aerospace

MDPI

Article

Impact-Angle Constraint Guidance and Control Strategies Based on Deep Reinforcement Learning

Junfang Fan [1,2,*], Denghui Dou [3] and Yi Ji [1]

[1] School of Automation, Beijing Information Science & Technology University, Beijing 100192, China; yijiinbit@163.com
[2] Beijing Key Laboratory of High Dynamic Navigation Technology, Beijing Information Science & Technology University, Beijing 100192, China
[3] School of Aerospace Engineering, Beijing Institute of Technology, Beijing 100081, China; denghuidou@163.com
* Correspondence: wyhffjf@bistu.edu.cn

Abstract: In this study, two different impact-angle-constrained guidance and control strategies using deep reinforcement learning (DRL) are proposed. The proposed strategies are based on the dual-loop and integrated guidance and control types. To address comprehensive flying object dynamics and the control mechanism, a Markov decision process is used to solve the guidance and control problem, and a real-time impact-angle error in the state vector is used to improve the model applicability. In addition, a reasonable reward mechanism is designed based on the state component which reduces both the miss distance and the impact-angle error and solves the problem of sparse rewards in DRL. Further, to overcome the negative effects of unbounded distributions on bounded action spaces, a Beta distribution is used instead of a Gaussian distribution in the proximal policy optimization algorithm for policy sampling. The state initialization is then realized using a sampling method adjusted to engineering backgrounds, and the control strategy is adapted to a wide range of operational scenarios with different impact angles. Simulation and Monte Carlo experiments in various scenarios show that, compared with other methods mentioned in the experiment in this paper, the proposed DRL strategy has smaller impact-angle errors and miss distance, which demonstrates the method's effectiveness, applicability, and robustness.

Keywords: deep reinforcement learning; proximal policy optimization; guidance and control; impact-angle constraint

Citation: Fan, J.; Dou, D.; Ji, Y. Impact-Angle Constraint Guidance and Control Strategies Based on Deep Reinforcement Learning. *Aerospace* **2023**, *10*, 954. https://doi.org/10.3390/aerospace10110954

Academic Editors: Piotr Lichota and Katarzyna Strzelecka

Received: 9 October 2023
Revised: 9 November 2023
Accepted: 10 November 2023
Published: 13 November 2023

1. Introduction

With the increasing complexity of flight vehicle mission scenarios, it is often required that the flight vehicle can approach the target at the desired impact angle while accurately hitting the target to meet special mission requirements [1–3]. The conventional guidance law can no longer meet these mission requirements, so more and more researchers have begun to study the guidance law with angle constraints. Constrained guidance was first proposed by Kim et al. in 1973 [4], and, since then, various schemes have been proposed to meet different needs, including the biased proportional navigation (PN) schemes [5,6], sliding-mode variable structures [7,8], and optimal guidance laws [9,10]. However, sliding mode guidance laws can easily generate chattering, whereas the optimal guidance law often requires adopting certain assumptions, which can result in poor robustness. Therefore, it is urgent to develop an efficient method to address the current problems in terminal descent angle-constrained guidance and control.

In recent years, artificial intelligence technology has developed rapidly, and reinforcement learning (RL) technology, with autonomous decision-making ability, has always been regarded as a technological frontier in the field of artificial intelligence [11]. Deep reinforcement learning (DRL) combines the autonomous decision-making ability of RL with the feature extraction ability of deep learning. DRL has been demonstrated to show

outstanding performance in decision-making for high-dimensional data [12,13]. Furthermore, compared to the aforementioned traditional method, DRL has a smaller model dependency and higher transferability and is widely applicable to aerospace fields, such as spacecraft rendezvous and docking [14,15], quadcopter flight control [16,17], and drone collaboration [18]. Additionally, DRL has enabled the development of new approaches for designing flying object guidance and control systems.

In [19], the authors proposed a Q-learning algorithm that uses intelligent decision-making to establish an RL-based variable proportionality coefficient based on the PN guidance law. By appropriately designing the reward function, this method could effectively reduce horizontal and vertical separation. A guidance law designed using reinforcement meta-learning (RML), which employs only the observation variables of the line-of-sight (LOS) angle and its rate of change, was proposed in [20]. The RML optimization policy mapped the stable LOS angle and its derivative directly to the commanded thrust for the flying object's divert thrusters. This policy was proven to perform better than the augmented zero-effort misguidance policy when knowledge about a target's acceleration is not completely available. In [21], a reward-shaping pre-training method based on the proximal policy optimization (PPO) algorithm [22], which does not require an accurate model, was proposed. This method facilitated the fast training of intelligent agents that can be used in new tasks, thus significantly improving the training efficiency and trajectory performance. An end-to-end, model-free DRL guidance strategy based on the twin delayed deep deterministic policy gradient (TD3) algorithm was introduced in [23]. Recent studies have also used RL to solve problems in hit–defense games. In [24], the authors established the Hamilton–Jacobi–Bellman (HJB) equation under the game-theoretic settings and then introduced the extended disturbance observer (EDO) to estimate the target information. The optimal guidance command for the nonlinear system was approximated using online RL. Reference [25] proposed an anti-interception guidance method based on DRL and designed a new mechanism of repeated batch training to reduce the serious error estimation problem in the critic network. In [26], a solution that could address the uncertain no-fly zones faced by a high-speed flying object was introduced. This solution involved predictive correction guidance, pre-training the inclination guidance model based on supervised learning, and further upgrading the inclination guidance model based on RL. This approach could effectively fulfill the adaptability requirements of future intelligent decision-making systems in uncertain bypass scenarios. In [27], the PPO algorithm was improved and then proposed to introduce a new objective of minimum overload regularization optimization that directly inhibits the excessive movement caused by the guidance strategy. Reference [28] applied DRL algorithms to integrated guidance and control for three-dimensional, high-maneuverability flying object–target interception. By constructing a multi-factor reward function, the agent was trained based on the deep deterministic policy gradient (DDPG) algorithm to directly generate pitch and yaw fin commands to control the flying object to intercept the target. The aforementioned methods primarily analyzed the geometric relationship between the motion of a projectile, assuming the conditions of a small angle of attack (AOA) and constant velocity, without considering the effects of projectile dynamics and controllers. Furthermore, the primary constraint used in these methods is the miss distance, and there was no constraint on the impact angle.

In [29,30], a model-based RML approach was employed to train a deep neural network as a predictive model of guidance dynamics. The trained model was implemented into the model predictive path integral control framework, and a novel guidance law suitable for a variable speed interceptor, in a case where the actuator failed and the objective was to intercept a maneuvering target, which considered the impact angle constraint was introduced. In [31], a reasonable reward mechanism was designed to minimize the longitudinal angle error and align the LOS angle to the expected value. Also, a distributed exploration strategy was used to improve the exploration efficiency of the simulation environment during the model training process. In [32], a guidance method based on predictive correction considering the impact angle constraint was designed. Namely, deep

supervised learning-based networks were designed to predict the impact angle in real-time based on the mapping between the flight status and the guidance constraints. The PPO algorithm was subsequently used to learn the bias corrections for impact-angle errors based on the PN guidance law. In [33], based on the PN guidance law, the constraints on the impact angle and the field of view angle were realized by using the DRL phased learning bias command, and it was proved that the proposed method could be easily extended to the model with additional elements such as rotational dynamics. However, the above-mentioned methods were designed based on a simplified dynamics model of a projectile and did not consider the influence of a control loop, resulting in an unnecessarily complex design process.

Motivated by the previous investigations, in this study, a DRL-based guidance and control model is constructed for an accurate interception guidance problem with impact-angle constraints. Based on this model, two guidance and control strategies with impact-angle constraints are designed.

