drones

MDPI

Article

Penetration Strategy for High-Speed Unmanned Aerial Vehicles: A Memory-Based Deep Reinforcement Learning Approach

Xiaojie Zhang [1,2,3], Hang Guo [2,3,4,*], Tian Yan [1,2,3], Xiaoming Wang [5], Wendi Sun [6], Wenxing Fu [1,2,3] and Jie Yan [2,3,4]

[1] Unmanned System Research Institute, Northwestern Polytechnical University, Xi'an 710072, China; xiaojzhang@mail.nwpu.edu.cn (X.Z.); tianyan@nwpu.edu.cn (T.Y.); wenxingfu@nwpu.edu.cn (W.F.)
[2] National Key Laboratory of Unmanned Aerial Vehicle Technology, Northwestern Polytechnical University, Xi'an 710072, China; jyan@nwpu.edu.cn
[3] Integrated Research and Development Platform of Unmanned Aerial Vehicle Technology, Xi'an 710072, China
[4] Research Center for Unmanned System Strategy Development, Northwestern Polytechnical University, Xi'an 710072, China
[5] Beijing Institute of Tracking and Telecommunication Technology, Beijing 100094, China; wxm_506@163.com
[6] Science and Technology on Complex System Control and Intelligent Agent Cooperative Laboratory, Beijing 100074, China; judysun07@163.com
* Correspondence: jsguoh@nwpu.edu.cn; Tel.: +86-15353745647

Abstract: With the development and strengthening of interception measures, the traditional penetration methods of high-speed unmanned aerial vehicles (UAVs) are no longer able to meet the penetration requirements in diversified and complex combat scenarios. Due to the advancement of Artificial Intelligence technology in recent years, intelligent penetration methods have gradually become promising solutions. In this paper, a penetration strategy for high-speed UAVs based on improved Deep Reinforcement Learning (DRL) is proposed, in which Long Short-Term Memory (LSTM) networks are incorporated into a classical Soft Actor–Critic (SAC) algorithm. A three-dimensional (3D) planar engagement scenario of a high-speed UAV facing two interceptors with strong maneuverability is constructed. According to the proposed LSTM-SAC approach, the reward function is designed based on the criteria for successful penetration, taking into account energy and flight range constraints. Then, an intelligent penetration strategy is obtained by extensive training, which utilizes the motion states of both sides to make decisions and generate the penetration overload commands for the high-speed UAV. The simulation results show that compared with the classical SAC algorithm, the proposed algorithm has a training efficiency improvement of 75.56% training episode reduction. Meanwhile, the LSTM-SAC approach achieves a successful penetration rate of more than 90% in hypothetical complex scenarios, with a 40% average increase compared with the conventional programmed penetration methods.

Keywords: memory-based DRL; LSTM-SAC; penetration method; high-speed UAV; intelligent decision making

check for updates

Citation: Zhang, X.; Guo, H.; Yan, T.; Wang, X.; Sun, W.; Fu, W.; Yan, J. Penetration Strategy for High-Speed Unmanned Aerial Vehicles: A Memory-Based Deep Reinforcement Learning Approach. *Drones* **2024**, *8*, 275. https://doi.org/10.3390/drones8070275

Academic Editor: Yushu Yu

Received: 25 March 2024
Revised: 6 June 2024
Accepted: 8 June 2024
Published: 21 June 2024

Copyright: © 2024 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 great development of hypersonic-related technologies, High Supersonic Unmanned Vehicles (HSUAVs) have become a major threat in future wars. It will stimulate the continuous development and innovation of defending systems against HSUAVs. Correspondingly, the interception scenarios against HSUAVs are becoming complicated, and the capabilities of interceptors are upgrading dramatically. The traditional penetration strategies of HSUAVs are becoming increasingly inadequate for penetration tasks in complex interception scenarios. Consequently, the penetration strategy of high-speed aircraft is gradually being regarded as a significant and challenging research topic. Inspiringly, the Artificial-Intelligence-based penetration methods have become promising candidate solutions.

In early research, the penetration strategies with programmed maneuver prelaunch were designed, which had been extensively applied in engineering, such as the sinusoidal maneuver [1], the spiral maneuver [2,3], the jump–dive maneuver [4], the *S* maneuver [5], the weaving maneuver [6,7], etc., but their penetration efficiency was very limited. In order to improve the penetration effectiveness, the penetration guidance laws based on modern control theory have been widely studied. The authors in [8] presented optimal-control-based evasion and pursuit strategies, which can effectively improve survival probability under maneuverability constraints in one-on-one scenarios. The differential game penetration guidance strategy was derived in [9], and the neural networks were introduced in [10] to optimize the strategy to solve the pursuit–evasion problem. The improved differential game guidance laws were derived in [11,12], and the desired simulation results were obtained. In [13,14], a guidance law in a more sophisticated target–missile–defender (TMD) scenario was derived with control saturation, which could ensure the missile escaped from the defender. These solutions are mainly performed based on the linearized models, resulting in a loss of accuracy.

The inadequacy of existing strategies' capabilities and the significant superiority of Artificial Intelligence motivate us to solve the problem of penetration in complex scenarios by intelligent algorithms. As far as the authors are concerned, in most of the research on penetration strategies based on modern control theory, the interception scenario was simplified to a two-dimensional (2D) plane, wherein only one interceptor was considered. However, in practical situations, the opponent usually launches at least two interceptors to deal with high-speed targets. Therefore, the performance of these penetration strategies would degrade distinctly in practical applications. In recent years, intelligent algorithms have been widely developed due to their excellent adaptability and learning ability [15]. Deep Reinforcement Learning (DRL) was proposed by DeepMind in 2013 [16]. DRL combines the advantages of Deep Learning (DL) and Reinforcement Learning (RL), with powerful decision making and situational awareness capabilities [17]. The designs of guidance laws based on DRL were proposed in [18–20], and it has been proven that the new methods are superior to the traditional proportional navigation guidance (PNG). DRL has been applied in much research, such as collision avoidance methods for Urban Air Mobility (UAM) vehicles [21] and Air-to-Air combat [22]. In the field of high-speed aircraft penetration or interception, a pursuit–evasion game algorithm was designed by PID control and Deep Deterministic Policy Gradient (DDPG) algorithm in [23], and it achieved better results than only PID control. Intelligent maneuver strategies using DRL algorithms have been proposed in [24,25] to solve the problem of one-to-one midcourse penetration of an aircraft, which can achieve higher penetration win rates than traditional methods. In [26,27], the traditional deep Q-network (DQN) was improved into a dueling double deep Q-network (D3Q) and a double deep Q-network (DDQN) to solve the problem of attack–defense games between aircraft, respectively. In [28], the GAIL-PPO method that combined the Proximal Policy Optimization (PPO) and imitation learning was proposed. Compared with classic DRL, the improved algorithm provided new approaches for penetration methods.

Based on a deep understanding of the penetration process, high-speed UAVs can only make correct decisions by predicting the intention of interceptors. Therefore, algorithms are required to be able to process state information over a period of time, which is not available in the current DRL due to the sampling method of the replay buffer. To attack this problem, a Recurrent Neural Network (RNN) that specializes in processing time series data is incorporated into the classic DRL. The Long Short-Term Memory (LSTM) networks [29] are mostly used as memory modules because of their stable and efficient performance and excellent memory capabilities. The classic DRL algorithms were extended by LSTM networks in [30,31], which demonstrated superior performance than classic DRL in the problems of Partially Observable MDPs (POMDPs). The conspicuous advantages of memory modules in POMDPs were demonstrated. The LSTM-DDPG (Deep Deterministic Policy Gradient) approach has been proposed to solve the problem of sensory data collection

of UAVs in [32], and the numerical results show that LSTM-DDPG could reduce packet loss more than classic DDPG.

