the threat area and the maneuver direction of the UAV to the target point is γ, the central point coordinate of the threat area at the time t is $D_i(x_t, y_t)$, and the expression is

$$\begin{cases} x_t = x_0 + v_{D_i} \cos \gamma \\ y_t = y_0 + v_{D_i} \sin \gamma \end{cases} \tag{10}$$

The expression for a dynamic circular threat area is

$$(x_t - x_0)^2 + (y_t - y_0)^2 = {R_D}^2 \tag{11}$$

In order to achieve effective avoidance, the following needs to be met:

$$d_{dd} \geq R_D + d_{safe}, d_{dd} = \sqrt{(x_d - x_m)^2 + (y_d - y_m)^2} \tag{12}$$

R_D is the dynamic threat area radius.

3. Deep Reinforcement Learning Algorithm

A deep reinforcement learning algorithm combines deep learning technology and reinforcement learning technology and can use the neural network representation strategy and state value function to realize autonomous learning and optimization. Deep reinforcement learning is based on the MDP (Markov decision process) to solve the optimal strategy.

The MDP process contains five elements $\langle S, A, R, T, \gamma \rangle$, where S is the set of state information obtained by the agent in each time step and environment interaction during training. The agent action set is represented by A, the reward R is obtained by performing the action, T is the state transfer function and represents the probability of transferring the state from s to s' after executing the action, and γ is the discount factor to balance long-term reward and immediate reward. The basic process of deep reinforcement learning is that the agent selects and executes the action according to the current environment, the environment gives the feedback reward and the next state according to the executed action,

and the agent updates the strategy according to the feedback of the environment to obtain a higher reward. By repeating the above process, the agent learns how to maximize the cumulative reward in the environment.

The cumulative reward R for the agent is

$$Rt = \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (13)$$

where r_t is the reward value the agent obtains at time t. The aim of deep reinforcement learning is to obtain the highest cumulative return and the best strategy in learning.

The state action value function represents the possible reward when the action a is executed in the state s using the strategy π, and the mathematical expression is

$$q_\pi(s,a) = E_\pi\left[\Sigma_{k=0}^{\infty}\gamma^k r_{t+k+1} | s_t = s, a_t = a\right] \quad (14)$$

The maximum state action value function is

$$q^*(s,a) = \max q_\pi(s,a) \quad (15)$$

The optimal policy π^* is the behavior chosen from $q^*(s,a)$ at the state s.

$$\pi^* = \max_{a \in A} q^*(s,a) \quad (16)$$

The deep reinforcement learning method has the characteristics of end-to-end and has outstanding advantages in planning the path of autonomous maneuvering of UAVs. The process of UAV maneuvering and obstacle avoidance requires continuous action and decision-making, and the state changes are also continuous. Common deep reinforcement learning algorithms for processing continuous action and state space include the DDPG (Deep Deterministic Policy Gradient) algorithm [25], TD3 (Twin Delayed DDPG) algorithm [26], SAC (Soft Actor–Critic) algorithm [27], PPO (Proximal Policy Optimization) algorithm [28], etc.

The DDPG algorithm is based on the Policy Gradient algorithm. It is difficult to choose the appropriate learning rate when processing a continuous action space, and a large amount of data is required for effective training. When dealing with complex and high-dimensional continuous action space, a deeper neural network is needed to adapt to the complexity of the problem. The TD3 algorithm uses the Relu activation function when dealing with the continuous action space problem, which is prone to gradient vanishing or explosion. SAC has high computational complexity, high resource consumption, and poor adaptability to environmental changes, and it is difficult to deploy in resource-constrained environments. The PPO algorithm combines the advantages of the DQN algorithm, Actor–Critic algorithm, and strategy gradient algorithm, meaning it has the characteristics of high efficiency, stability, and flexibility. The PPO algorithm supports online strategy and offline strategy and can be widely used in a variety of task scenarios, especially in handling the problem of continuous motion space.

3.1. PPO

The PPO algorithm improves on the basis of the TRPO (Trust Region Policy Optimization) algorithm, introduces Importance Sample and Clipping technology, reduces the complexity of the algorithm, and improves the stability and efficiency of training. The PPO algorithm implementation steps are as follows:

(1) Define the network parameters and initialize the policy network and the value network;
(2) Collect data through interaction with current policies and the environment;
(3) Use GAE (Generalized Advantage Estimation) to calculate the advantage function;
(4) Update the policy network according to Clipping technology and control the update range;

(5) Evaluate the performance of the updated policy network;
(6) Repeat the above steps until the desired effect is achieved.

The PPO algorithm is a deep reinforcement learning algorithm based on Policy Gradient, and the network uses the Actor–Critic structure [29]. In training, the PPO algorithm uses the rise of gradient to maximize the objective function $L_t^{CLIP+VF+S}(\theta)$. In this way, the network parameters θ can be updated to achieve optimal policy learning. The training process of the PPO algorithm is shown in Figure 2.

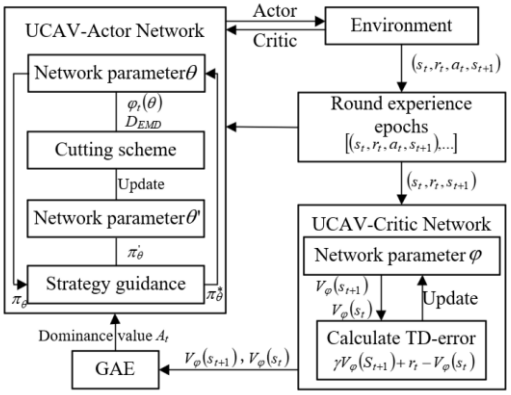


Figure 2. Training process of PPO.

The importance sampling formula is

$$E_{x\sim p}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim p}\left[f(x)\frac{p(x)}{q(x)}\right] \tag{17}$$

$\frac{p(x)}{q(x)}$ is the importance weight for correcting the difference between p and q.
The algorithm gradient in the offline strategy is expressed as

$$\nabla\overline{R}(\tau) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t\nabla\log p_\theta(a_t^n|s_t^n))\right] \tag{18}$$

The objective function is expressed as

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t)\right] \tag{19}$$

The PPO algorithm can be subdivided into PPO-penalty and PPO-Clip according to different update methods. PPO-penalty optimizes the objective function based on the KL divergence penalty term, and it regards the non-negative constraint as a reward and punishment mechanism. When the agent performs an action that does not meet the constraint conditions, the policy is punished by introducing a penalty term into the loss function. According to the non-negativity of the behavior, the penalty term punishes the behavior that does not meet the constraint conditions so as to force the strategy to learn to generate the behavior that meets the constraint conditions. PPO-clip can synchronously control the amplitude of the update strategy when optimizing the strategy, avoid the drastic changes of the strategy caused by an update amplitude that is too large, and effectively ensure the stability of the algorithm. PPO-clip uses the shear function to ensure that the policy is updated within a predetermined range while retaining the better parts of the original policy, improving policy performance and stability.

3.2. Hierarchical Reinforcement Learning

Hierarchical ideas are sourced when solving a complex problem, where it can be broken down into multiple simple sub-problems. When using deep reinforcement learning to solve problems, when the environment is very complex or the task is relatively difficult, it

is difficult for the algorithm to obtain the ideal effect due to the huge agent action and state space, so currently, hierarchical reinforcement learning is mainly used to handle sparse reward and state action space caused by training difficulties [30].