The main contributions of this study are as follows:

(1) To consider the effects of comprehensive flying object dynamics and the control mechanism, this study develops a Markov decision process model to address the guidance and control problem. Two DRL guidance and control strategies with the impact-angle constraint are proposed. The design of these strategies is based on the principles of dual-loop and integrated guidance and control;

(2) An improved reward mechanism that includes the state component is designed. This mechanism combines the sparse reward and dense reward to correct the impact-angle errors while reducing the flying object–target distance. The designed dense reward mechanism can ensure that when the state components of the state are near the target value, the small improvement in the strategy can still obtain a large reward difference so that the network can converge more easily to the optimal. Furthermore, the dense reward mechanism can normalize the reward value and reduce the influence of the scalar value of each different reward on the overall reward, which improves the sparse reward problem. In addition, to address the negative effects of an unbounded distribution on a bounded action space, this study modifies the PPO actor network's probability distribution by replacing the Gaussian distribution with a Beta distribution. This is because Beta distribution sampling can constrain the sampling value in the interval of [0, 1] so that the sampling action can be mapped to any desired action interval;

(3) In order to ensure the applicability of the model to different impact angle controls, we introduce the real-time impact-angle error into the state, which makes the agent pay more attention to the impact-angle error rather than a certain impact angle in the learning process. In addition, combined with an engineering background, we designed a reasonable initialization training scenario to ensure that the guidance control strategies can meet different scenarios and expected impact angles.

The remainder of this paper is organized as follows: the problem formulation is stated in Section 2. The specific process of DRL-based guidance and control strategies with an impact-angle constraint design method is given in Section 3. Section 4 is the verification of the results of the proposed method, and the conclusion is described in Section 5.

2. Environmental Description

This study aims to use the DRL method to design guidance and control strategies with impact-angle constraints, so it is necessary to transform the guidance and control problem into the framework of RL. Firstly, it is necessary to establish the environmental model required for the interaction of the agent in RL. The research object in this paper is a flying object with qualitative homogeneity, which is an aerodynamically controlled vehicle whose coupling between pitching and yawing channels can be omitted [34]. Hence, a guidance model is developed for the longitudinal plane motion of a flying object and a target with the background of air-to-ground guidance, as presented in Figure 1, wherein the model

parameters defined as follows: $x_1o_1y_1$ is a Cartesian inertial reference frame, M denotes the position of a flying object, and T is the position of a target; a_y is the normal acceleration; q is the LOS angle, and φ is the lead angle; θ represents the ballistic inclination; R is the relative distance between the flying object and the target; and V is the velocity.

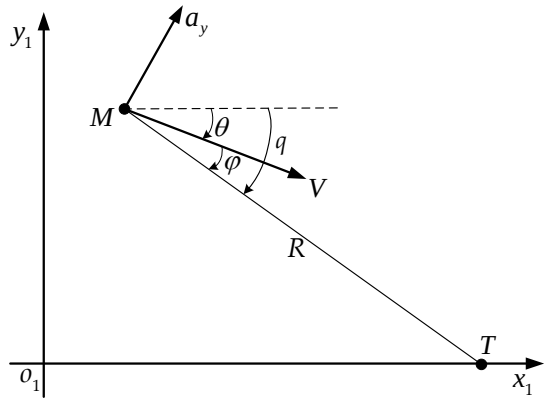


Figure 1. Illustration of the flying object–target geometric relationship.

For a stationary target, the two-dimensional relative kinematics between the flying object and target can be expressed as follows [32]:

$$\begin{cases} \dot{R} = -V\cos(q-\theta) \\ R\dot{q} = V\sin(q-\theta) \end{cases} \tag{1}$$

The relationship between the angles of the flying object in the longitudinal plane is shown in Figure 2.

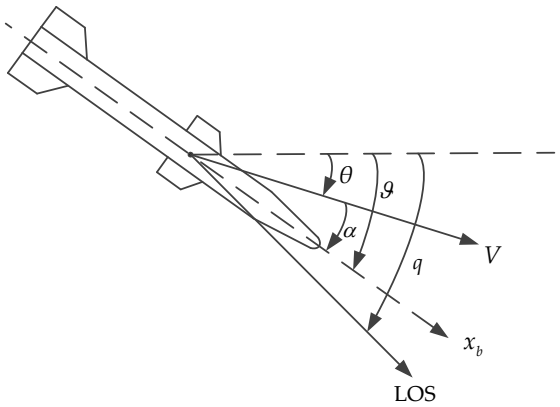


Figure 2. The relationship between the angles.

In Figure 2, x_b is the vertical axis of the flying object body; LOS represents the direction of the line of sight; and ϑ and α represent the pitch angle and AOA, respectively. Similar to the works of reference [35], the flying object dynamics are given as follows:

$$\begin{cases} \dot{V} = \frac{P\cos\alpha - c_x q_d S - mg\sin\theta}{m} \\ \dot{\theta} = \frac{P\sin\alpha}{mV} + \frac{c_y^{\alpha} q_d S}{mV}\alpha + \frac{c_y^{\delta} q_d S}{mV}\delta_z - \frac{g}{V}\cos\theta \\ \dot{\omega}_z = \frac{m_z^{\bar{\omega}} q_d SL}{J_z}\omega_z + \frac{m_z^{\alpha} q_d SL}{J_z}\alpha + \frac{m_z^{\delta} q_d SL}{J_z}\delta_z \\ \dot{\alpha} = \omega_z - \dot{\theta} \\ a_y = \frac{q_d S(c_y^{\alpha}\alpha + c_y^{\delta}\delta_z)}{m} \end{cases} \tag{2}$$

where m represents the flying object mass; ω_z is the pitch angular velocity; J_z is the pitch moment of inertia; P is the axial thrust of a flying object; S denotes the reference area; L is the reference length; q_d represents the incoming flow pressure, and $q_d = \rho V^2/2$, where ρ represents the air density at the flying altitude of the flying object; δ_z is the elevator deflection angle; $c_x = c_{x0} + c_x^{\alpha^2}\alpha^2$ is the resistance coefficient; c_{x0} is the zero-lift drag coefficient; $c_x^{\alpha^2}$ represents the derivative of the induced resistance coefficient to α^2; c_y^{α} represents the derivative of the lift coefficient to the AOA; c_y^{δ} is the derivative of the lift coefficient to the elevator deflection angle; m_z^{α} represents the derivative of the pitching moment coefficient to the AOA; $m_z^{\overline{\omega}}$ denotes the derivative of the pitching moment coefficient to the dimensionless pitch angular rate; and m_z^{δ} is the derivative of the pitching moment coefficient to the elevator deflection angle.

In order to intercept the target with the expected impact angle, the terminal position, and impact angle must satisfy the following relationships:

$$x_f = x_d, y_f = y_d, \theta_f = \theta_d \tag{3}$$

where the subscript "f" indicates the final state, and the subscript "d" relates to the expected state.

3. DRL-Based Guidance and Control Design with Impact-Angle Constraint

The main aim of using DRL for solving guidance and control problems is to attain the desired flying object's motion state by studying the guidance instructions via the trial-and-error method. There are two main approaches to the design of guidance and control systems: one is to design the guidance loop and control loop separately and independently, and another is to integrate the guidance and control loops into an overall control system. Therefore, this study applies the DRL method to design two distinct guidance and control strategies with the miss distance and impact-angle constraints.

Strategy 1 is double-loop guidance and control with the impact-angle constraint using DRL (DLGCIAC-DRL). As shown in Figure 3, IMU represents the inertial measurement unit. The DLGCIAC-DRL design includes the guidance and control loops, where the guidance loop uses the DRL policy to acquire the acceleration command a_{yc}, and the control loop drives the servo motors to control the flying object's motion based on the acceleration command a_{yc}.