The above research demonstrates the advantages of the DRL algorithm with memory modules, but it has not yet been applied to the penetration strategy of high-speed UAVs. The Soft Actor–Critic (SAC) algorithm [33] was proposed in 2018 and has been applied with the advantages of wide exploration capabilities. Therefore, the LSTM networks are incorporated into the SAC to explore the application of LSTM-SAC in the penetration scenario of a high-speed UAV escaping from two interceptors. The main contributions in this paper can be summarized as follows:

(a) A more complex 3D space engagement scenario is constructed, where the UAV faces two interceptors with a surrounding case and its penetration difficulty dramatically increases.
(b) A penetration strategy based on DRL is proposed, in which a reward function is designed to enable high-speed aircraft to evade interceptors with low energy consumption and minimum deviation, resulting in a more stable effect than conventional strategies.
(c) A novel memory-based DRL approach, LSTM-SAC, is developed by combining the LSTM network with the SAC algorithm, which can effectively make optimal decisions on input temporal data and significantly improve training efficiency compared with classic SAC.

The rest of the paper is organized as follows: The engagement scenario and basic assumptions are described in Section 2. The framework of LSTM-SAC is introduced in Section 3, and the state space, action space, and reward function are designed based on MDP and engagement scenarios. In Section 4, the algorithm is trained and validated. Finally, the simulation results are analyzed and summarized.

2. Engagement Scenario

The combat scenario modeling, kinematic, and dynamic analysis are formulated in this chapter. To simplify the scenario, some basic assumptions are made as follows:

Assumption 1: Both the high-speed aircraft and interceptors are described by point-mass models. The 3-DOF particle model is established in the ground coordinate system:

$$\begin{cases} \frac{dV}{dt} = g(n_x - \sin\theta) \\ \frac{d\theta}{dt} = \frac{g}{V}(n_y - \cos\theta) \\ \frac{d\varphi}{dt} = -\frac{g}{V\cos\theta} n_z \end{cases} \quad \begin{cases} \frac{dx}{dt} = V\cos\theta\cos\varphi \\ \frac{dy}{dt} = V\sin\theta \\ \frac{dz}{dt} = -V\cos\theta\sin\varphi \end{cases} \tag{1}$$

where x, y, z are the coordinates of the vehicle. V, θ, φ indicate the vehicle's velocity, flight path angle, and flight path azimuth angle, and g represents gravitational acceleration. n_x, n_y, n_z denote the projection of the overload vector on each axis of the direction of velocity. Specifically, overload is the ratio of the force exerted on the aircraft to its own weight, which is a dimensionless variable used to describe the acceleration state and the maneuverability of the aircraft.

$$n_i = \frac{N_i}{G}, i = x, y, z \tag{2}$$

where N_i represents the projection of the external force on the i-axis in the direction of velocity, and G represents the gravity of the aircraft. n_x is tangential overload, which indicates the ability of the aircraft to change the magnitude of its velocity, and n_y, n_z are normal overloads, which represent the ability of the aircraft to change the direction of its flight in the plumb plane and the horizontal plane, respectively.

Assumption 2: The high-speed UAV cruises at a constant velocity toward the ultimate attack target when encountering two interceptors. The enemy is able to detect the high-speed UAV at a range of 400 km and launches two interceptors from two different

launch positions against one target. The onboard radar of the aircraft starts to work at a range of 50 km relative to the interceptor and then detects the position information of the interceptor missile.

Assumption 3: The enemy is capable of recognizing the ultimate attack target of the high-speed aircraft. The guidance law of the interceptors does not switch during the entire interception procession. The interceptors adopt proportional navigation guidance (PNG) law with varying navigation gain:

$$\begin{aligned} a_I^C &= -N \cdot \dot{r}_{HI} \cdot \dot{q}_{HI} \\ N &= \begin{cases} 4 \; r_{HI} > 15km \\ 6 \; r_{HI} \leq 15km \end{cases} \end{aligned} \tag{3}$$

where a_I^C denotes the acceleration command of the interceptor. r_{HI}, q_{HI} denote the relative distance and line of sight (LOS) between the interceptor and the high-speed aircraft, respectively.

The above assumptions are widely used in the design of maneuvering strategies of aircraft, which can simplify the calculation process while providing accurate approximations.

In 3D planar, it is difficult to directly obtain the analytical relationship between LOS and fight-path angle. The motion of an aircraft in three-dimensional space can be simplified as a combination of horizontal and plumb planes, which can help us analyze and understand the motion characteristics of the aircraft more easily. Therefore, for the convenience of engineering applications, the motion of the 3D planar is projected onto two 2D planes (horizontal and plumb). Due to the two interceptors surrounding the horizontal plane instead of the plumb plane to strike high-speed aircraft, it is more difficult to escape on the horizontal plane. Maneuvers on the horizontal plane should be primarily considered. In addition, limited by engine technology conditions, maneuvering is primarily conducted on the horizontal plane.

Taking the horizontal plane as an example, the engagement between two interceptors and one high-speed aircraft is considered. As in Figure 1, X-O-Z is a Cartesian inertial reference frame. The notations I_1, I_2 and H are two interceptors and an aircraft, respectively, and the variables with subscripts I_1, I_2 and H represent the variables of the two interceptors and the high-speed aircraft. The velocity and acceleration are denoted by V, a, respectively. The notations of q, φ, ϕ are LOS angle, fight-path azimuth angle, and lead angle. The notation r is the relative distance between the interceptor and the high-speed aircraft.

Taking the interceptor-I as an example, the relative motion equations of the interceptor-I and the high-speed aircraft are:

$$\begin{cases} \dot{r}_{HI_1} = V_{rHI_1} = -V_H \cos \phi_H + V_{I_1} \cos \phi_{I_1} \\ \dot{q}_{HI_1} = \left(V_H \sin \phi_H - V_{I_1} \sin \phi_{I_1} \right) / r_{HI_1} \\ \phi_H = \varphi_H - q_1 \\ \phi_{I_1} = \varphi_{I_1} - q_1 \\ \dot{\varphi}_H = a_H / V_H \\ \dot{\varphi}_{I_1} = -a_{I_1} / V_{I_1} \end{cases} \tag{4}$$

To make the simulation model more realistic, the characteristics of an onboard autopilot for both interceptors and high-speed aircraft are incorporated. The first-order lateral maneuver dynamic is assumed as follows:

$$\dot{n} = (n^C - n) / \tau_T \tag{5}$$

where τ_T is the time constant of the target dynamics, and n^C denotes the overload command.

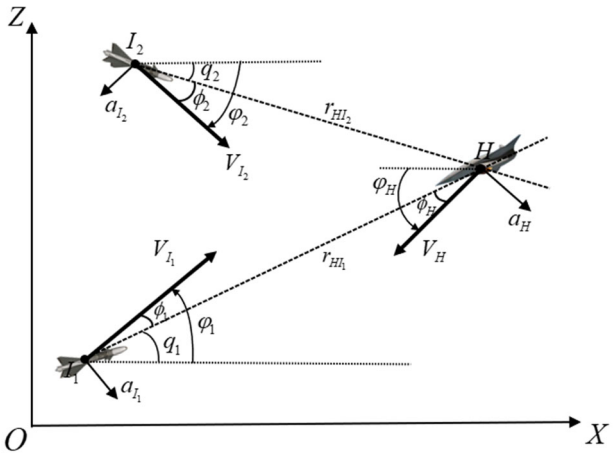


Figure 1. Engagement geometry.

3. Design of Penetration Strategy Based on Improved SAC

3.1. Design of MDP

3.1.1. Model of MDP

The optimal decision model is obtained by interacting with an environment without data labels using the DRL algorithm. As in Figure 2, the agent obtains the state information from the environment at each step, computes to obtain the action, and outputs it back to the environment. After obtaining the data, the agent updates its strategy by learning to maximize the total reward value. The Markov Decision Process formalizes the environment of RL [34], which is a memoryless random process.

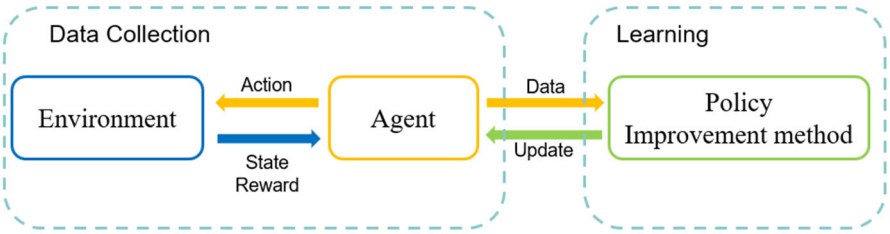


Figure 2. Schematic diagram of reinforcement learning.