The methods of hierarchical reinforcement learning are mainly divided into option-based and goal-based methods. The upper controller selects the option/goal in a long time span, and the lower controller selects the action according to the option/goal in a short time span. The upper controller not only provides options/goals for the lower level, but can also feed back the corresponding intrinsic reward according to the quality of the lower strategy. Even if the external reward is 0, the lower controller can still obtain the reward so as to alleviate the sparse reward. The representative algorithm includes the Feudal Learning method, option-based method, and a method based on the MaxQ (maximum cumulative reward value) function decomposition.

The feudal hierarchical learning method divides the problem into multiple layers; the upper layer calls the lower level to solve the problem, and the lower level carries out the command of the upper level. In the option-based method, the option is selected by the top level and then by the option. Methods based on the MaxQ value can artificially break down the Markov process into multiple subtasks.

3.3. H-PPO

In path planning tasks with high dimensionality, complex constraints, or high global optimization requirements, single-level reinforcement learning strategies (e.g., making decisions directly for the final goal) often face challenges such as a large search space, slow training convergence, and local optimal traps. To cope with these problems, the introduction of the HRL (Hierarchical Reinforcement Learning) concept becomes an effective solution. Under the HRL framework, different levels of strategies are assigned specific decision-making tasks, which effectively reduces the decision space and improves the training efficiency and strategy quality.

The UAV maneuvering and obstacle-avoiding problems need to consider the UAV's self-performance and a variety of constraints in order to further improve the performance and stability of the algorithm and obtain more ideal results. The H-PPO (Hierarchical Proximal Policy Optimization) method is proposed in this paper, which combines the HRL method based on the option and the PPO method.

In the hierarchical reinforcement learning algorithm part, the maneuver breakout task is separated into two networks, high-level and low-level networks. The high-level network takes the current state and maneuvering path of the UAV as an input and outputs a sub-maneuvering target, that is, the next target position. The high-level network divides the whole maneuvering task into multiple sub-tasks, and when the current sub-maneuvering goal is completed, the next sub-goal is continued. The high-level network outputs a macro strategy, while the low-level network takes the subtarget that needs to be executed as the input and outputs the micro instructions, such as the maneuver angle, speed, and acceleration transformation of the agent. Both levels adopt the PPO algorithm to deal with problems, which simplifies the maneuvering decision-making problem in a complex environment by introducing the idea of hierarchical reinforcement learning. The high-level network guides the UAV maneuvering process through the low-level network, and the high-level network only needs to focus on the completion of sub-objectives, while the low-level network executes complex and specific actions, so each level can focus on solving specific tasks. Thus, the robustness of the system is enhanced. The low-level network proposed in this paper operates at the same frequency as the high-level network, which can greatly reduce maneuver errors and achieve safer and more efficient maneuvering.

3.3.1. Network Principle

In the H-PPO path planning structure explored in this paper, the high-level policy and the low-level policy work closely together through a sub-goal mechanism. The model flow is shown in Figure 3.

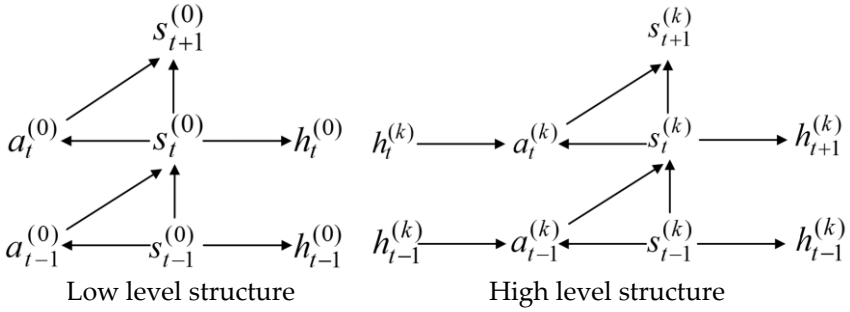


Figure 3. Network structure diagram.

Each layer is viewed as a strategy, given k layers, $k \in \{0, 1, \ldots, K\}$, in which action a_t^k is selected from strategy $\pi_{\theta_k}\left(s_t^{(k)} \middle| a_t^{(k)}, h_t^{(k)}\right)$, where π_{θ_k} represents the strategy of layer k, i.e., the probability distribution of the selected action. θ_k represents a parameter of the strategy of layer k, $h_t^{(k)} \in H^{(k)}$ represents the possible uncertainty in the previous layer, $H^{(k)}$ denotes the potential space, and t is the time. The lower layer is able to formulate the Markov decision process by selecting action a_t^0 from $\pi_{\theta_0}\left(s_t^{(0)} \middle| a_t^{(0)}\right)$, inputting potential uncertainties to the higher layer, which receives the uncertainties from the lower layer and also provides potential uncertainties.

Specifically, high-level strategies focus on macro-planning of the global task, with decisions acting on a longer time scale, based on current state and route information, by selecting intermediate sub-goal points (or regions), i.e., selecting a larger action in the environment to drive the agent closer to the goal, and pointing out the direction for the low-level strategies. These sub-goals are often located at several key location points between the current state of the agent and the final goal state to form a hierarchical decomposition path. Instead of having to make frequent decisions about each step of the action, the high-level strategy guides the low-level strategy to gradually approach the final goal by setting clear intermediate task goals. This move is essentially equivalent to breaking down the original large and potentially redundant search problem into a series of relatively simple and short-term reachable sub-problems.

In terms of sub-goal selection, every interval has a certain number of steps, the high-level network will determine whether a new sub-goal needs to be selected, and if the current sub-goal has already been accomplished, i.e., the agent has already arrived at the sub-goal position or the goal position, then the sub-goal will be selected again.

Low-level strategies, on the other hand, focus on executing specific behavioral decisions on a local scale. Given the current state (e.g., the coordinates of the agent in a 2D grid) and a sub-goal from a higher level, the low-level strategy uses PPO for updating in order to quickly adapt to changes in the environment and fine-tune the action selection, including changes in velocity, angle, and acceleration. The role of the low-level policy is more similar to that of a "controller"; it is responsible for guiding the agent from its current state to the sub-goal position robustly and efficiently at a local scale, avoiding collisions with obstacles and unnecessary movement costs. By applying the PPO-based Policy Gradient method to the low-level policy, the low-level policy update can achieve relatively smooth performance improvement, making the local optimization more efficient and stable under the guidance of the high-level goal.

The high-level strategy and the low-level strategy are designed for different tasks, so each level is designed with a separate reward function R_k.

The framework of the H-PPO network is shown in Figure 4.

The Algorithm 1's steps are described as follows:

Algorithm 1: H-PPO

Initialize high-level policy parameters θ_h, value parameters ϕ_h
Initialize low-level policy parameters $\theta_{\updownarrow}$, value parameters $\phi_{\updownarrow}$
Set total training steps H, rollout length T, high-level interval M, PPO clipping range ε, discount factor γ, GAE parameter λ, and number of optimization steps N
for iteration = 1 to max_iterations **do**
  $s \leftarrow$ env.reset()
  $done \leftarrow False$
  Clear high-level buffer D^H and low-level buffer D^L
  Obtain initial subgoal $g \sim \pi_h(\cdot|s;\theta_h)$
  $h_state \leftarrow s$
  for $t = 1$ to T **do**
    $a \sim \pi_{\updownarrow}\left(\cdot\middle|s,g;\theta_{\updownarrow}\right)$
    $(s', r, done, info) \leftarrow$ env.step(a)
    Store $(s, g, a, r, done)$ in D^L
    $s \leftarrow s'$
    if t mod $M = 0$ or $done$ **then**
      $R_h = \sum_{\Gamma=t-M+1}^{t} r_\Gamma$
      Store $(h_state, g, R_h, done)$ in D^H
      $h_state \leftarrow s$
      if not $done$ **then**
        $g \sim \pi_h(\cdot|s;\theta_h)$
      end if
    end if
    if $done$ **then**
      $s \leftarrow$ env.reset()
      $h_state \leftarrow s$
      $g \sim \pi_h(\cdot|s;\theta_h)$
    end if
  end for
  $(A_h, G_h) \leftarrow \mathrm{GAE}\left(D^H, \gamma, \lambda, V(h_state;\phi_h)\right)$
  $\left(A_{\updownarrow}, G_{\updownarrow}\right) \leftarrow \mathrm{GAE}\left(D^L, \gamma, \lambda, V\left(s, g;\phi_{\updownarrow}\right)\right)$
  for $update_step = 1$ to N **do**
    Compute $r(\theta_h)$ and optimize high-level PPO loss:
    $L^h_{policy} = -\mathrm{E}[\min(r(\theta_h)A_h, \mathrm{clip}(r(\theta_h), 1-\varepsilon, 1+\varepsilon)A_h)]$
    $L^h_{value} = \mathrm{E}\left[\left(V(h-state;\phi_h) - G_h\right)^2\right]$
    Update (θ_h, ϕ_h)
  end for
  for $update_step = 1$ to N **do**
    Compute $r\left(\theta_{\updownarrow}\right)$ and optimize low-level PPO loss:
    $L^{\updownarrow}_{policy} = -\mathrm{E}\left[\min\left(r\left(\theta_{\updownarrow}\right)A_{\updownarrow}, \mathrm{clip}\left(r\left(\theta_{\updownarrow}\right), 1-\varepsilon, 1+\varepsilon\right)A_{\updownarrow}\right)\right]$
    $L^{\updownarrow}_{value} = \mathrm{E}\left[\left(V\left(s.g;\phi_{\updownarrow}\right) - G_{\updownarrow}\right)^2\right]$
    Update $\left(\theta_{\updownarrow}, \phi_{\updownarrow}\right)$
  end for
end for

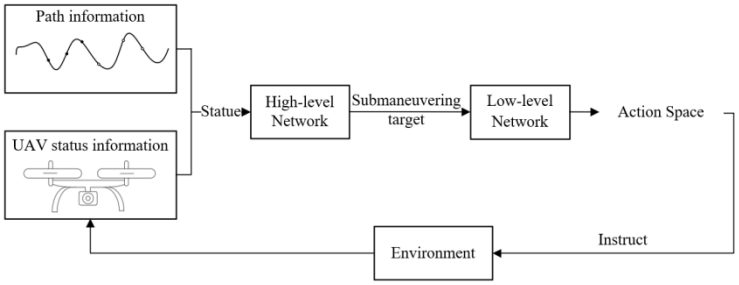


Figure 4. Framework of H-PPO.

3.3.2. Network Training

The high-level and low-level strategies are trained and optimized independently at their respective time scales and task abstraction levels but are effectively linked through sub-goal information sharing. While high-level strategies learn to select meaningful sub-goals (e.g., progressively advancing in chunks from the starting point toward the goal or bypassing the anchor position of a specific obstacle band) after passing through several training phases, low-level strategies gain experience in continuously executing and optimizing specific path control actions. During training, whenever an intelligent agent completes a sub-goal or fails in a collision during an attempt, the high-level strategy obtains the corresponding delayed reward feedback based on the sub-goal completion or the change in the distance to the final goal, thus adjusting the future sub-goal selection strategy. The goal of training the high-level strategy network is to maximize the rewards for reaching the sub-goal and to guide the low-level network to select the actions to maneuver in the right direction.

The goal of the low-level policy network is to select specific actions that enable the intelligent agent to approach the sub-goal safely and efficiently. The low-level network learns the optimal action strategy through a combination of states and sub-goals. The low-level policy optimizes the immediate reward for each time step and each local action, ensuring that the agent's behavioral choices are progressively improved by the given sub-goals.

The high-level and low-level networks work together during network training. The high-level network selects a sub-goal based on the state, and the low-level network performs specific actions based on the current state and that sub-goal. If the low-level network successfully reaches the sub-goal, or if a collision or timeout occurs, it triggers the high-level network to re-select the sub-goal.

This reward decomposition approach allows high-level strategies to focus mainly on longer-term planning effects (e.g., reducing the total distance to the goal or the overall layout of bypassing a large obstacle), while the low-level strategies can flexibly cope with local constraints and fine-tune the paths. This not only accelerates the speed of strategy convergence, but also avoids the problem of credit allocation faced by traditional single-layer reinforcement learning in complex environments.

In addition, from the perspective of the algorithm learning process, the hierarchical PPO structure utilizes the core idea of PPO, i.e., the use of the moderately tailored objective function and AF (Advantage Function) estimation for the policy update to avoid the instability triggered by a policy update that is implemented too quickly. Since the hierarchical mechanism eliminates the need for the high-level strategies to change their target points too frequently, the strategy distribution of the low-level strategies has more opportunities to be fully explored and optimized under the guidance of relatively stable sub-objectives. The high-level strategies also have the ability to plan at the overall level by periodically reassessing the global situation (e.g., deciding on a sub-goal every few steps), which reduces the behavior of blind search in high-dimensional environments.

4. UAV Maneuvering Decision Model

In this paper, the H-PPO algorithm is used to deal with the UAV autonomous maneuvering path planning problem with the aim of controlling the agent to select the optimal decision to execute the action based on the state of the situation at a certain moment (s_t, a_t), to obtain the corresponding rewards R_{t+1}, and to realize the autonomous maneuvering decision of the UAV in maneuvering obstacle avoidance only.

4.1. State Space

The state space of the UAV in the maneuvering environment includes its own state, the state of the threat area, and the target relative state. The UAV detection area is shown in Figure 5. The blue area indicates the safe region, which does not pose a threat to the UAV and can maneuver normally. The gray area represents the target area that poses a threat to UAV maneuvering, such as birds, other UAVs, fixed obstacles, etc. The red area indicates the warning area, that is, if the threat area is detected, it will enter the threat area and affect the normal maneuvering of the UAV.

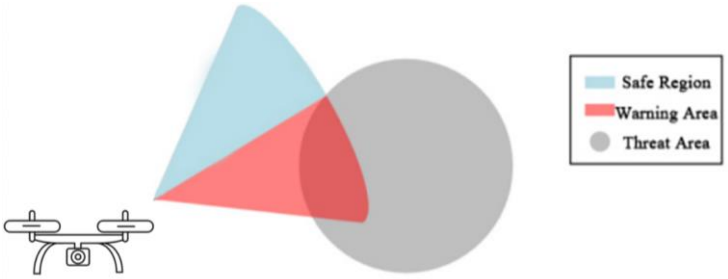


Figure 5. Diagram of UAV detection areas.

The UAV maneuvering mission area is set as L_l in length and L_w in width, and its own state description expression is

$$S_m = \left[\frac{x}{L_w}, \frac{y}{L_l}, \frac{v}{v_{\max}}, \frac{\alpha}{\pi}, t\right] \tag{20}$$

where (x, y) are the coordinates of the UAV's position in the area, v denotes the velocity, α is the angle between the maneuvering direction and the y-axis, $\alpha \in [-180°, 180°]$, and t is the maneuvering time.

The threat zone state describes the relative position of static/dynamic threats, no-fly zones, and terrain obstacles present in the UAV at the current moment, as shown in Figure 6, with the expression

$$S_d = \left[\frac{l_1}{L}, \frac{l_2}{L}, \ldots, \frac{l_6}{L}\right] \tag{21}$$

where l_n denotes the distance between the UAV position coordinates and the edge of the threat zone, L denotes the detection radius, and $l_n \leq R$. The UAV detection model is shown in Figure 6. The gray area represents the threat target area, the blue area represents the drone detection area, and $l_1 \sim l_6$ express the detection range. $l_1, l_6 = L$, that is, no threat area is detected, and $l_2, l_3, l_5 < L$ indicates that the threat area is detected.