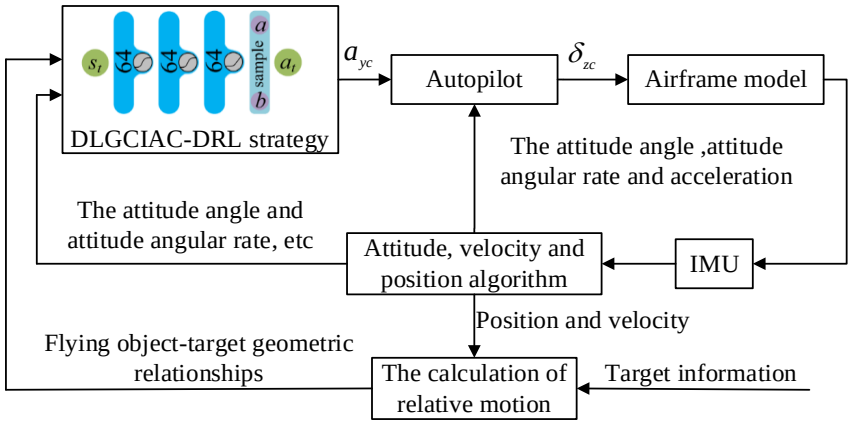


Figure 3. Block diagram of the DLGCIAC-DRL strategy.

Strategy 2 is integrated guidance and control with the impact-angle constraint using DRL (IGCIAC-DRL). As shown in Figure 4, the IGCIAC-DRL strategy considers guidance and control as a whole and is designed to generate elevator deflection angle commands directly using the DRL policy based on the relative kinematic information between the flying object and target, as well as the attitude and acceleration information of the flying object.

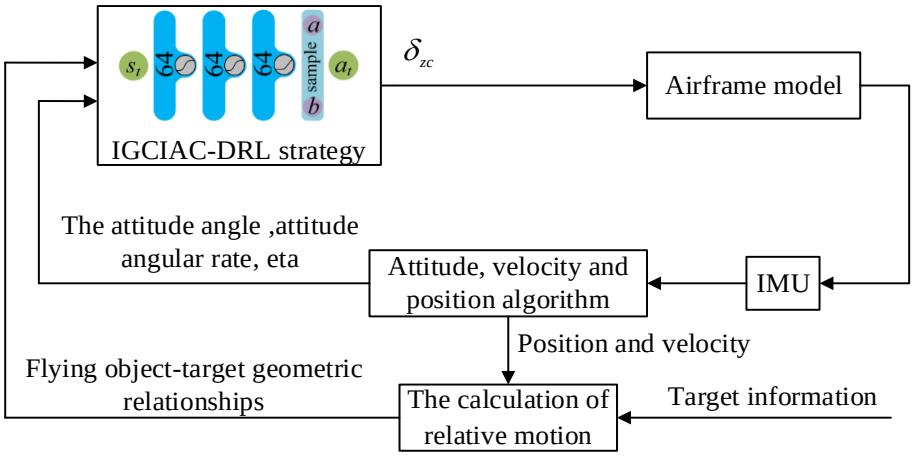


Figure 4. Block diagram of the IGCIAC-DRL strategy.

In the DLGCIAC-DRL strategy, the autopilot adopts the two-loop autopilot model previously studied and proposed in [36]. The autopilot model has been proven to have excellent control performance and will not be presented in detail in this paper.

3.1. DRL Model

The flight state of a flying object at two adjacent moments can be approximately regarded as a transfer between the states under a given control command, and the flying object's state in the next moment is related only to its state at the current moment; thus, the flight state of the flying object has a Markov property. By discretizing the flight process in the time domain, the guidance process of a flying object can be approximately modeled as a discrete-time Markov chain (DTMC). The specific design of the guidance process is to add the decision-making command to the Markov chain of the flying object flight process; therefore, the design process of the flying object guidance can be modeled as a Markov decision process (MDP).

The MDP is a sequential decision process that can be described by a five-tuple denoted by $\langle S,\ A,\ P(\ .\ ),\ R(\ .\ ),\ \gamma \rangle$. The specific model of the MDP for a guidance and control definition is as follows [13]:

$$\text{MDP}\begin{cases} S:\ \text{State set},\ S=\{s_1,\ s_2,\ \cdots s_n\} \\ A:\ \text{Action set},\ A=\{a_1,a_2,\cdots a_n\} \\ P(s_{t+1}|s_t,\ a_t):\ \text{The state transition function of the environment} \\ R(s_t,\ a_t,\ s_{t+1}):\ \text{The reward function of the environment} \\ \gamma:\ \text{Discount factor} \end{cases}$$

The interaction process between the agent and the environment in the MDP is depicted in Figure 5. The environment produces information describing the state s_t, and the agent interacts with the environment by observing the state s_t and choosing action a_t. The environment accepts the action a_t and transitions to the next state s_{t+1}, then returns the next state s_{t+1} and a reward to the agent. In this study, an agent cannot directly access the transition function $P(s_{t+1}|s_t,\ a_t)$ and the reward function $R(s_t,\ a_t,\ s_{t+1})$ and can obtain only specific information about its state s_t, action a_t, and reward r_t by interacting with the surrounding environment.

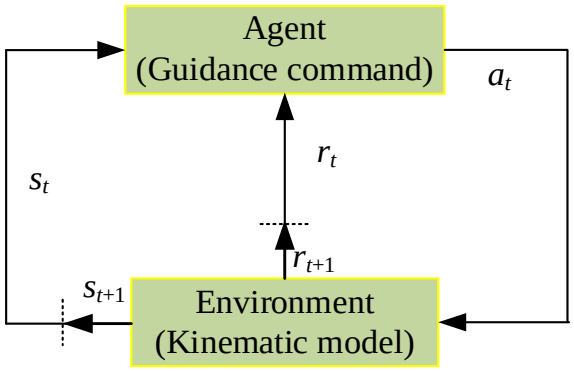


Figure 5. The interaction process between the agent and the surrounding environment.

The guidance control problem represents a continuous control problem. In each round, an agent observes its state s_t at time t and makes a decision on an action a_t to be taken according to the current policy. The policy π defines the mapping from a particular agent state to the corresponding action ($\pi : a \sim \pi(s)$). The guidance control model defines the next state under the performed action and obtains a reward from the environment. The trajectory τ generated by the control loop from the initial state s_0 to the final state s_T is expressed as follows: $(s_0, a_0, r_0), (s_1, a_1, r_1), \cdots, (s_T, a_T, r_T)$.

According to the optimization objective, the reward is defined as a weighted sum of all rewards on trajectory τ:

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t \tag{4}$$

where $\gamma \in [0, 1]$ represents the discount factor and T represents the state number of a scene data trajectory.

The objective function $J(\tau)$ is defined as an expectation of the trajectory return, and it is expressed as follows:

$$J(\tau) = E_{\tau\sim\pi}[R(\tau)] = E_{\tau\sim\pi}[\sum_{t=0}^{T} \gamma^t r_t] \tag{5}$$

Before implementing the DRL algorithm for model training, it is necessary to define the amount of information required for the interaction between an agent and the environment; namely, it is necessary to define the state s_t, action a_t, and reward r_t.

In the guidance and control problems with impact-angle constraints, the state vector must fully describe a flying object's motion and target geometry while considering the guidance constraints. Therefore, the state vector is defined as follows:

$$s_t : [V, \theta, \omega_z, \vartheta, R, q, \theta_{error}, dq] \tag{6}$$

where $\theta_{error} = q - \theta_d$ represents the error between the actual and expected values of the impact angle.