MDP consists of 5 elements $[s, A, P, R, \gamma]$, where s is a finite set of states, A is a finite set of actions, P is a state transition probability matrix, R is a reward function used to score the decisions of the agent, and γ is a discount factor, and $\gamma \in [0, 1]$. The total return over the complete MDP process is obtained by weighted accumulation of r with γ. The training of DRL is the process that maximizes the total return:

$$U_t = R_t + \gamma \cdot R_{t+1} + \gamma^2 \cdot R_{t+1} + \ldots + \gamma^{n-t} \cdot R_n \tag{6}$$

The learning process of MDP is to maximize the rewards by learning strategies during the interaction with the environment.

3.1.2. Design of State Space and Action Space

During the process of penetration, the state and action of the high-speed aircraft are continuous. The construction of state space is based on the correct analysis of combat scenarios and missions, and the state space should be able to provide the complete required

information during the penetration process. The high-speed aircraft's own state information and relative motion information are selected as the state variables:

$$State = \left\{ ConsumeFuel, \Delta_x^1, \Delta_y^1, \Delta_z^1, \ldots \right. \\ \left. \ldots \Delta_x^2, \Delta_y^2, \Delta_z^2, \theta, \varphi, \Delta r \right\} \tag{7}$$

where *ConsumeFuel* denotes the aircraft's fuel usage, aiming at constraining the energy consumption of the aircraft during the penetration process. During the training process, it is represented as the overload n_H of the aircraft at each step. $\Delta_x^i, \Delta_y^i, \Delta_z^i$ denote the relative position vector between the high-speed aircraft and the i^{th} interceptor in x, y, z axis, respectively, which are computed by $\Delta_x^i = x_i - x_H, \Delta_y^i = y_i - y_H, \Delta_z^i = z_i - z_H$. They are used to indicate the motion state of the offense and defense. θ, φ denote the trajectory angle and inclination angle of the aircraft, respectively, which are set to describe the flight status of the aircraft. Δr denotes the distance between aircraft and the target.

The above state variables have different dimensions and scales, and being used directly, can lead to the learning process becoming unstable or even training not converging. The different unit features were manually scaled for dimensionless preprocessing before entering the DRL network. The processed state variables are all between -1 and 1, so that each state feature can be treated fairly by the algorithm. Normalization is beneficial for stabilizing gradients, improving training efficiency, and improving the generalization ability of the algorithm. The processing method is shown in Table 1.

Table 1. Scaling preprocessing.

Variable	Pretreatment
$ConsumeFuel$	$n_H / n_{\max}$
$\Delta_x^i, \Delta_y^i, \Delta_z^i$	$\Delta_x^i / \Delta_{x0}^i, \Delta_y^i / \Delta_{y0}^i, \Delta_z^i / \Delta_{z0}^i$
$q_\varepsilon^i, q_\beta^i$	$q_\varepsilon^i / 2\pi, q_\beta^i / 2\pi$
ε, β	$\varepsilon / 2\pi, \beta / 2\pi$
Δr	$\Delta r / \Delta r_0$

$n_{\max}$ denotes the maximum overload of the aircraft, and $\Delta_{x0}^i, \Delta_{y0}^i, \Delta_{z0}^i$ and Δr_0 denote the relative positions and relative distance at the initial moment, respectively.

The overload commands of the high-speed aircraft are selected as the action space of the agent. In this case, the intelligent agent makes maneuvering decisions completely autonomously throughout the entire process. We design the action space as follows:

$$Action = \left[n_y^C, n_z^C \right] \tag{8}$$

where n_y^C and n_z^C are the longitudinal overload and normal overload commands generated by the agent, which, respectively, control the flight direction of the aircraft on the plumb and horizontal planes. Since the UAV is assumed to fly at a constant velocity during the penetration process, n_x, which is used to control the magnitude of the velocity, it is not considered in the action state. It is worth noting that the action space directly outputted by the network is between -1 and 1, so it needs to be multiplied by the maximum overload $n_{\max}$ before outputting to the environment.

3.1.3. Design of Reward Function

The reward function maps the state information into enhancement signals, reflecting the understanding of the task logic. It is used to evaluate the quality of actions and determine whether the agent can learn the required strategy. The reward function is designed as follows:

$$reward = reward\_stage + reward\_end \tag{9}$$

where *reward_stage* is used to evaluate the effectiveness of actions taken by the agent in each simulation step. *reward_end* is used to evaluate whether the aircraft has completed the evasion of all interceptors. The accumulation of the two constitutes a reward function.

reward_stage can be specifically represented as:

$$\begin{aligned} &reward\_stage = reward_1 + reward_2 \\ &\begin{cases} reward_1 = -ConsumeFuel \times k_1 \\ reward_2 = -k_2 \times [r_{mt}(i+1) - r_{mt}(i)] \end{cases} \end{aligned} \tag{10}$$

where $reward_1$ represents the penalty for $ConsumeFuel$, which is used to constrain the energy consumption during the penetration process. $reward_2$ is used to guide the agent to continuously fly towards to the target direction, which is beneficial for constraining the agent to not deviate from the preset trajectory after successful penetration.

reward_end can be specifically represented as:

$$reward\_end = \begin{cases} k_3 \times n, \ r > r_d \\ -k_4, \ r \leq r_d \end{cases} \tag{11}$$

where r represents the miss distance of the aircraft, and r_d represents the killing radius of the interceptor. When the miss distance is larger than the killing radius, the penetration task is successful.

k_1, k_2, k_3, k_4 are the constants to shape the reward function. Their values are set based on the actual penetration mission and adjusted during the training process to obtain the optimal results.

3.2. Model of SAC

The SAC algorithm is based on the Actor–Critic (AC) framework. The AC framework contains two Deep Neural Networks (DNN), which are used to fit the Actor network and Critic network. The AC framework is shown in Figure 3, which separates the process of evaluating action value and updating strategies. The Actor network is a policy-based method responsible for making and updating decisions. The Critic network is a value-based method responsible for estimating the $V_\pi(s)$ and $Q_\pi(s_t, a_t)$ of the Actor network.

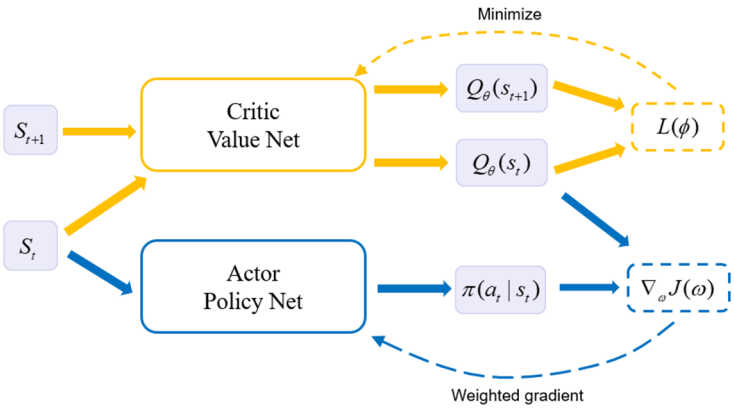


Figure 3. Actor–Critic framework.

SAC is an Off-Policy DRL algorithm based on the maximum entropy reinforcement learning framework. Entropy is a measure of uncertainty in probability distributions:

$$H(p) = -\int p(x) \log p(x) dx \tag{12}$$

The learning process of the SAC algorithm is to obtain a policy that maximizes both its cumulative reward and the entropy of each action:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \underset{\tau\sim\pi}{E}\left[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t,s_{t+1}) + \alpha H(\pi(\bullet|s_t)))\right] \tag{13}$$

where α is the entropy temperature coefficient that indicates the degree of randomness in the strategy. It can randomize the strategy. That is, the probability of each output action is as dispersed as possible.