The target relative state is denoted as

$$S_t = \left[\frac{x_t}{L_w}, \frac{y_t}{L_l}, \frac{\varphi}{\pi}\right] \tag{22}$$

where (x_t, y_t) denotes the coordinates of the target in the mission area, φ is the azimuth of the target point, and $\varphi \in [-180°, 180°]$.

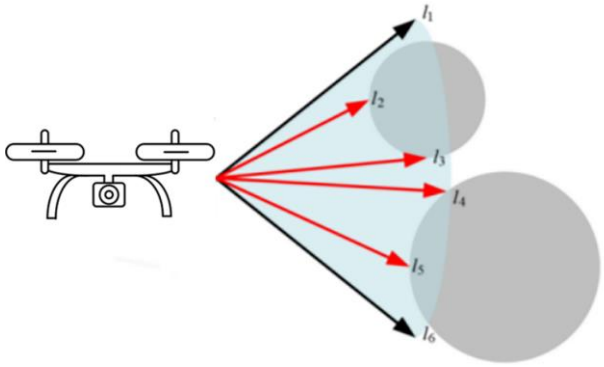


Figure 6. UAV detection model.

The total state space is

$$S = [S_s, S_d, S_t] \tag{23}$$

4.2. UAV Action Space

The rate of change of velocity set during UAV maneuvering is denoted by a, the angular velocity is denoted by ω, and its action space is denoted as

$$A_m = [a, \omega] \tag{24}$$

Considering the power and load limitations of the UAV, the maximum acceleration $a_{\max}$ and the maximum angular velocity $\omega_{\max}$ are set.

4.3. Setting Reward Function

The effective setting of the reward function is the key for the algorithm to play an effective role; the agent obtains the feedback after executing the action, that is, the reward value, which is able to effectively evaluate the rationality of the behavioral decision, thus guiding the agent to fully learn to choose the action decision with a higher reward. The UAV maneuvering obstacle avoidance task is complex, and the reward function of the high-level strategy network is set as

$$Ra = \begin{cases} -10, & \text{incomplete the sub-goal} \\ 100, & \text{complete the sub-goal} \end{cases} \tag{25}$$

Rewards are given when the artificial intelligence completes the sub-goal, and penalties are given when it does not complete the sub-goal.

The reward function for the low-level strategy network can be divided into four sub-aspects to set the reward value, including maneuver guidance, threat zone avoidance, route optimization, and task completion.

(1) The motorized bootstrap reward function is set to

$$R_{app} = d_{t-1} - d_t \tag{26}$$

d_{t-1} represents the distance between the UAV and the target point at the previous moment $t-1$, and d_t represents the distance between the drone and the target point at the current t moment in km. R_{app} indicates that the closer the drone is to the target, the greater the reward value is.

(2) The threat avoidance reward function is set to

$$R_{safe} = \sum_{n=1}^{3} l_n + \min(l_{4\sim6}) - 4L \tag{27}$$

The formula l_1~l_3 represents the detection distance of no threat in front of the maneuver, and l_4~l_6 represents the distance facing the threat zone. During detection, if the

detection distance is not less than the detection radius, and the threat zone is detected, then a negative reward is obtained, and if it is greater than the detection radius, then a positive reward is obtained.

(3) The path-finding reward function is set to

$$R_{mot} = v_m \cos \alpha \tag{28}$$

ϕ indicates the angle between the target's bearing and the direction of speed, gaining a greater bonus value when the UAV maneuvers quickly toward the target.

(4) Rewards for task completion

The ultimate goal of achieving autonomous maneuvering obstacle avoidance for UAVs is to reach the specified target point. In this regard, a reward function for obstacle avoidance and a reward function for reaching the specified target point are distinguished. During the maneuvering process, the UAV will face multiple threatening situations, and each successful obstacle avoidance will calculate a reward, and when the UAV reaches the target point, it will then calculate the task completion reward.

Avoidance rewards:

$$R_p = \begin{cases} 100, & \text{(Successfully avoided obstacles)} \\ -50, & \text{(Unsuccessfully avoided obstacles)} \end{cases} \tag{29}$$

The reward for reaching the specified target point is

$$R_s = \begin{cases} 60, & \text{(Arriving at the target point)} \\ 5, & \text{(Not Arrie)} \\ -0.001t, & \text{(othertime)} \end{cases} \tag{30}$$

Due to the complexity of the UAV maneuvering obstacle avoidance task and the difficulty of learning, it is necessary to gradually guide the agent to learn the optimal task completion strategy, and on the basis of the traditional negative reward given for not completing the task, a smaller positive reward is set to achieve the purpose of gradually guiding the agent to learn.

In order to improve the efficiency of the algorithm operation and to ensure that the task is executed within the given time, a reward of $-0.001\ t$ for other times is set within the reward function for reaching the specified target point, indicating that a negative reward is obtained with the prolongation of time.

When combining the above reward functions, the total reward function in the UAV maneuvering obstacle avoidance task is

$$R = \varepsilon_1 R_{app} + \varepsilon_2 R_{safe} + \varepsilon_3 R_{mot} + \varepsilon_4 R_p + \varepsilon_5 R_s \tag{31}$$

ε is introduced to represent each sub-reward coefficient to balance the degree of influence of different reward values [31].

5. Experimental Results and Analysis

5.1. Experimental Environment

In this paper, the PyTorch framework is used to construct H-PPO (Hierarchical Proximal Policy Optimization) networks in the Python3.0 programming environment, and the computer configurations used for algorithm training and testing are Intel(R) Xeon(R) Platinum i9-13900k from Intel company in Santa Clara, CA, USA, and Nvidia GeForce RTX 4090 from Nvidia Company in Santa Clara, CA, USA; additionally, 32 GB of operating memory is used. The task scenario is a finite two-dimensional area of 600 m $\times$ 1000 m, as shown in Figure 7. The

red dots in the figure indicate the UAV departure locations, which are set to be randomly generated on the short side of the mission area as follows: $x_0 \in [0, 60]$, $y_0 \in [0, 300]$. The blue circle indicates the target location, and the coordinate points are randomly generated in the area of $x_t \in [900, 1000]$, $y_t \in [500, 600]$. The gray circle indicates the static threat area, and the orange circle indicates the dynamic threat area.

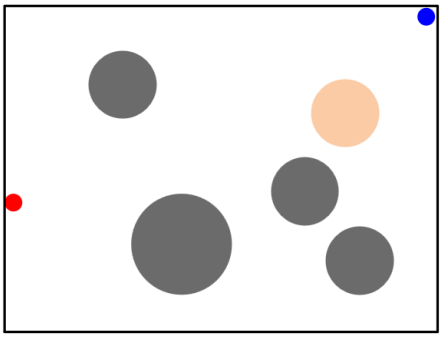


Figure 7. A schematic diagram of the experimental task area.

5.2. Experimental Parameters

The UAV performance parameters are shown in Table 1.

Table 1. Drone performance parameters.

Mission Parameters	Numerical Value	Unit
Cruising speed	5	m/s
Maximum speed	15	m/s
Accelerations	10	m/s^2
Detection angle	60	rad
Detection radius	0.2	km
Threat area radius	10–200	m

The model parameters were set as shown in Table 2.

Table 2. Experimental parameter settings.