In this study, one of the state components is designed as an angle error instead of a specified impact angle. The idea of this modification is to encourage the RL to focus more on the angle error rather than on the precise impact angle. In this way, the suitability for different impact-angle requirements of the guidance model can be increased.

Standardizing the state vector is necessary to eliminate the impact of dimensionality on neural network training. In this study, the mean and standard deviation of each state component are used to standardize the state vector. The state vector normalization is performed using the following formula:

$$s_t{}^i = \frac{s'_t{}^i - \mu_t{}^i}{\sigma_t{}^i} \tag{7}$$

where ${s'_t}^i$ represents the components of the state vector before normalization; ${s_t}^i$ represents the components of the state vector after normalization; and ${\mu_t}^i$ and ${\sigma_t}^i$ are the mean and standard deviation of ${s_t}^i$, respectively; the ${\mu_t}^i$ and ${\sigma_t}^i$ values are updated in the training process through the single-step updating of the sampled data. Initially, their values may exhibit significant deviation, but as the network deepens, they become increasingly close to their true values.

The values of ${\mu_t}^i$ and ${\sigma_t}^i$ are updated as follows:

$$\begin{cases} {\mu_t}^i = {\mu_{t_{\text{old}}}}^i + ({x_n}^i - {\mu_{t_{\text{old}}}}^i)/n \\ {\sigma_t}^i = \sqrt{\frac{n-1}{n}({\sigma_{t_{\text{old}}}}^i)^2 + \frac{1}{n-1}({\mu_t}^i - {x_n}^i)} \end{cases} \tag{8}$$

where ${\mu_{t_{\text{old}}}}^i$ and ${\sigma_{t_{\text{old}}}}^i$ represent the expected value and standard deviation of the state variables before the update, respectively; n is the current number of sampling steps; and ${x_n}^i$ represents the state variables at the nth step of sampling.

In the DLGCIAC-DRL strategy, an agent obtains an acceleration command a_{yc} by sampling the probability distribution selected in the policy π. As a result, it configures the action a_t as an acceleration command a_{yc}. Therefore, to ensure safety, it is imperative to constrain the acceleration command a_{yc} as follows:

$$a_t : a_{yc} \in [-a_{yc}^{\max}, a_{yc}^{\max}] \tag{9}$$

where $a_{yc}^{\max}$ represents the upper limit of the acceleration command.

Similarly, in the IGCIAC-DRL strategy, an action can be designed as follows:

$$a_t : \delta_{zc} \in [-\delta^{\max}, \delta^{\max}] \tag{10}$$

where $\delta^{\max}$ represents the upper limit of the elevator deflection angle command.

In this study, policy π adopts the Beta distribution and normalizes actions to the range of [0, 1] during the sampling process. Therefore, it is necessary to reverse the normalization of action values during the interaction between an agent and its environment. The reverse normalization formula for action values in the DLGCIAC-DRL strategy is given by:

$$a_{yc} = a_{yc}^{\max}(2a_t - 1) \tag{11}$$

Similarly, in the IGCIAC-DRL strategy, an action can be designed as follows:

$$a_{yc} = \delta^{\max}(2a_t - 1) \tag{12}$$

The reward signal represents an objective that agents need to maximize. Therefore, using a reasonable reward mechanism is crucial to achieving the optimal training effects as this mechanism directly affects the convergence rate and even the feasibility of the RL algorithms. In addition to constraining the impact angle, the guidance processes need to also constrain the off-target deviation. This study designs the reward function as a function related to the state components as follows:

$$r_{i,i\in s} = \frac{2}{e^{|i^{cur} - i^{tar}| \times \Theta}} - 1 \tag{13}$$

where i^{cur} represents the value of the current state component, i^{tar} is the target value of the state component of an intelligent agent, and Θ is the scaling coefficient of the intelligent agent.

In order to describe the advantages of this reward function design in more detail, a specific example will be given below to explain. For instance, for the Θ value of 0.02, the reward function is shown in Figure 6.

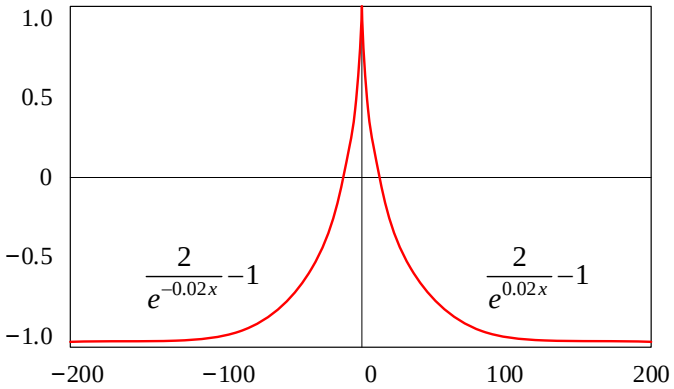


Figure 6. The reward function curve.

As shown in Figure 6, the closer the state component is to the target value, the greater the reward value is. Moreover, the reward value changes dramatically near the target value, which ensures that an agent can obtain a large reward difference even when making decisions near the target value, which makes the network easier to converge to the optimal solution. Moreover, the proposed reward function normalizes the reward values to the range of $[-1, 1]$, which is beneficial to reducing the impact of different scalar reward values on the overall reward value and addressing the problem of sparse rewards. The total reward value represents a weighted sum of the rewards of all relevant state components, and it is defined by:

$$r_t = \sum_{i \in s} w_i \cdot r_i \tag{14}$$

where w_i represents the weight coefficient, indicating the significance level of the ith state component to the intelligent agent, and it satisfies the condition of $\sum_{i \in s} w_i = 1$.

In practical engineering, the main focus is on the impact-angle error and the distance R between a flying object and a target, so the reward value can be calculated by:

$$r_t = w_{\theta_{error}} r_{\theta_{error}} + w_R r_R \tag{15}$$

Further, to increase the convergence speed of the training process, additional sparse rewards are introduced in addition to the aforementioned reward. Therefore, the final form of the reward can be obtained as follows:

$$r_t = \begin{cases} r_t + 20 - R, & R < 20 \text{ m} \\ r_t - 50, & H > 6000 \text{ m} \\ r_t - 20, & H = 0 \cap R > 100 \text{ m} \end{cases} \tag{16}$$

According to Equation (16), it can be observed that there is an additional reward of $20 - R$ when R is less than 20 m, an additional penalty of -50 when H is over 6000 m, and an additional penalty of -20 when the H equals 0 while the flying object is still far from the target, i.e., more than 100 m. These extra rewards and penalties contribute to improving the training efficiency. The values of the relevant parameters of the reward function are listed in Table 1.

Table 1. The values of the relevant parameters of the reward function.

${\theta_{error}}^{tar}$	R^{tar}	$w_{\theta_{error}}$	w_R	Θ
0	0	0.8	0.2	0.2

3.2. PPO Algorithm and Enhancement

The PPO algorithm represents an on-policy method based on RL that uses an actor-critic framework. The PPO employs a stochastic policy, denoted by $\pi_\phi(a_t|s_t)$, where each action is considered a random variable that follows a predefined probability distribution,

and to implement the policy, the probability distribution should be parameterized. The PPO uses the output of an actor network as the relevant parameters to select the corresponding actions by sampling from the distributed set [22].