The action-value function containing α is defined as follows:

$$Q_\pi(s_t,a_t) = \underset{\tau\sim\pi}{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1}) + \alpha\sum_{t=1}^{\infty}\gamma^t H(\pi(\bullet|s_t))\right] \tag{14}$$

Similarly, the state-value function is defined as follows:

$$V_\pi(s) = \underset{\tau\sim\pi}{E}[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t,s_{t+2}) + \alpha H(\pi(\bullet|s_t)))] \tag{15}$$

Therefore, the relationship between V_π and Q_π can be written as:

$$V_\pi(s) = \underset{\alpha\sim\pi}{E}[Q_\pi(s,a)] + \alpha H(\pi(\bullet|s)) \tag{16}$$

The SAC algorithm is applicable to both continuous and discrete action space simultaneously, so it is widely used in complex missions because of its excellent exploration ability and robustness. The complete summary of the SAC algorithm is shown in Algorithm 1.

Algorithm 1. SAC algorithm.

Initialize the value function and target value function with parameter vectors $\psi, \overline{\psi}$
Initialize the soft q-function and policy network with parameter vectors θ, ϕ
Initiate the experience buffer D
for each iteration **do**
  for each environment step **do**
    Generate the action from actor network based on the current state
    $a_t \sim \pi_\phi(a_t|s_t)$
    Applying the a_t to the environment yields the reward r_t and a new state
    $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
    Store the transition experience (s_t, a_t, r_t, s_{t+1}) in the experience buffer D
    $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$
  end for
  for each gradient **do**
    Randomly batch sample from experience buffer D
    Calculate the TD error and loss function $L(\phi, D), L(\alpha)$
    Update the value critic network $\psi \leftarrow \psi - \lambda_V\hat{\nabla}_\psi J_V(\psi)$
    Update the soft q-function $\theta_i \leftarrow \theta_i - \lambda_Q\hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for i $\in \{1, 2\}$
  Update the policy network $\phi \leftarrow \phi - \lambda_\pi\hat{\nabla}_\phi J_\pi(\phi)$
  Adjust the temperature $\alpha \leftarrow \alpha - \lambda\hat{\nabla}_\alpha J(\alpha)$
        Update the target value critic network $\overline{\psi} \leftarrow \tau\psi + (1-\tau)\overline{\psi}$
  end for
end for

The traditional SAC algorithm is based on the sampling method of reply buffer and fully connected layers, which can only fit the policy and value function based on the inputs at the current moment. However, the state space is closely related in time during the penetration process, and the decision instructions output by the algorithm also depend

on the states at the moments before and after. Therefore, the traditional SAC algorithm is considered to be improved in order to make accurate decisions on time series data.

3.3. Architecture of LSTM-SAC

The LSTM network is improved from RNN, which excels in handing times series data with long-term dependencies. LSTM can selectively store information through its ability to remember and forget, as shown in Figure 4. Each LSTM unit consists of an input gate, an output gate, and a forget gate.

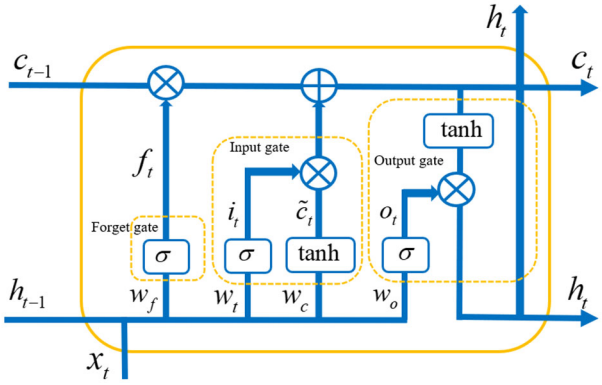


Figure 4. Framework of LSTM.

The calculation formula for LSTM is as follows:

$$\begin{aligned} f_t &= \sigma(w_f[h_{t-t}, x_t] + b_f) \\ i_t &= \sigma(w_i[h_{t-t}, x_t] + b_i) \\ \widetilde{c}_t &= \tanh(w_c[h_{t-t}, x_t] + b_c) \\ c_t &= f_t \times c_{t-1} + i_t \times \widetilde{c}_t \\ o_t &= \sigma(w_o[h_{t-t}, x_t] + b_o) \\ h_t &= o_t \times \tanh(c_t) \end{aligned} \tag{17}$$

where σ and tanh represent the sigmoid function and hyperbolic sine function, respectively. f_t represents the output of the forget gate, which measures the degree of forgetting about the output c_{t-1} from the previous unit. $\widetilde{c}_t$ represents the updated value of unit status, which is calculated from input data x_t and hidden node h_{t-1} through a layer of neural networks. i_t represents the output of the input gate, which is used to indicate valid information extracted from the current input $\widetilde{c}_t$. h_t is obtained from the output gate o_t and the cell state c_t, and the last unit of h_t is selected as the output of the network. w_f, w_i, w_c, w_o represent the weights of each part, respectively, and b_f, b_i, b_c, b_o, respectively, represent the bias of the forget gate, input gate, and output gate.

The traditional SAC algorithm is improved by incorporating the LSTM network. An AC framework with an LSTM layer and fully connected layer is created, in which the characterization layers of LSTM could learn beneficial information from the previous states and adjust the weights and bias to assist with decision making based on the SAC algorithm. Therefore, LSTM networks can predict future state information based on time series information before a certain time period t. The architecture of the LSTM-SAC is shown in Figure 5.

Due to the temporal processing characteristics of the LSTM network, the input of the algorithm is no longer the state values at a single simulation step but a column of time-continuous state data. Any length of historical data can be processed. In this paper, n consecutive states $s_{t-n}, s_{t-n+1} \ldots, s_{t-1}, s_t$ are selected as a set of inputs for LSTM-SAC after comparative experiments, which requires the replay buffer to implement random continuous batch sampling. Similarly, the actor network obtains actions based on n consecutive

states. The LSTM can extract effective features, make predictions based on continuous input states, and finally output them to the SAC. In this way, the agent has the ability to memorize, and the efficiency of the algorithm training is improved.

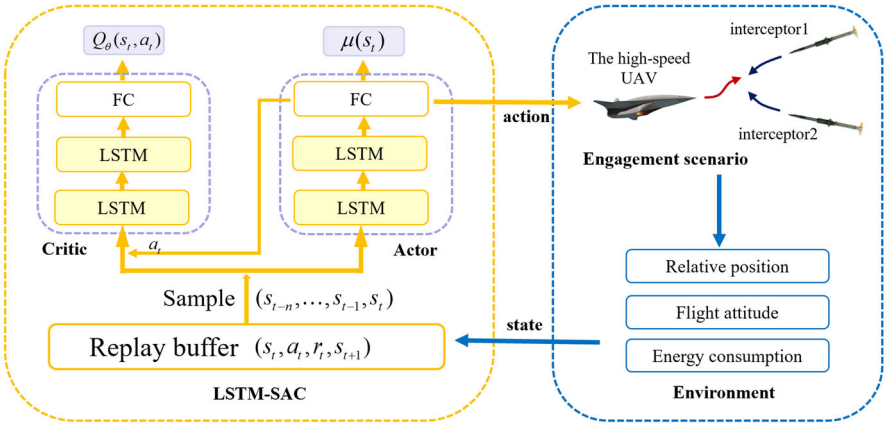


Figure 5. Architecture of the LSTM-SAC.

4. Experimentation and Verification

4.1. Simulation Parameter Settings

In this section, the LSTM-SAC agent is used to generate penetration strategy commands for a high-speed UVA. Only the head-on engagement scenarios are considered because of the speed limitation of interceptors.

The experiments were implemented in the Python3.11, PyTorch2.0.1, and cuda117 environments with simulation step size set to 0.01 s, and the fourth-order Runge–Kutta (RK4) method is used for ballistic calculations in the engagement scenario. The enemy's launch sites are distributed on both sides of the high-speed aircraft flight trajectory. Therefore, after the enemy detects our aircraft, two interceptors are launched, and a perimeter is formed to intercept the aircraft. The maneuverability of the interceptors is higher than that of the aircraft, which makes the escape of the aircraft more difficult.

To conform to real engagement scenarios and prevent overfitting, the specific launch locations of the interceptors are randomly generated in a circle with a radius of 3 km, centered on the enemy launch center site. That is, the initial states of the interceptors are different in each episode of training. The simulation parameter settings are shown in the Table 2.