Parameter Name	Numerical Value
Maximum number of training rounds E	500
Maximum training round step length T	600
Hierarchy update frequency T_{train}	20
Learning rate l	0.001
Discount factor γ	0.995
GAE parameters λ	0.95
PPO clipping parameters	0.2

Each time the artificial intelligence completes the task, enters the threat zone, or reaches the maximum number of steps in the round, the round is terminated, at which point the environment is reset and a new round of training begins. The initial value of the network learning rate is set to 0.001.

ε_1, ε_2, ε_3, ε_4, and ε_5 are set to 0.1, 0.2, 0.2, 0.3, and 0.2. A high-level policy network, MLP (Multi-Layer Perceptron), includes three fully connected layers with 64 hidden units, and the activation function is ReLU. A low-level policy network, MLP, includes three fully connected layers, each of which has 256 hidden units, and the activation function is ReLU.

5.3. Results Analysis

In order to demonstrate the scalability and performance of the algorithm in complex scenarios, the obstacle numbers are set as 5, 10, and 15, respectively, to conduct comparison experiments. The H-PPO algorithm was trained for 500 rounds, and the initial network learning rate was 0.001. The task area contained five threat areas, which included one dynamic threat area and four static threat areas. The change curve of the reward function is shown in Figure 8. In order to verify the effectiveness of the learning rate setting, the comparison experiment with the learning rate micros of 0.1, 0.01, and 0.0001 was added here.

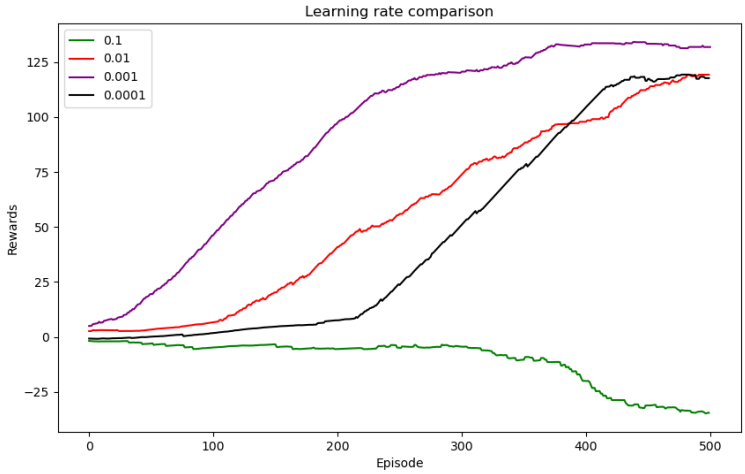


Figure 8. Reward function curves for different learning rates.

As shown in Figure 8, the learning rate is set at 0.001. After 500 rounds of training, the reward function value of the network rises the fastest and becomes stable the earliest, and the value of the reward function is the highest compared with other settings.

In order to further verify the performance of the H-PPO algorithm, the comparison experiment was carried out in the same experimental environment, the other parameters of the experiment were set unchanged, and the change curve of the network reward function was obtained as shown in Figure 9.

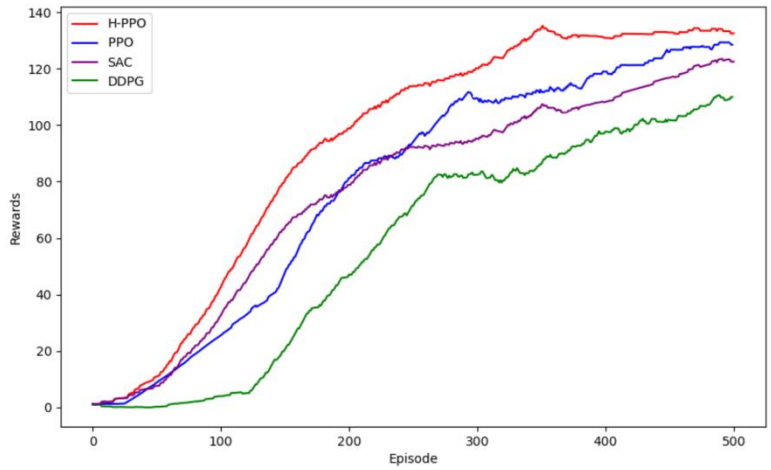


Figure 9. Maneuver reward curve in training phase (5 obstacles).

As shown in Figure 9, the abscissa represents the training round, and the ordinate indicates the reward value of the training. With the iteration of learning over multiple rounds, the H-PPO reward curve rose rapidly between 50 and 330 rounds and gradually converged to stability after reaching 137 at around the 360th round. There were no large

fluctuations, and the convergence rate was faster. The reward value of the PPO algorithm started to rise rapidly after training for 130 rounds, but the increase was less than that of the H-PPO algorithm, and the increase in the 390th round slowed down and gradually converged to about 129. Compared with the PPO algorithm, the H-PPO algorithm can achieve faster convergence, and the model achieved a higher reward value, which proves that the combination of hierarchical reinforcement learning and PPO can accelerate network convergence and improve network performance. The reward function value of the SAC algorithm began to rise rapidly in the 80th round, and the rise rate was faster than that of PPO algorithm at the initial stage and became slower after around 230 rounds, and the reward function value was lower than that of the PPO algorithm. The reward function value of the DDPG algorithm began to rise rapidly around the 120th round, but the increase rate was lower than that of the PPO algorithm, and the reward value of the network after 500 rounds of training was about 110, which is significantly lower than that of the H-PPO, PPO, and SAC algorithms. Compared with the other two networks, the fluctuation in the DDPG network reward function was greater, and the model performance was not stable enough. By comparison, it can be concluded that the H-PPO algorithm constructed in this paper effectively improved the network convergence speed and model performance and obtained a higher reward value.

To further verify the model robustness in complex task scenarios, we tested the maneuver decision against multiple-threat target performance under the premise of the experiment in other conditions. The threat target number was 10, including 3 dynamic threat targets and 7 static threat targets, which were still randomly distributed in the task area, and the model reward function graph is shown in Figure 10.

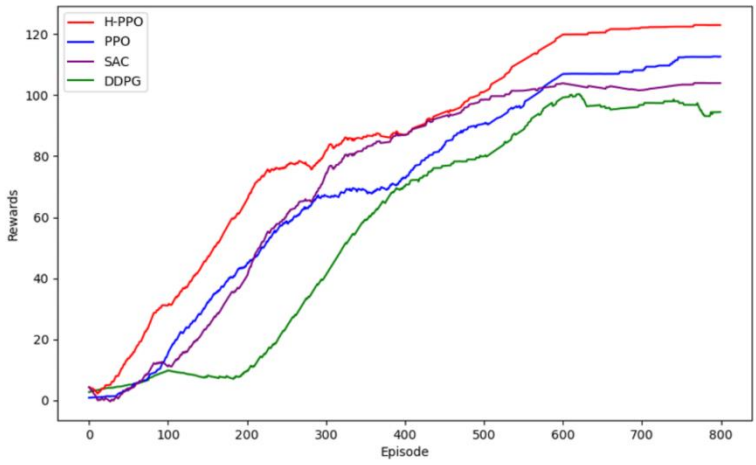


Figure 10. Maneuver reward curve in training phase (10 obstacles).

Due to the increase in the number of threat targets, the complexity of the task scene deepens, and we set the number of training rounds to 800. Figure 10 shows that the H-PPO algorithm, after starting training at about the 10th round, the reward function value rose rapidly, and the rising trend was significantly higher than those of the PPO algorithm, SAC algorithm, and DDPG algorithm. This shows that the model achieved good convergence at around the 580th round, the reward function value reached 120, which is significantly higher than the other two models, and the DDPG had significantly more changes in amplitude; additionally, the network was still fluctuating toward the end of 800 rounds of training, and its performance was not stable.