The PPO typically uses Gaussian distribution as a probability distribution for policy sampling. However, due to the Gaussian distribution's infinite range, sampled actions are often restricted within the acceptable action boundaries, which can affect algorithm efficiency. In view of that, this study proposes an alternative bounded probability distribution, the Beta distribution, to address this limitation. The Beta distribution is a probability distribution determined by two parameters a and b, and the domain is [0, 1]. The sampling value obtained through the Beta distribution must be in the interval [0, 1] so that the experiment can map the sampling value in the interval [0, 1] to any desired action interval. Beta distribution can be expressed as follows:

$$\begin{cases} f(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}, x \in [0,1] \\ B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}dx \end{cases} \tag{17}$$

The probability density function (PDF) and the cumulative distribution function (CDF) of the Beta distribution are presented in Figure 7.

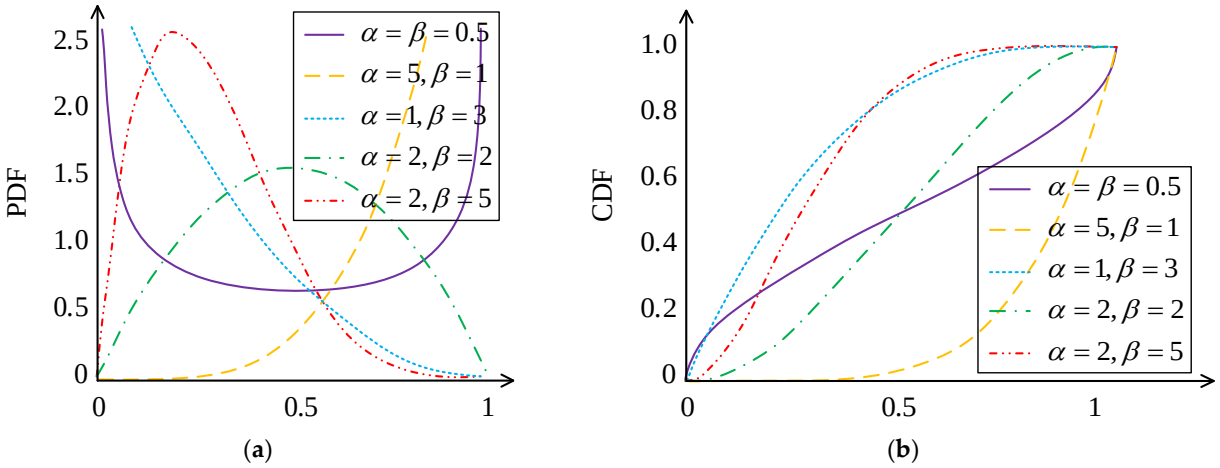


Figure 7. The PDF and CDF curves of the Beta distribution: (**a**) PDF and (**b**) CDF.

As shown in Figure 7, using the Beta distribution for sampling can restrict actions within the interval of [0, 1] so the sample values can be mapped to any desired action range, which eliminates the detrimental effect of unbounded distribution on the action space that is limited.

The PPO algorithm stores the trajectories obtained after sampling into a replay buffer. The policy used in the sampling process is referred to as an old policy, denoted by $\pi_{\phi_{\text{old}}}(a_t|s_t)$, and it differs from the updated policy. To obtain the total environmental rewards at each state, the expected return is typically expressed in the form of value functions. Particularly, the state value function $V_\chi^\pi(s_t)$ and the action value function $Q_\theta^\pi(s_t,\ a_t)$ are defined as follows:

$$\begin{cases} V_\chi^\pi(s_t) = \mathbb{E}_{s_0=s_t,\tau\sim\pi_\phi}\left[\sum_{t=0}^{T}\gamma^t r_t\right] \\ Q_\theta^\pi(s_t,\ a_t) = \mathbb{E}_{s_0=s_t,a_0=a_t,\tau\sim\pi_\phi}\left[\sum_{t=0}^{T}\gamma^t r_t\right] \end{cases} \tag{18}$$

The value of $Q_\theta^\pi(s_t,\ a_t)$ is strongly related to the value of $V_\chi^\pi(s_t)$, which represents the anticipated Q-value of a feasible action a_t for a policy π_ϕ and a particular state s_t, and it is calculated by:

$$V_\chi^\pi(s_t) = \mathbb{E}_{a_t\sim\pi_\phi(s_t)}[Q_\theta^\pi(s_t,\ a_t)] \tag{19}$$

The critic network of the PPO learns the mapping relationship between a state s_t and the value function $V_\chi^\pi(s_t)$. Namely, the critic network takes a state s_t as input and generates the value function $V_\chi^\pi(s_t)$ as output. The PPO employs the advantage function $A^\pi(s_t,\ a_t)$ as a reinforcement signal, which measures the superiority of a specific action in a particular state compared to the average policy. The advantage function is defined by:

$$A^\pi(s_t,\ a_t) = Q_\theta^\pi(s_t,\ a_t) - V_\chi^\pi(s_t) \tag{20}$$

This paper applies the generalized advantage estimation (GAE) method to estimate the advantage function and reduce the variance of estimates while minimizing bias. Specifically, GAE calculates the exponentially weighted average of all n-step return advantage functions:

$$A_{\mathrm{GAE}}^\pi(s_t, a_t) = \sum_{\ell=0}^{\infty} (\gamma\lambda_{\mathrm{GAE}})^\ell \delta_{t+1} \tag{21}$$

where λ_{GAE} represents the GAE parameter, and δ_t can be obtained by computing the advantage function using the old critic network as follows:

$$\delta_t = r_t + \gamma V_{\chi_{\mathrm{old}}}^\pi(s_{t+1}) - V_{\chi_{\mathrm{old}}}^\pi(s_t) \tag{22}$$

During the training process, the critic network's parameters χ are adjusted to ensure a close approximation of the output $V_\chi^\pi(s_t)$ to the desired value $Q_\theta^\pi(s_t,\ a_t)$. Consequently, the loss function of the critic network can be formulated as follows:

$$E(\theta) = \frac{1}{T_s}\sum_{t=0}^{T_s} \left\| A_{\mathrm{GAE}}^\pi(s_t, a_t) + V_{\chi_{\mathrm{old}}}^\pi(s_t) - V_\chi^\pi(s_t) \right\|^2 \tag{23}$$

where T_s represents the size of a single batch update.

Further, the critic network's parameters are updated as follows:

$$\chi = \chi - \alpha_\chi \nabla_\chi J(\chi) \tag{24}$$

where α_χ represents the learning rate of the critic network.

The actor network of the PPO algorithm takes the input state s_t and determines the values of parameters a and b for the Beta distribution used by the policy π_ϕ. The PPO algorithm combines the benefits of the policy gradient (PG) and trust region policy optimization (TRPO). By leveraging the concept of importance sampling, the TRPO considers every policy update step as an optimization problem. The TRPO introduces a modified surrogate objective function (SOF), which is expressed as follows:

$$\begin{gathered} J(\phi) = \frac{1}{T_s}\sum_{t=0}^{T_s}\left[\frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{\mathrm{old}}}(a_t|s_t)} A_{\mathrm{GAE}}^\pi\right] \\ \text{s.t. } \overline{D}_{KL}\left(\pi_{\phi_{\mathrm{old}}}, \pi_\phi\right) \leqslant \delta \end{gathered} \tag{25}$$

where δ is a constant, and B is the Kullback–Leibler (KL) divergence between the old and new strategies which is used to measure the difference in the probability distribution between the old and new strategies.

Therefore, at each update of the policy, $\pi_\phi(a_t \mid s_t)$ can be optimized using samples collected by the old policy $\pi_{\phi_{\mathrm{old}}}(a_t \mid s_t)$ and the advantage function $A_{\mathrm{GAE}}^\pi(s_t, a_t)$ estimated by the old critic network.