Table 2. Simulation parameter settings.

Parameter	Value
Enemy launch position center site $(x,y,z)/\mathrm{km}$	(0,0,30), (0,0,−30)
High − speed aircraft initiate location $(x,y,z)/\mathrm{km}$	(400,25,0)
Target location $(x,y,z)/\mathrm{km}$	(−30,0,0)
Interceptor killing radius r_d/m	10
Interceptor maximum velocity $V_{I\_\max}/\mathrm{m}\cdot\mathrm{s}^{-1}$	1850
High − speed aircraft cruising velocity $V_H/\mathrm{m}\cdot\mathrm{s}^{-1}$	2000
Interceptor maximum overload $\lvert n_I\_\max\rvert$	100
High − speed aircraft maximum overload $\lvert n_H\_\max\rvert$	25
Autopilot time constant τ/s	0.2

The network parameter settings for LSTM-SAC are shown in the Table 3, with the *StateDim* representing the dimension of state space, and the *ActionDim* representing the dimension of action space. The actor network and critic network are represented by two-layer LSTM networks and a fully connected layer. The parameter n represents the length of the historical states entered into the algorithm. The rectified linear units (ReLU) function

is chosen as the activation function of the FC layers because it can solve the problem of overfitting and accelerate training efficiency. The ReLU function is defined as:

$$f(x) = \max(0, x) \tag{18}$$

Table 3. Network layer settings.

Layer	Actor Net	Critic Net
Input layer	$StateDim \times n$	$StateDim \times n + ActionDim$
LSTM-1 layer	256	256
LSTM-2 layer	128	128
FC layer	64	64
Output layer	$ActionDim$	1

The Tanh function is chosen as the activation function for the output layer of the actor network, which limits the output of the actor network to between -1 and 1. The Tanh function is defined as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{19}$$

The constant parameters used to set the reward function in the simulation scenario of LSTM-SAC training are shown in Table 4. Among the *reward_stage*, the energy limitation has the highest proportion, which aims to minimize the energy consumption during the penetration process without affecting the subsequent strike mission. In addition, the degree of deviation from the predetermined route during the escape process is reduced for the vehicle under energy limitation. Therefore, the range constraint k_2 has a much lower percentage of *reward_stage*.

Table 4. Constant parameters in reward function.

Constant Parameter	Value
k_1	0.7
k_2	0.1
k_3	200
k_4	500

The termination condition for a single training process is set to:

$$\frac{dr_{HI_i}}{dt} > 0 \rightarrow end \tag{20}$$

It can be concluded that the penetration process has ended when the relative distance begins to increase. The miss distance is used to determine the results of a single training process. If the miss distance is larger than the killing radius of the interceptor, the penetration is successful.

The selection of hyperparameters is important for the DRL training process. By continuously adjusting during the training process, the hyperparameters setting that is suitable for SAC-LSTM is obtained. The hyperparameter settings are shown in Table 5.

Table 5. Hyperparameter settings.

Hyperparameter	Value
StateDim	10
ActionDim	2
History states length n	1, 3, 5
Bidirectional of LSTM	False
Dropout of LSTM	0
Optimizer	Adam
Batch size	128
Experience buffer D	10,000
Noise variance	0.5
Actor learning rate *policy_lr*	1×10^{-3}
Critic learning rate *value_lr*	3×10^{-3}
Alpha learning rate *q_lr*	3×10^{-4}
Discounting factor, γ	0.98
Training episodes	1000

4.2. Analysis of Simulation Results

4.2.1. Training Results

The scenario of the penetration process of the high-speed aircraft is trained based on DRL algorithms. In order to verify the effectiveness of the LSTM network, the classical SAC algorithm is used for comparison with the LSTM-SAC approach. The fully connected network layers of the classic SAC algorithm are the same as that of LSTM, and the activation function is the ReLU function. States with different historical lengths ($l = 1 \sim 5$) are input into the LSTM-SAC to verify the ability of the LSTM networks to handle temporal dependencies. Meanwhile, the advanced Proximal Policy Optimization (PPO) algorithm is used to compare with the LSTM-SAC and SAC. The simulation parameters and hyperparameters settings are the same for all three.

The PPO algorithm was proposed in 2017 [35]. Based on the strategy gradient method and AC framework, the PPO algorithm can be efficiently applied to continuous state and action spaces. A new strategy update mechanism called 'Clipping' has been introduced into PPO, which limits the magnitude of policy updates. This avoids training instability caused by significant updates. The objective function of the PPO algorithm is

$$J_{PPO(\theta)} = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)] \tag{21}$$

where $r_t(\theta)$ is the ratio of the new strategy to the old strategy, and $\hat{A}_t$ is the advantage estimation function. $clip, \varepsilon$ are the truncation function and truncation constant, respectively, which limit the ratio of new and old strategies to $1-\varepsilon$ and $1+\varepsilon$ to ensure that the gap between the two strategies is not too large.

The variation in the average rewards is demonstrated as Figure 6. As all algorithms have converged, only the average rewards for the first 600 episodes are shown for clarity. The values of the reward function determine the degree to which the agent learns the ideal strategy, indicating that the agent is able to continuously learn and improve its strategy during the training process.

It can be seen that the PPO algorithm performs well in the first 200 episodes during training, but the reward function shows a gradual downward trend after convergence. This is because the algorithm eventually converges to an aggressive strategy: the agent attempts to escape with minimal energy consumption. The highest rewards will be received once the escape is successful, but this causes the vehicle to be easily intercepted. Therefore, the SAC and the LSTM-SAC perform better in this scenario.

The values in brackets in the legend represent the length of the input historical states, which is set with the values reference [31]. It can be seen that both the traditional SAC and the LSTM-SAC can converge quickly and make effective decisions for aircraft penetration strategies. In such an engagement scenario, LSTM-SAC with three historical states showed

the best performance, demonstrating the advantage of the memory component in complex scenarios. LSTM-SAC(4) has higher training efficiency in the first 100 rounds but lower average rewards in the later stages. This is because the penetration success rate of the agent is lower after training convergence. LSTM-SAC(5) takes second place, the reason considered is that due to the short simulation step, the three sliding windows are similar to the five sliding windows. However, the inputs of the five history states increase the difficulty of the algorithm to converge and the computation resource. LSTM-SAC with longer history states should work better if it is used in more complex engagement scenarios or Partially Observable Markov Decision Processes (POMDPs).

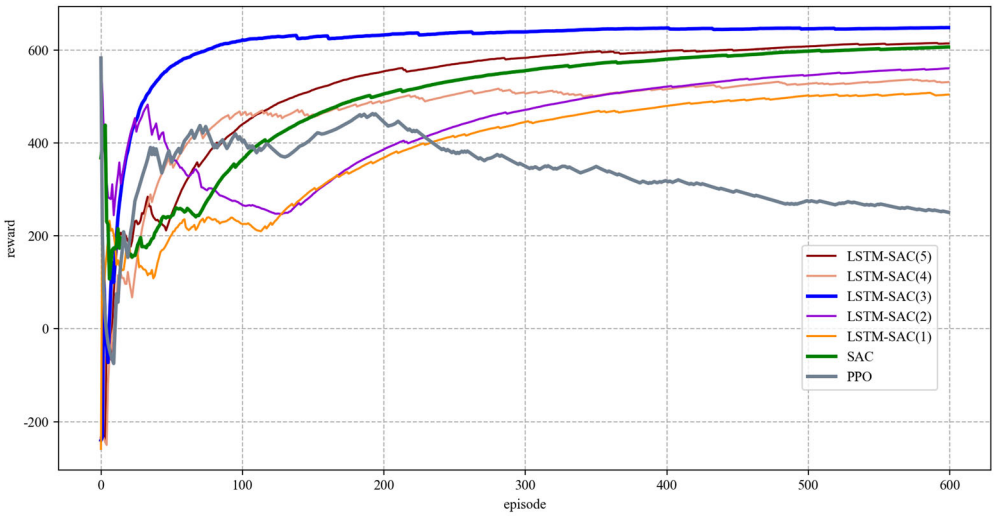


Figure 6. Average reward in the training process.