We continued to increase the number of threat targets to 15, including 7 dynamic threat targets and 8 static threat targets. The graph of the resulting model reward function is shown in Figure 11.

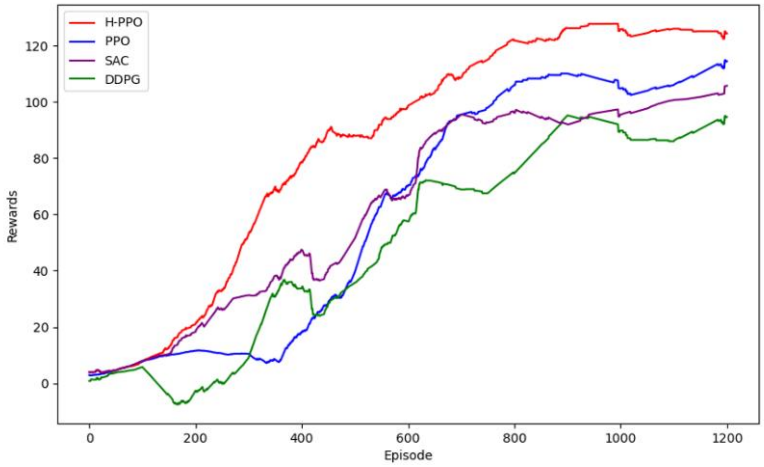


Figure 11. Maneuver reward curve in training phase (15 obstacles).

With 1200 rounds of training, the H-PPO algorithm can be obtained from the other two algorithms. The reward function value starts to rise significantly after the 170th round, earlier than the PPO algorithm, SAC algorithm, and DDPG algorithm, and the reward function value grows faster, with the network converging at around 950 rounds, and the reward function value is around 120. However, the PPO algorithm, SAC algorithm, and DDPG algorithm are not well converged; in particular, the SAC algorithm and DDPG algorithm fluctuate greatly, and the model is not well converged and does not have strong ability to cope with complex environment changes.

With the increases in the number of threat targets and the proportion of dynamic threat targets, H-PPO algorithm reward function value can achieve better convergence compared to the PPO algorithm and DDPG algorithm; it can also converge faster and obtain a higher final reward value, proving that the H-PPO algorithm can better realize intelligent maneuver decision in a complex dynamic environment, and the network's decision performance and robustness are stronger.

The average maneuver success rate of 100 tests was calculated, and the results are shown in Figure 12. The abscissa indicates the number of threat targets, and the ordinate indicates the success rate.

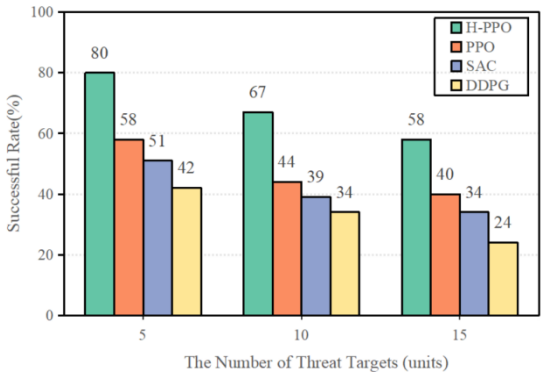


Figure 12. Number of movable threat targets and success rate.

It can be seen when comparing multiple obstacles in the experiment that with the increase in the number of obstacles, the total number of obstacles the H-PPO algorithm is five, and its success rate of 80% is 22% higher than the PPO algorithm, 29% higher than the SAC algorithm, and 38% higher than the DDPG algorithm. With the dynamic threat target proportion, the H-PPO algorithm can still maintain a high success rate with a total threat target number of 15 and seven dynamic threat targets, accounting for 47%, and it

can still achieve a 58% success rate, while the PPO algorithm's success rate is 40%, the SAC algorithm's success rate is 34%, and the success rate of the DDPG algorithm is only 24%. The results prove that the algorithm can be trained to help the agent achieve a higher success rate in the unknown dynamic complex scene, the model is relatively robust, and its adaptability to a complex environment is strong.

In order to verify the effectiveness and practical application value of the algorithm for mobile penetration, the algorithm's performance in 100 tests was measured when the number of threat targets was 5, 10, and 15 respectively, and the results are shown in Tables 3–5.

Table 3. Algorithm performance comparison (5 obstacles).

Algorithm	Success Rate (%)	Average Maneuver Range (km)	Average Flight Time (s)	Average Calculation Time (s)
H-PPO	80	96	128	1.64
PPO	58	144	192	2.46
SAC	51	161	216	5.75
DDPG	42	192	256	3.21

Table 4. Algorithm performance comparison (10 obstacles).

Algorithm	Success Rate (%)	Average Maneuver Range (km)	Average Flight Time (s)	Average Calculation Time (s)
H-PPO	67	141	188	2.44
PPO	44	189	252	3.19
SAC	39	216	284	6.68
DDPG	34	237	316	3.82

Table 5. Algorithm performance comparison (15 obstacles).

Algorithm	Success Rate (%)	Average Maneuver Range (km)	Average Flight Time (s)	Average Calculation Time (s)
H-PPO	58	147	196	2.54
PPO	40	194	259	3.14
SAC	34	224	297	7.49
DDPG	24	263	351	4.37

It can be seen from Table 3 that when the number of threat targets in the task scenario is five, the success rate of the H-PPO algorithm is the highest, reaching 80%, which is significantly higher than the success rate of the PPO algorithm (58%), SAC algorithm (51%), and DDPG algorithm (42%). The average maneuvering range is 96 km, which is 48 km shorter than the PPO algorithm, 65 km shorter than the SAC algorithm, and 96 km shorter than the DDPG algorithm. The average flight time is 128 s, 64 s faster than the PPO algorithm, 88 s faster than the SAC algorithm, and 128 s faster than the DDPG algorithm. The average calculation time of the model is only 1.64 s, and the calculation speed is faster, making it suitable for real-time mission scenarios.

The number of threat targets was increased to 10, the decision-making performance of the algorithm in complex environments was verified, and 100 experimental results were counted and are presented in Table 4.

The number of threat targets was increased to 15, and the decision-making performance of the algorithm in complex environments was verified. The results of 100 experiments were counted, and Table 5 was obtained.

As can be seen from Tables 3–5, with the increase in the number of threat targets and the deepening of environmental complexity, the success rate of the H-PPO algorithm decreases, but when the number of threat targets is 15, the success rate still reaches 58%, and the average maneuver range, time, and model calculation time are significantly lower than the PPO algorithm, SAC algorithm, and DDPG algorithm. It is proven that the H-PPO algorithm has stronger adaptability to a complex environment, better model robustness, and can better realize intelligent autonomous decision-making tasks in complex environments.

The visual effect generated by the network test is shown in Figure 13. In the experiment, the static threat area is represented by the gray area, the orange area is the dynamic target area, and the position and size are generated randomly. The red dot represents the location of the UAV, the blue dot represents the location of the established target, and the red curve represents the maneuvering route of the UAV. The three figures in Figure 13, respectively, show the UAV maneuvering schematics when the number of static areas is 4, 7, and 8 and when the corresponding number of dynamic areas is 1, 3, and 7. The meanings of the different colored areas in this figure are the same as those in Figure 7.