The TRPO uses a constant A to constrain the magnitude of policy updates, but computing the KL divergence can be relatively complex in practice and, thus, difficult to implement. In contrast, the PPO directly modifies the SOF via a clipping function to restrict the magnitude of policy updates, which simplifies the computational process. To encourage the

exploration using a variety of actions, in this study, the PPO objective function includes an entropy regularization term, and it can be expressed as follows:

$$J(\phi) = \frac{1}{T_s}\sum_{t=0}^{T_s}[\min(r_t(\phi)A^{\pi}_{\mathrm{GAE}},\ \mathrm{clip}(r_t(\phi),\ 1-\varepsilon,\ 1+\varepsilon)A^{\pi}_{\mathrm{GAE}}) + \varsigma H_t] \tag{26}$$

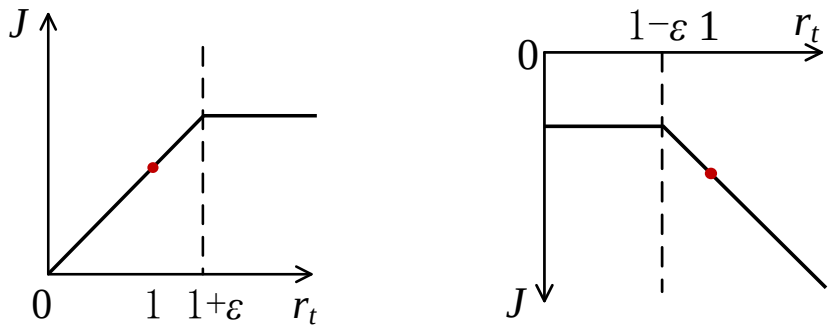
where ε is the clipping parameter used for the constraint policy update; ς is the entropy regularization factor; H_t is the policy entropy; $\mathrm{clip}(r_t(\phi),\ 1-\varepsilon,\ 1+\varepsilon)$ is a trimming function; and $r_t(\phi)$ is the ratio function; it is defined as follows:

$$r_t(\phi) = \frac{\pi_{\phi}(a_t \mid s_t)}{\pi_{\phi_{\mathrm{old}}}(a_t \mid s_t)} \tag{27}$$

Further, $\mathrm{clip}(r_t(\phi),\ 1-\varepsilon,\ 1+\varepsilon)$ is defined as follows:

$$\mathrm{clip}(r_t(\phi),\ 1-\varepsilon,\ 1+\varepsilon) = \begin{cases} 1-\varepsilon, & r_t(\phi) < 1-\varepsilon \\ 1+\varepsilon, & r_t(\phi) > 1+\varepsilon \\ r_t(\phi), & 1-\varepsilon \le r_t(\phi) \le 1+\varepsilon \end{cases} \tag{28}$$

The relationship between the objective function $J(\phi)$ and the ratio function $r_t(\phi)$ is presented in Figure 8, where it can be seen that increasing the probability of the corresponding action $\pi_{\phi}(a_t \mid s_t)$ can increase the objective function's value when $A^{\pi}_{\mathrm{GAE}} > 0$. Nevertheless, if $\pi_{\phi}(a_t \mid s_t)$ surpasses $(1+\varepsilon)\pi_{\phi_{\mathrm{old}}}(a_t \mid s_t)$, the objective function will be truncated and calculated as $(1+\varepsilon)A^{\pi}_{\mathrm{GAE}}(s_t, a_t)$, which obstructs $\pi_{\phi}(a_t \mid s_t)$ from deviating excessively from $\pi_{\phi_{\mathrm{old}}}(a_t \mid s_t)$. Analogously, when $A^{\pi}_{\mathrm{GAE}} < 0$, the similar changing trend of the objective function can be observed.



(**a**) (**b**)

Figure 8. The variation law of the objective function J with the change in the rt value: (**a**) $A^{\pi}_{\mathrm{GAE}} > 0 > 0$; (**b**) $A^{\pi}_{\mathrm{GAE}} > 0 < 0$.

The updating formula of parameters ϕ of the actor network is as follows:

$$\phi = \phi + \alpha_{\phi}\nabla_{\phi}J(\phi) \tag{29}$$

where α_{ϕ} represents the learning rate of the actor network.

To ensure training stability in the later stages, this study decreases the learning rate during the training process. The learning rate updating method is defined as follows:

$$\begin{cases} \alpha_{\chi} = \alpha_{\chi}(1 - \kappa(T_{\mathrm{total}}/T_{\mathrm{train}})) \\ \alpha_{\phi} = \alpha_{\phi}(1 - \kappa(T_{\mathrm{total}}/T_{\mathrm{train}})) \end{cases} \tag{30}$$

where κ represents the decay factor of the learning rate, T_{total} is the total number of training iterations, and T_{train} denotes the current number of training iterations.

3.3. Network Structure and Learning Process

In this study, both the actor and critic networks of the PPO algorithm represent a five-layered structure consisting of an input layer, three hidden layers, and an output layer. Through attempts, the specific parameters of the network structure for this study are determined, as seen in Table 2. The numbers in the table represent the number of neurons in the layers.

Table 2. Network structural parameters.

Layer	Actor Network	Critic Network
Input layer	8 (state dimension)	8 (state dimension)
Hidden layer 1	64	64
Hidden layer 2	64	64
Hidden layer 3	64	64
Output layer	2 (parameters of the Beta distribution)	1 (action dimension)

In every network layer except in the output layer, an activation function of hyperbolic tangent (*Tanh*) is used to activate each layer of the networks and prevent input saturation, defined as follows:

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{31}$$

Since the parameters of the Beta distribution are required to be larger than one, the *Softplus* function is used in the output layer of the actor network. Namely, adding one to the output value of the *Softplus* function ensures meeting the requirement for the Beta distribution parameters as the output of the *Softplus* function is always larger than zero. The *Softplus* function is defined as follows:

$$Softplus(x) = \log(1 + e^x) \tag{32}$$

The output layer of the actor network does not have an activation function. The specific structures of the actor and critic networks are shown in Figure 9.

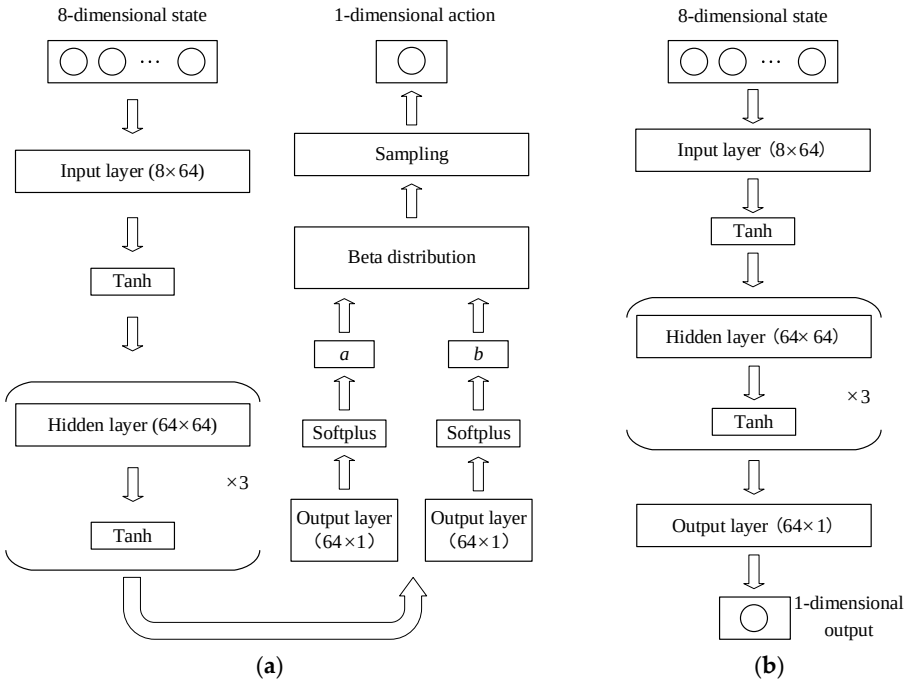


Figure 9. The structures of the actor and critic networks: (**a**) actor network and (**b**) critic network.