It is worth noting that the LSTM-SAC(1) with only current state inputs and the LSTM-SAC (2) with only one historical state perform lower than the classical SAC with the same inputs. It is indicated that if the input data do not have a time-dependent relationship, the LSTM networks cannot extract effective information from them, which makes the performance of the memory-based DRL worse.

In summary, the LSTM-SAC(3) with an effective time series can explore the optimal strategy faster and converge to a higher reward value compared with the classic SAC. The training efficiency of LSTM-SAC(3) is improved by about 75.56% over the SAC, indicating that the improved algorithm has significant improvements in reducing training times, saving resources and time. To simplify the description, the LSTM-SAC in the following text refers to LSTM-SAC (3) with three historical state inputs.

For the LSTM-SAC approach, the average rewards in the first 100 episodes are low and unstable, which means the agent has not learned the ideal penetration strategy and the algorithm has not converged to the optimum. In this case, the high-speed aircraft maneuvering decisions are highly stochastic, resulting in easy interception and high energy consumption. As the training progressed, the agent gradually learned the penetration strategy we expected. In the later episodes of training, the reward value was maintained at a high level. The agent learned the ideal penetration strategy is to avoid collisions with lower energy. However, low energy consumption results in vehicles that are easily intercepted; the agent needs to maintain a balance between energy consumption and win rate. Due to the agent still exploring the optimal strategy and random inferiority initial states, it is extremely difficult to achieve 100% evasion of interceptors.

4.2.2. Verification Results

In order to verify the effectiveness of the memory-based DRL, locally generated files of intelligent decision networks based on LSTM-SAC, SAC, and PPO are saved after training convergence. Sinusoidal maneuvers with fixed or random parameters, square maneuvers,

and no maneuver scenarios are used to compare with the DRL maneuver. The same simulation scenario and parameters are used in the verification.

The overload commands of sinusoidal maneuvers are written in the body coordinate system as

$$N_{C\_\sin} = \begin{bmatrix} 0 \\ A_1 \times \sin(\frac{2\pi}{T_1} \cdot t + \omega_1) + K_1 \\ A_2 \times \sin(\frac{2\pi}{T_2} \cdot t + \omega_2) + K_2 \end{bmatrix} \tag{22}$$

where A, T, ω are the amplitude, period, and phase of the sinusoidal maneuver command function. K denotes the bias, which is reflected as the translation of the function.

The overload commands of square maneuver can be written in the body coordinate system as:

$$N_{c\_square} = \begin{bmatrix} 0 \\ \sum_{n=-\infty}^{+\infty} (A_3(t - n \cdot T_3 + \frac{1}{2}) - A_3(t - n \cdot T_3 - \frac{1}{2})) \\ \sum_{n=-\infty}^{+\infty} (A_4(t - n \cdot T_4 + \frac{1}{2}) - A_4(t - n \cdot T_4 - \frac{1}{2})) \end{bmatrix} \tag{23}$$

where A_3, A_4 are the amplitude constants, and T_3, T_4 is the period of the square maneuver.

Different parameters of the sinusoidal maneuver have different effects on the penetration results. Therefore, two forms of sinusoidal maneuver instructions with fixed and random parameters are considered. The sinusoidal maneuver with random parameters is instructed to randomly generate parameters in a fixed range at fixed intervals. After simulation verification, the optimal parameter settings of sinusoidal maneuvers and square maneuvers are shown in Table 6.

Table 6. Parameter settings in maneuver commands.

Maneuver Methods	Parameter	Value (Range)	Parameter	Value (Range)
Sinusoidal maneuver with fixed parameters	A_1	5	A_2	24
	T_1	10	T_2	8
	ω_1	0	ω_2	$\pi/4$
Sinusoidal maneuver with random parameters	A_1	$[5, 10]$	A_2	$[15, 25]$
	T_1	$[3, 10]$	T_2	$[3, 10]$
	ω_1	$[0, \pi/2]$	ω_2	$[0, \pi/2]$
Square maneuver	A_3	5	A_4	24
	T_3	5	T_4	5

The above maneuver methods are tested with the same simulation scenario 1000 times, and the test results are recorded. The launch positions of the interceptors are randomly generated in launch sites with a radius of 5 km to verify the robustness of the penetration strategies.

The win rate statistics are shown in Table 7. Excluding the case of no maneuver, the overload response obtained from the 10th test and the corresponding trajectory graph are recorded as shown in Figures 7 and 8, respectively. The corresponding miss distance values are recorded in Table 7. If the miss distance is less than 10 m, it indicates interception.

When not maneuvering, the aircraft still has the opportunity to evade with its speed advantage. The addition of maneuver strategies all results in higher escape success rates. For the DRL maneuver strategy, LSTM-SAC and SAC maneuvers have the highest escaping probabilities and are not affected by the different initial flight states of the interceptors. The LSTM-SAC maneuver is considered to possess strong robustness. However, the PPO maneuver has a relatively low win rate, which is related to the aggressive strategy of the

PPO algorithm. Among procedural maneuvers, the square maneuver has the highest win rate, followed by the sinusoidal maneuver.

Table 7. Win rate and miss distance statistics.

Maneuver Method	Win Rate	Miss Distance In 10th Test
LSTM-SAC maneuver	94.3%	32.82 m, 903.75 m
SAC maneuver	92.6%	11.86 m, 1186.13 m
PPO maneuver	25.0%	12.64 m, 12.22 m
Square maneuver	82.8%	23.64 m, 5285.42 m
Sinusoidal maneuver with random parameters	43.4%	14.21 m, 15.06 m
Sinusoidal maneuver with phase $\pi/4$	18.1%	1.26 m, 876.20 m
No maneuver	8.4%	0.11 m, 1684.01 m