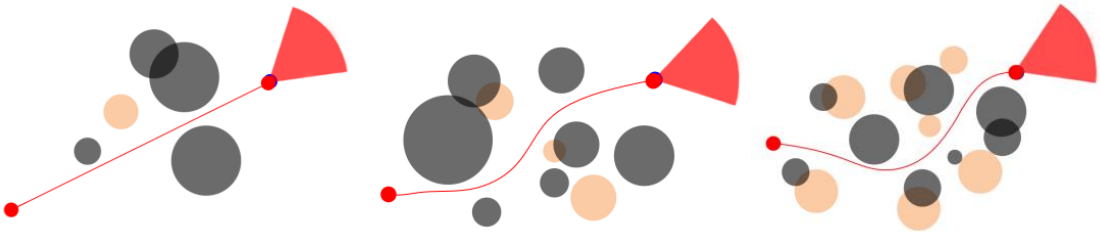


Figure 13. Intelligent agent maneuver test diagram.

As can be seen from Figure 13, in the process of controlling the agent's maneuver through H-PPO, the agent can choose the optimal path and maneuver to the designated target location with the shortest distance under the premise of effectively avoiding the threat area, with good ability to deal with multiple threat areas.

The number of static threat areas is set to 7, the number of dynamic threat areas is set to 3, the movement speed of dynamic threat areas is 2 m/s, and the position is generated randomly. The visualization results of the test are shown in Figure 14. The meanings of the different colored areas in this figure are the same as those in Figure 7.

As can be seen from Figure 14, after the trial starts, the agent can continuously maneuver to the established target area and can choose the best path between the starting point and the trend of the threat target. At 40 s, it avoids the first closer threat target, effectively makes the avoidance decision on the threat area in the maneuvered route, and replans the maneuver direction to find the optimal path to move forward, showing good generalization and adaptability to complex scenarios. At 96 s, the agent chooses a closer route between the two maneuver directions and passes through, indicating that the trained agent can solve more complex maneuver decision problems. At 126 s, the agent safely and efficiently reaches the specified target point. The whole experiment verifies the feasibility of the algorithm to realize autonomous path planning in complex dynamic regions, and it has good robustness and generalization.

In this experiment, the UAV starting point and target point position are randomly generated. The number and radius of threat targets in the mission area are diverse. In the complex environment, the agent can effectively achieve the goal of security maneuver,

effectively avoid threat areas, and choose the optimal path in the maneuver with no obvious risky behavior, showing that the model has good versatility.

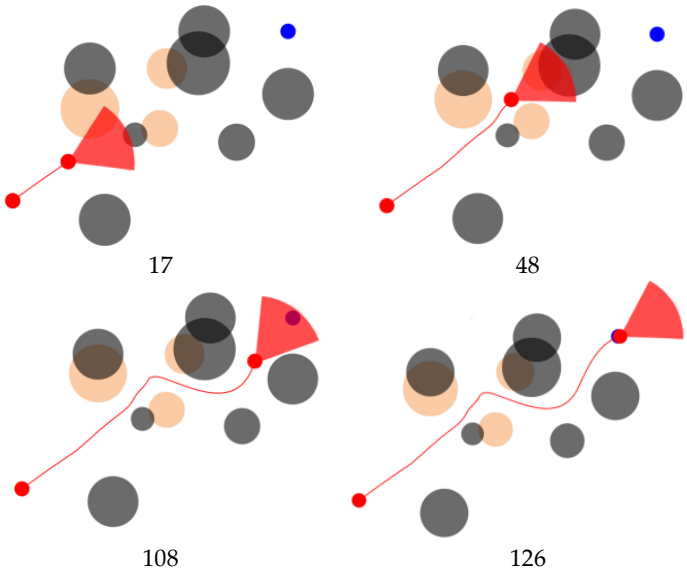


Figure 14. Dynamic scene test diagram.

The maneuvering speed is one of the important performance indexes of UAVs. The maneuvering speed directly affects the efficiency of mission completion and response speed. In a complex dynamic environment, it is necessary to react quickly and adjust the maneuvering strategy in time in the face of multi-target threats. A faster flight speed means that a more efficient maneuver decision-making network is needed to adjust maneuvering strategies in time. In order to further verify the influence of speed on the maneuvering success rate, different maximum speeds are set under the condition that the experimental environment and other parameters are unchanged, and the success rate of maneuvering to the specified target position is determined in 100 tests, respectively. The experimental results are shown in Figure 15.

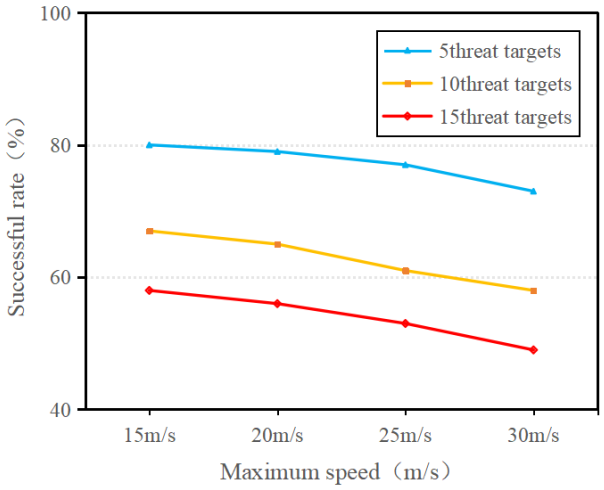


Figure 15. Relationship between success rate and maximum maneuvering speed.

As can be seen from Figure 15, with the increase in the maximum maneuvering speed, the success rate of the UAV in performing maneuvering tasks decreases somewhat, but the decrease is not large. An increase in speed will increase the possibility of collision with the threat target. However, the H-PPO network deals with the maneuvering decision problem using the hierarchical method, which effectively reduces the complexity of the

stage maneuvering problem. And the low-level strategy can adjust the maneuver strategy more accurately under the guidance of the high-level strategy. In this way, the UAV can quickly make more accurate obstacle avoidance actions at a higher maximum flight speed, thus maintaining the success rate.

6. Conclusions

This study focuses on the UAV autonomous maneuver decision problem in a complex and large decision space, considering that in the typical processing of a continuous state action space, the deep reinforcement learning algorithm is prone to gradient disappearance and gradient explosion, and it deals with complex maneuver decision problems poorly. Thus, this paper puts forward an H-PPO intelligent maneuvering obstacle avoidance algorithm through the high-strategy and low-strategy PPO, respectively, and using the target mechanism to establish a decision-level division framework, the algorithm can effectively improve the solution efficiency of complex path planning problems and solution quality. The high-level strategy decides the "path skeleton" of the high-dimensional plan. The low-level strategy is responsible for filling the fine behavior execution on this skeleton. Under the synergistic action of the two, the algorithm can maintain stable and fast convergence to achieve a high-quality solution in the face of large-scale navigation tasks with numerous obstacles and sparse feasible paths. In the next step, considering the close connection between the UAV maneuvering process and the actual environment and the corresponding temporal and spatial characteristics, in order to improve the robustness of the method and the rationality of the decision output, it is considered to introduce the Long Short-Term Memory (LSTM) network into the H-PPO network to fully consider the current environmental state. Additionally, the relationship between the action and the environment used in the past is considered to develop a more reasonable and efficient maneuver decision scheme. At the same time, considering the application in future practical scenarios, ArcGIS will be used to convert the environment and target location information from the actual scenarios into experimental scenarios and further explore the application value in the actual environment.

Author Contributions: Y.W. was responsible for the resources, formal analysis, program compilation, and writing of the original draft. Y.J. was responsible for writing—review and editing. H.X. was responsible for the methodology and project administration. Y.J. was responsible for the investigation and resources. C.X. was responsible for the resources and formal analysis. K.Z. was responsible for the resources and formal analysis. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article.

Acknowledgments: The authors would like to express their gratitude to Zhu Pingyun, an advisor, for his control and guidance regarding the overall idea of this article during the research period. At the same time, the authors would like to thank Xiao Chuanliang for his careful teaching and guidance regarding the structure of the article.