As mentioned before, RL is a machine learning technique where an agent interacts with the surrounding environment to maximize its reward by continuously learning and optimizing its policy. During the agent–environment interaction, RL updates the agent state s_t based on the motion equations and obtains an action a_t using the probability distribution of the policy sampling. Then, it calculates the real-time reward based on the state components. In a single learning process, first, the RL-based algorithm interacts N_s times with the environment using the old policy to generate a trajectory sequence denoted by $\tau: (s_0, a_0, r_0), \cdots, (s_{N_s}, a_{N_s}, r_{N_s})$, where N_s is the length of the replay buffer used in the PPO algorithm. Next, it stores the generated trajectory τ into the replay buffer. To improve the training efficiency, this study performs batch training on the data acquired by sequentially processing a number of T_s-length trajectories in the replay buffer.

In the actor network updating process, first, the advantage values are estimated using Equation (23). Then, the action probability $\pi_{\phi_{\text{old}}}(a_t \mid s_t)$ is calculated for the already taken actions using the probability density function of the Beta distribution obtained from the old policy ϕ_{old}. Next, the action probability $\pi_{\phi}(a_t \mid s_t)$ is calculated based on the updated policy ϕ. Then, the ratio of action probabilities $r_t(\phi)$ is obtained to compute the objective function using Equation (28). Finally, the Adam optimizer is employed to update the actor network parameters and maximize the objective function value.

In the critic network updating process, the state value function $V_{\chi}^{\pi}(s_t)$ value is calculated based on the updated critic network and the loss function using Equation (25). Then, the Adam optimizer is employed to update the critic network parameters to minimize the loss. During a single learning process, the network parameters are updated five times based on the data stored in the replay buffer.

The PPO is a typical on-policy algorithm where the sequence trajectory τ generated from the agent–environment interaction cannot be reused. After each learning iteration, the replay buffer is cleared, and the learning process is repeated until the training process is completed. The learning process of the proposed guidance and control strategy based on the PPO algorithm is illustrated in Figure 10.

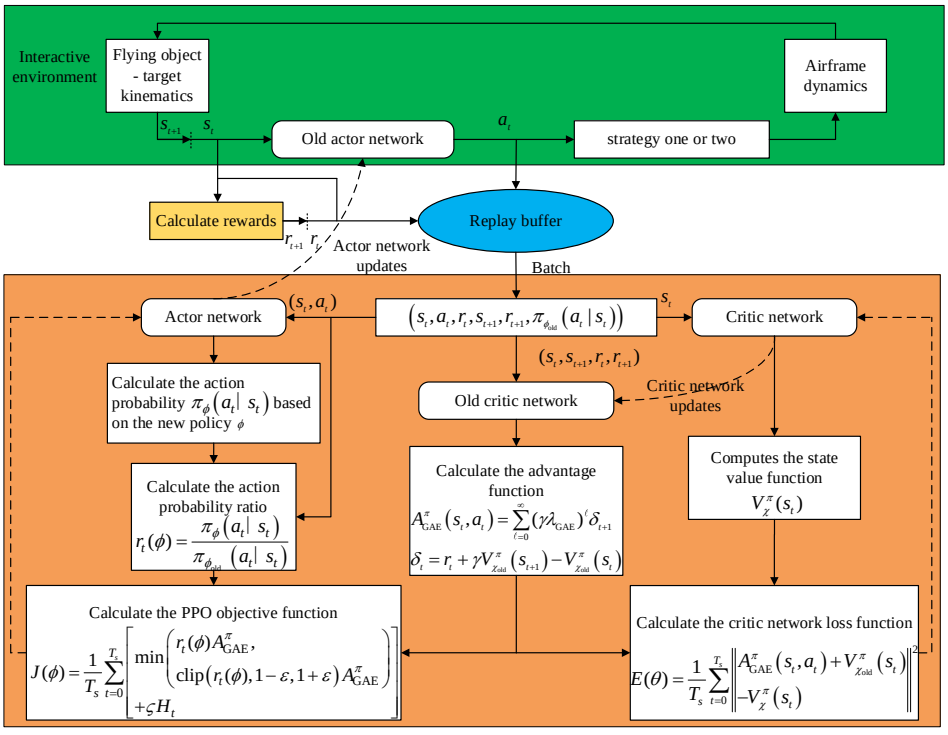


Figure 10. The learning scheme of the guidance and control strategy based on the PPO algorithm.

4. Simulation and Analysis

Numerical simulations were conducted to evaluate the efficiency of the proposed DRL-based guidance and control strategies with an impact-angle constraint. First, training scenarios were designed based on an engineering background and reasonable hyperparameters were used to train the two strategies. Next, the training results were compared and analyzed, and the two guidance control strategies were tested under different operating conditions. Then, Monte Carlo simulations were conducted with the introduction of error models. Finally, the proposed guidance and control strategies were compared with the biased proportional navigation-based angle constraints guidance (BPN-ACG) [5] and trajectory shaping guidance (TSG) [9], both widely used in engineering.

4.1. Training Process

During the training process, the initialization of the environmental parameters was performed at the beginning of each training epoch. For a good generalization capability of the obtained guidance and control strategies, the training was conducted for different initial states. In this research, the initial launch height of the flying object was about 2500 m, and the lateral distance between the flying object and the target was about 5000 m. Considering the engineering application scenario, the target position was set to (5000, 0), and the initial position of a flying object was randomly selected in the green area shown in Figure 11.

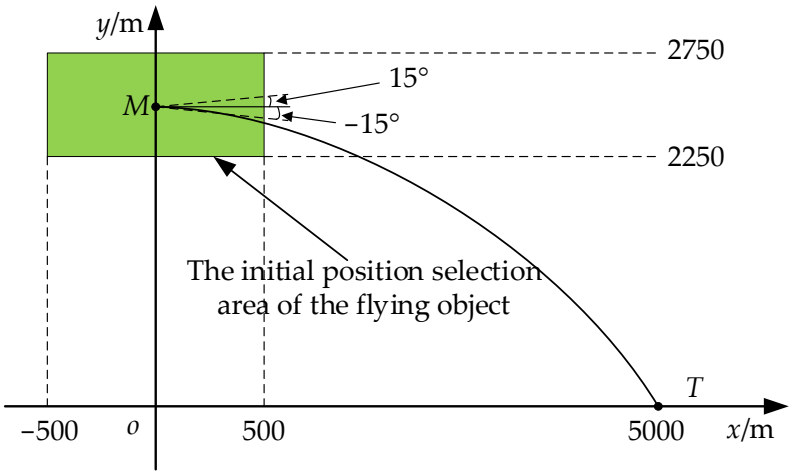


Figure 11. The initial training scenario.

Table 3 presents the initial values of the state parameters during the model training process.

Table 3. The settings of relevant model training parameters.

Parameter	Value
Initial horizontal coordinate of flying object position	−500–500 m
Initial vertical coordinate of flying object position	2250–2750 m
Initial velocity V	180–250 m/s
Initial ballistic inclination θ	−10–10°
Expected impact angle θ_d	−15–−80°

In this study, the flying object's parameters used in the experiments were as follows: the mass of the flying object was m = 11.7 kg, the reference area was S = 0.0176625 m^2, the reference length was L = 0.55 m, the air density was ρ = 1.225 kg/m^3, and the moment of inertia was J_z = 0.5. The axial thrust of flying object P was set to a constant value of 90 N. The relevant aerodynamic parameters are shown in Table 4. Further, to reduce the effect of the integration step size on the miss distance, the training environment was updated

using a fourth-order Runge–Kutta integrator. Particularly, when the flying object altitude was above 0.6 m, the integration step size was set to 0.002 s; when the flying object altitude was less than or equal to 0.6 m but larger than 0.06 m, the integration step size was set to 0.0002 s; when the flying object altitude was less than or equal to 0.06 m, the integration step size was set to 0.00001 s.