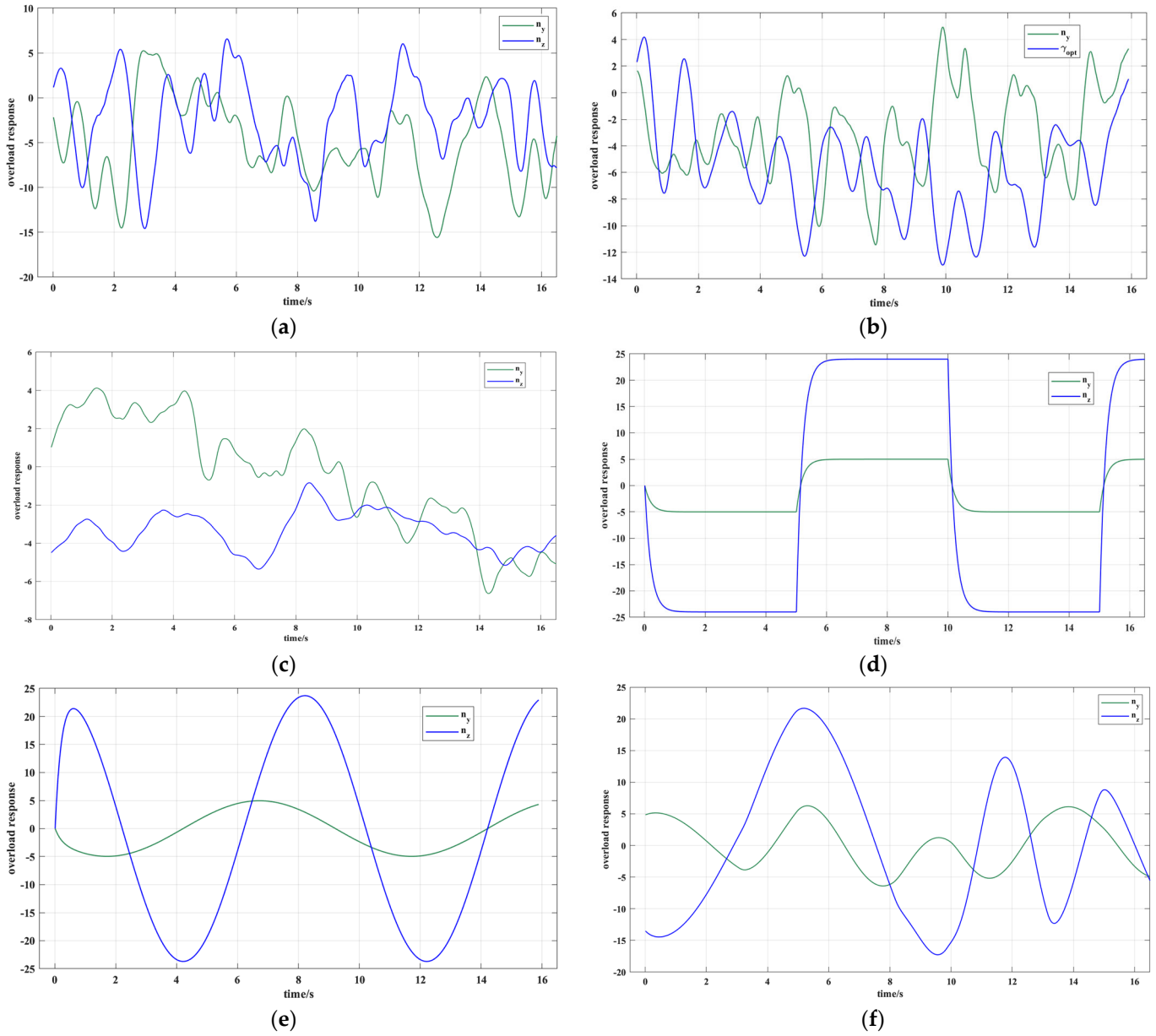
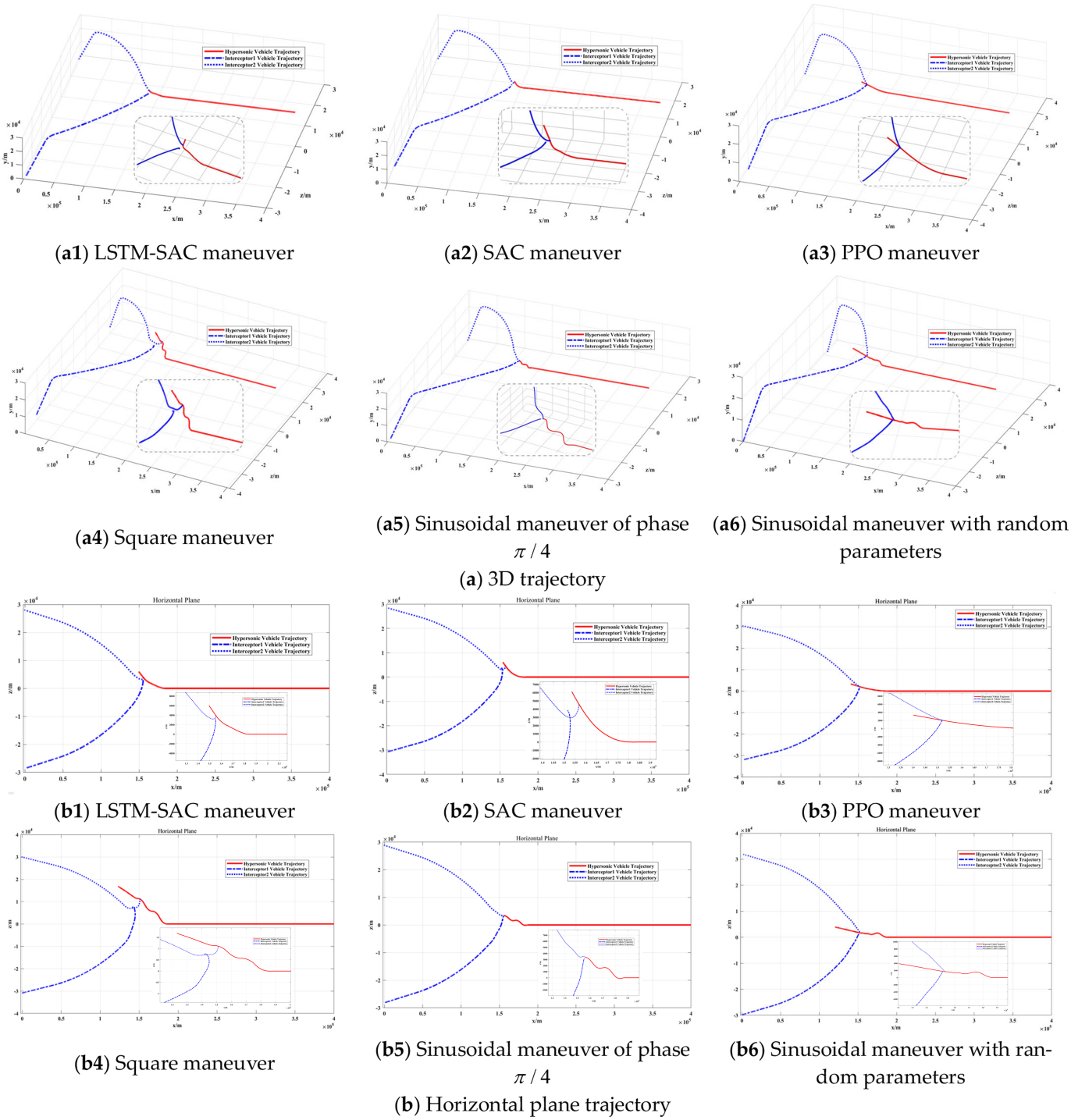


Figure 7. Overload commands response: (**a**) LSTM-SAC maneuver, (**b**) SAC maneuver, (**c**) PPO maneuver, (**d**) square maneuver, (**e**) sinusoidal maneuver of phase $\pi/4$, (**f**) sinusoidal maneuver with random parameters.



(**a1**) LSTM-SAC maneuver (**a2**) SAC maneuver (**a3**) PPO maneuver

(**a4**) Square maneuver (**a5**) Sinusoidal maneuver of phase $\pi / 4$ (**a6**) Sinusoidal maneuver with random parameters

(**a**) 3D trajectory

(**b1**) LSTM-SAC maneuver (**b2**) SAC maneuver (**b3**) PPO maneuver

(**b4**) Square maneuver (**b5**) Sinusoidal maneuver of phase $\pi / 4$ (**b6**) Sinusoidal maneuver with random parameters

(**b**) Horizontal plane trajectory

Figure 8. Verification results for all maneuver strategies.

According to the overload command response situation, it can be seen that the DRL maneuver commands change in real time based on the state information. The overload responses generated by the LSTM-SAC and SAC maneuvers are roughly the same, indicating that both LSTM-SAC and SAC can converge to the optimal solution. Compared with LSTM-SAC, the overload command of SAC changed more frequently and had a larger amplitude, making it more difficult for the onboard control system to track the command. Due to the 'Clipping' strategy update mechanism, the PPO maneuver has a smoother and smaller overload variation. This saves fuel for higher average rewards but can easily lead to penetration failure.

The commands for programmed maneuvers are input prelaunch and cannot be changed during flight to suit the actual situation. The sinusoidal maneuver with fixed parameters has the most regular change in overload, making it easy for enemies to predict its trajectory. Therefore, the win rate of this method is the lowest. Sinusoidal maneuvers with random parameters are harder to predict in comparison and consume energy better than the former.

The trajectory plots show that the PPO maneuver has the least amount of deviation from the original trajectory during the penetration process. The maximum energy consumption of the square maneuver leads to a higher win rate of escaping, but this makes it more difficult to return to the predetermined trajectory after the penetration process. The LSTM-SAC maneuver is able to maintain the win rate while deviating from the original trajectory and consuming the least amount of energy. It can be considered that the penetration strategy of LSTM-SAC has achieved optimal results in this engagement scenario.

After the penetration process, the high-speed aircraft switches back to the horizontal flight and prepares for guiding back to the preset trajectory.

5. Conclusions

In this paper, the penetration strategy of a high-speed UAV is transformed into decision-making issues based on MDP. A memory-based SAC algorithm is applied in decision algorithms, in which LSTM networks are used to replace the fully connected networks in the SAC framework. The LSTM networks can learn from previous states, which enables the agent to deal with decision-making problems in complex scenarios. The architecture of the LSTM-SAC approach is described, and the effectiveness of the memory components is verified by mathematical simulations.