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

Abbreviations

UAVs	Unmanned Aerial Vehicles
PPO	Proximal Policy Optimization
HRL	Hierarchical Reinforcement Learning
H-PPO	Hierarchical Proximal Policy Optimization

RL	Reinforcement Learning
DL	Deep Learning
DDPG	Deep Deterministic Policy Gradient
DQfD	Deep Q-Learning from Demonstrations
DQN	Deep Q-Network
MDP	Markov Decision Process
TD3	Twin Delayed DDPG
SAC	Soft Actor–Critic
TRPO	Trust Region Policy Optimization
GAE	Generalized Advantage Estimation
LSTM	Long Short-Term Memory
MaxQ	Maximum Cumulative Reward Value
AF	Advantage Function
MLU	Multi-Layer Perceptron

References

1. Yin, S.H. *Research on UAV Maneuver Decision-Making Method based on the Game Mode*; University of Science and Technology of China: Hefei, China, 2023; pp. 20–35.
2. Wu, Y.N.; Zhou, H.A. Triangular intuitionistic fuzzy multi-attribute group decision making method based on improved TOPSIS. *J. Inn. Mong. Norm. Univ.* **2024**, *53*, 637–643.
3. Xie, X.J.; Ma, H.; Huang, P.; Xue, S.F. An interval intuitionistic fuzzy multi-attribute decision model based on projection measure and TOPSIS method. *Stat. Decis.* **2024**, *40*, 183–188.
4. Zhang, J.; Luo, X.-Y.; Li, Z.-X.; Zhao, Y. Research on UAV track model based on dynamic Bayesian network. *J. Saf. Sci. Technol.* **2023**, *19*, 188–193.
5. Jiang, L.C.; Shang, X.B.; Jin, B.; Zhang, W.; Zhang, Z. Ship trajectory prediction based on genetic algorithm-v support vector regression. *J. Harbin Eng. Univ.* **2024**, *45*, 2001–2006.
6. Nicholas, E.; Cohen, K.; Schumache, R.D. Collaborative tasking of UAVs using a genetic fuzzy approach. In Proceedings of the 51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, AIAA, Grapevine, TX, USA, 7–10 January 2013; p. 1032.
7. Pan, Y.H.; Zeng, Y.F. Research on interactive dynamic influence graph and its optimal K model solution. *Chin. J. Comput.* **2018**, *41*, 28–46.
8. Ehtamo, H.; Raivio, T. On applied nonlinear and bilevel programming or pursuit-evasion games. *J. Optim. Theory Appl.* **2001**, *108*, 65. [CrossRef]
9. Zhang, S.M.; Zhu, Y.W.; Yang, P.L.; Yang, F. Receding horizon optimization for spacecraft pursuit-evasion strategy in rendezvous. *J. Natl. Univerity Def. Technol.* **2024**, *46*, 21–29.
10. Chen, W.X.; He, D.F.; Liao, F.; Zhang, X.; Li, S. Rolling time domain planning of rendezvous trajectory between fixed wing and flying platform based on contractile moving block. *Chin. High Technol. Lett.* **2024**, *34*, 524–534.
11. Sutton, R.S.; Barto, A.G. *Reinforcement learning, An Introduction*; MIT Press: Cambridge, MA, USA, 2018; pp. 10–25.
12. Bellman, R. The theory of dynamic programming. *Bull. Am. Math. Soc.* **1954**, *60*, 503–515. [CrossRef]
13. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
14. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef] [PubMed]
15. Zhao, D.B.; Shao, K.; Zhu, Y.H.; Li, D.; Chen, Y.; Wang, H.; Liu, D.R. Review of deep reinforcement learning and discussions on the development of computer Go. *Control. Theory Appl.* **2016**, *33*, 701–717.
16. Qiu, X.Q.; Gao, C.S.; Jing, W.X. Maneuvering penetration strategies of ballistic missiles based on deep reinforcement learning. Proceedings of the Institution of Mechanical Engineers, Part G. *J. Aerosp. Eng.* **2022**, *236*, 3494–3504.
17. Zhang, W.Q.; Yu, W.B.; Li, J.L.; Chen, W. Cooperative reentry guidance for intelligent lateral maneuver of hypersonic vehicle based on downrange analytical solution. *Acta Armamentarii* **2021**, *42*, 1400–1411. (In Chinese)
18. Wang, Y.K.; Zhao, K.; Guirao, J.L.G.; Pan, K.; Chen, H. Online intelligent maneuvering penetration methods of missile with respect to unknown intercepting strategies based on reinforcement learning. *Electron. Res. Arch.* **2022**, *30*, 4366–4381. [CrossRef]
19. Guo, C.Y.; Liu, Z.F.; Tian, J.Z.; Liu, X. Heuristic multi-agent path finding VIA imitation learning and reinforcement learning. *Big Data Data Technol.* **2024**, *43*, 33–40.

20. Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Dulac-Arnold, G.; et al. Deep Q-learning from demonstrations[EB/OL]. *arXiv* **2017**, arXiv:1704.03732. Available online: https://arxiv.org/abs/1704.03732.pdf (accessed on 25 February 2021).
21. Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning[EB/OL]. *arXiv* **2017**, arXiv:1710.02298. Available online: https://arxiv.org/abs/1710.02298.pdf (accessed on 25 February 2021). [CrossRef]
22. Liu, G.; Lao, S.; Hou, L.; Yuan, C. A simulation system of anti-ship missile path planning oriented ship information. *Adv. Mater. Res.* **2012**, *532*, 645–649. [CrossRef]
23. Zhang, L. *On Path Planning and Attitude Control of Anti-Ship Missile*; Nankai University: Tianjin, China, 2014.
24. Zhang, Y.; Gong, D.W.; Zhang, J.H. Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing* **2013**, *103*, 172–185. [CrossRef]
25. Yang, Q.; Zhu, Y.; Zhang, J.; Qiao, S.; Liu, J. UAV air combat autonomous maneuver decision based on DDPG algorithm. In Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA), Edinburgh, UK, 16–19 July 2019; pp. 37–42.
26. Fu, X.W.; Xu, Z.; Zhu, J.D.; Wang, N. Research on maneuvering decision-making of multi-UAV attack-defence confrontation based on PER-MATD3. *Acta Aeronaut. Astronaut. Sin.* **2022**, *44*, 1–5. (In Chinese) [CrossRef]
27. Song, Y.Z.; Li, J.H.; Wang, H.J.; Su, X.; Yu, L. Path Planning algorithm of manipulator based on path imitation and SAC reinforcement learning. *J. Comput. Appl.* **2024**, *44*, 439–444.
28. Lv, X.L.; Zang, Z.X.; Li, S.B.; Wang, J. Attention-based Recurrent PPO Algorithm and Its Application. *Comput. Technol. Dev.* **2024**, *34*, 136–142.
29. Zhang, X.; Dong, W.H.; Yin, H.; He, L.; Zhang, P. Research ob Autonomous Maneuver Decision Method for Unmanned Aeial Combat Based on an Improved PPO Algorithm. *J. Air Force Eng. Univ.* **2024**, *25*, 77–86.
30. Pan, X.; Feng, G.L.; Hou, X.G. Research on AUV path tracking technology based on hierarchical reinforcement learning. *J. Nav. Univ. Eng.* **2021**, *33*, 106–112.
31. Citroni, R.; Di Paolo, F.; Livreri, P. A Novel Energy Harvester for Powering Small UAVs: Performance Analysis, Model Validation and Flight Results. *Sensors* **2019**, *19*, 1771. [CrossRef] [PubMed]

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