The reward function based on the motion states of both sides is designed to encourage the aircraft to intelligently escape from the interceptors under energy and range constraints. By the LSTM-SAC approach, the agent is able to continuously learn and improve its strategy. The simulation results demonstrate that the converged LSTM-SAC has similar penetration performance to the SAC, but the LSTM-SAC agent with three historical states has a training efficiency improvement with 75.56% training episodes reduction compared with the classical SAC algorithm. In the engagement scenarios wherein two interceptors with strong maneuverability and random initial flight states within a certain range are considered, the probability of successful evasion of the high-speed aircraft is higher than 90%. Compared with the conventional programmed maneuver strategies represented by the sinusoidal maneuver and the square maneuver, the LSTM-SAC maneuver strategy possesses a win rate of over 90% and more robust performance.

In the future, noise and unobservable measurements are about to be introduced to the engagement scenarios, and the effect of LSTM-SAC will be evaluated in more realistic engagement scenarios. Additionally, the 3-DOF dynamic model is based on simplified aerodynamic effects. Due to the complex aerodynamic effects in the real battlefield, there is no guarantee that the strategies trained on the point-mass model will perform as well in real scenarios as they do in the simulation environment. In subsequent research, the 6-DOF dynamics model, which can better simulate complex maneuvers, will be considered for implementing penetration strategy research. In summary, the LSTM-SAC can effectively improve the training efficiency of classic DRL algorithms, providing a new promising method for the penetration strategy of high-speed UAVs in complex scenarios.

Author Contributions: Conceptualization, W.F. and J.Y.; methodology, H.G. and T.Y.; software, X.Z.; validation, X.Z., H.G. and T.Y.; formal analysis, X.Z., H.G. and T.Y.; investigation, W.S. and X.W.; resources, W.S. and X.W.; data curation, X.Z. and H.G.; writing—original draft preparation, X.Z.; writing—review and editing, H.G.; visualization, X.Z.; supervision, W.F. and J.Y.; project administration, W.S. and X.W.; funding acquisition, W.S. and X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (62003264).

Data Availability Statement: All data used during the study appear in the submitted article.

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

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

References

1. Kedarisetty, S.; Shima, T. Sinusoidal Guidance. *J. Guid. Control. Dyn.* **2024**, *47*, 417–432. [CrossRef]
2. Rusnak, I.; Peled-Eitan, L. Guidance law against spiraling target. *J. Guid. Control. Dyn.* **2016**, *39*, 1694–1696. [CrossRef]
3. Ma, L. The moedling and simulation of antiship missile terminal maneuver penetration ability. In Proceedings of the 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 2622–2626.
4. Yang, Y. *Static Instability Flying Wing UAV Maneuver Flight Control Technology Research*; Nanjing University of Aeronautics & Astronautics: Nanjing, China, 2015.
5. Huo, J.; Meng, T.; Jin, Z. Adaptive attitude control using neural network observer disturbance compensation technique. In Proceedings of the 9th International Conference on Recent Advances in Space Technologies (RAST), Istanbul, Turkey, 11–14 June 2019; pp. 697–701.
6. Zarchan, P. Proportional navigation and weaving targets. *J. Guid. Control. Dyn.* **1995**, *18*, 969–974. [CrossRef]
7. Lee, H.-I.; Shin, H.-S.; Tsourdos, A. Weaving guidance for missile observability enhancement. *IFAC-Pap.* **2017**, *50*, 15197–15202. [CrossRef]
8. Shima, T. Optimal cooperative pursuit and evasion strategies against a homing missile. *J. Guid. Control. Dyn.* **2011**, *34*, 414–425. [CrossRef]
9. Singh, S.K.; Reddy, P.V. Dynamic network analysis of a target defense differential game with limited observations. *IEEE Trans. Control. Netw. Syst.* **2022**, *10*, 308–320. [CrossRef]
10. Wu, Q.; Li, B.; Li, J. Solution of infinite time domain spacecraft pursuit strategy based on deep neural network. *Aerosp. Control.* **2019**, *37*, 13–18.
11. Bardhan, R.; Ghose, D. Nonlinear differential games-based impact-angle-constrained guidance law. *J. Guid. Control. Dyn.* **2015**, *38*, 384–402. [CrossRef]
12. Ben-Asher, J.Z. Linear quadratic pursuit-evasion games with terminal velocity constraints. *J. Guid. Control. Dyn.* **1996**, *19*, 499–501. [CrossRef]
13. Hang, G.; Wenxing, F.; Bin, F.; Kang, C.; Jie, Y. Smart homing guidance strategy with control saturation against a cooperative target-defender team. *J. Syst. Eng. Electron.* **2019**, *30*, 366–383.
14. Rusnak, I. The lady, the bandits and the body guards—A two team dynamic game. *IFAC Proc. Vol.* **2005**, *38*, 441–446. [CrossRef]
15. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst.* **1999**, *12*.
16. 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.
17. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
18. Jung, B.; Kim, K.-S.; Kim, Y. Guidance law for evasive aircraft maneuvers using artificial intelligence. In Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Austin, TX, USA, 11–14 August 2003; p. 5552.
19. Gaudet, B.; Furfaro, R. Missile homing-phase guidance law design using reinforcement learning. In Proceedings of the AIAA Guidance, Navigation, And Control Conference, Minneapolis, MN, USA, 13–16 August 2012; p. 4470.
20. He, S.; Shin, H.-S.; Tsourdos, A. Computational missile guidance: A deep reinforcement learning approach. *J. Aerosp. Inf. Syst.* **2021**, *18*, 571–582. [CrossRef]
21. Bacon, B.J. Collision Avoidance Approach Using Deep Reinforcement Learning. In Proceedings of the AIAA SciTech Forum, San Diego, CA, USA, 7 January 2022.
22. Pope, A.P.; Ide, J.S.; Mićović, D.; Diaz, H.; Rosenbluth, D.; Ritholtz, L.; Twedt, J.C.; Walker, T.T.; Alcedo, K.; Javorsek, D. Hierarchical reinforcement learning for air-to-air combat. In Proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 15–18 June 2021; pp. 275–284.
23. Tan, L.; Gong, Q.; Wang, H. Pursuit-evasion game algorithm based on deep reinforcement learning. *Aerosp. Control.* **2018**, *36*, 3–8.
24. Yameng, Z.; Hairui, Z.; Guofeng, Z.; Zhuo, L.; Rui, L. A design method of maneuvering game guidance law based on deep reinforcement learning. *Aerosp. Control.* **2023**, *40*, 28–36.
25. Gao, A.; Dong, Z.; Ye, H.; Song, J.; Guo, Q. Loitering munition penetration control decision based on deep reinforcement learning. *Acta Armamentarii* **2021**, *42*, 1101.

26. Nan, Y.; Jiang, L. Midcourse penetration and control of ballistic missile based on deep reinforcement learning. *Command. Inf. Syst. Technol.* **2020**, *11*, 1–9.
27. Xiangyuan, H.; Jun, C.; Hao, G.; Zhuoyang, Y.; Bo, T. Attack-Defense Game based on Deep Reinforcement Learning for High Speed Vehicle. *Aerosp. Control.* **2022**, *40*, 76–83.
28. Wang, X.; Gu, K. A Penetration Strategy Combining Deep Reinforcement Learning and Imitation Learning. *J. Astronaut.* **2023**, *44*, 914.
29. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
30. Heess, N.; Hunt, J.J.; Lillicrap, T.P.; Silver, D. Memory-based control with recurrent neural networks. *arXiv* **2015**, arXiv:1512.04455.
31. Meng, L.; Gorbet, R.; Kulić, D. Memory-based deep reinforcement learning for pomdps. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 5619–5626.
32. Li, K.; Ni, W.; Dressler, F. LSTM-characterized deep reinforcement learning for continuous flight control and resource allocation in UAV-assisted sensor network. *IEEE Internet Things J.* **2021**, *9*, 4179–4189. [CrossRef]
33. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
34. Van Otterlo, M.; Wiering, M. Reinforcement learning and markov decision processes. In *Reinforcement Learning: State-of-the-Art*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 3–42.
35. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.

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.