Table 4. Related aerodynamic parameters.

c_{x0}	$c_x^{\alpha^2}$ (rad^{-2})	c_y^{α} (rad^{-1})	c_y^{δ} (rad^{-1})	m_z^{α} (rad^{-1})	m_z^{δ} (rad^{-1})	$m_z^{\bar{\omega}}$ (s·rad^{-1})
0.3092	16.4163.	15.2131	2.9395	−1.2864	−1.1976	−0.8194

The training iteration terminated when any of the following conditions were met:

(1) The flying object landed;
(2) The flying object attained an altitude of more than 6000 m;
(3) The maximum number for the training epoch had been reached.

Table 5 lists the relevant hyperparameters of the PPO algorithm.

Table 5. Hyperparameters for the PPO algorithm.

Parameter	Value	Parameter	Value
The maximum number of training epochs for the DLGCIAC-DRL strategy	10,000	ς	0.01
The maximum number of training epochs for the IGCIAC-DRL strategy	15,000	Hidden layer size	64
Algorithm decision interval	0.02 s	Optimizer eps parameter	1×10^{-5}
N_s	1536	α_χ	3×10^{-4}
T_s	512	α_ϕ	3×10^{-4}
λ	0.95	κ	0.9
The number of times the experience replay buffer data are used for each update	5	ε	0.2
γ	0.99	Random seed	10

Following the proposed training plan, the DLGCIAC-DRL and IGCIAC-DRL strategies were trained separately. The changes in the cumulative reward, miss distance, and terminal impact-angle error of the two strategies during the training process are presented in Figures 12 and 13.

The results in Figures 12 and 13 demonstrate the convergence of the reward values for the two strategies, as well as a gradual reduction in both the miss distance and terminal impact-angle error with the introduction of autonomous learning. The DLGCIAC-DRL strategy exhibited stability after approximately 3000 training iterations, whereas the IGCIAC-DRL strategy required approximately 10,000 iterations to fully converge. This was due to the control loop in the DLGCIAC-DRL strategy which had been designed in advance, so the agent only needed to learn the guidance strategy in conjunction with the established control loop, accelerating the training process. Although the IGCIAC-DRL strategy simplified the design process by integrating the guidance and control loops, this approach introduced a complex coupling relationship. In contrast to the DLGCIAC-DRL strategy, the IGCIAC-DRL strategy had to learn the guidance and control strategy from the beginning, which significantly increased the learning complexity and reduced the convergence speed.

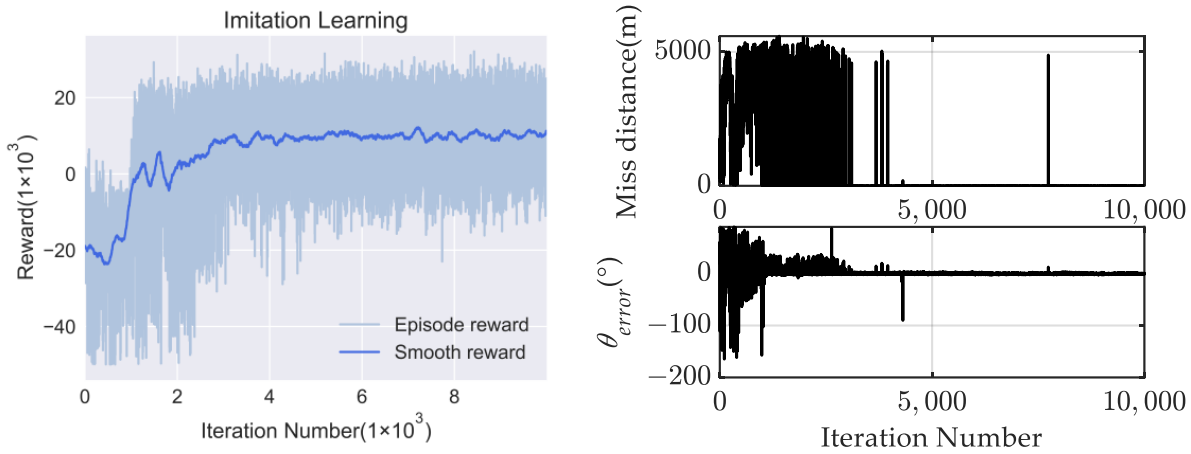


Figure 12. The training performances of the DLGCIAC-DRL strategy: (**a**) Cumulative reward; (**b**) Miss distance and terminal impact-angle error.

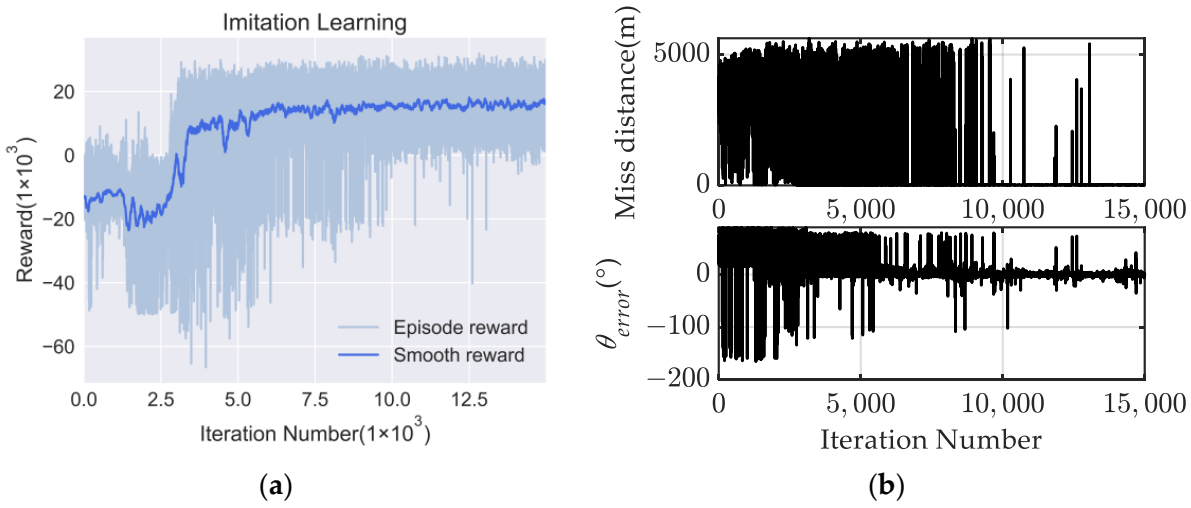


Figure 13. The training performances of the IGCIAC-DRL strategy: (**a**) cumulative reward and (**b**) miss distance and terminal impact-angle error.

4.2. Test Results

4.2.1. Comparative Analysis of the DLGCIAC-DRL and IGCIAC-DRL Strategies

After training, the actor networks of the two strategies were used as learned guidance and control models. To verify the applicability of the two models, a series of scenarios were constructed to perform simulation tests on the two guidance and control models. The expected impact angle was set to 15°, 30°, 45°, 60°, and 80°, and fifty randomly selected initial states were simulated for each expected impact angle based on Table 3. Figure 14 is a comparison of the simulation results under the different expected impact angles using the DLGCIAC-DRL strategy.

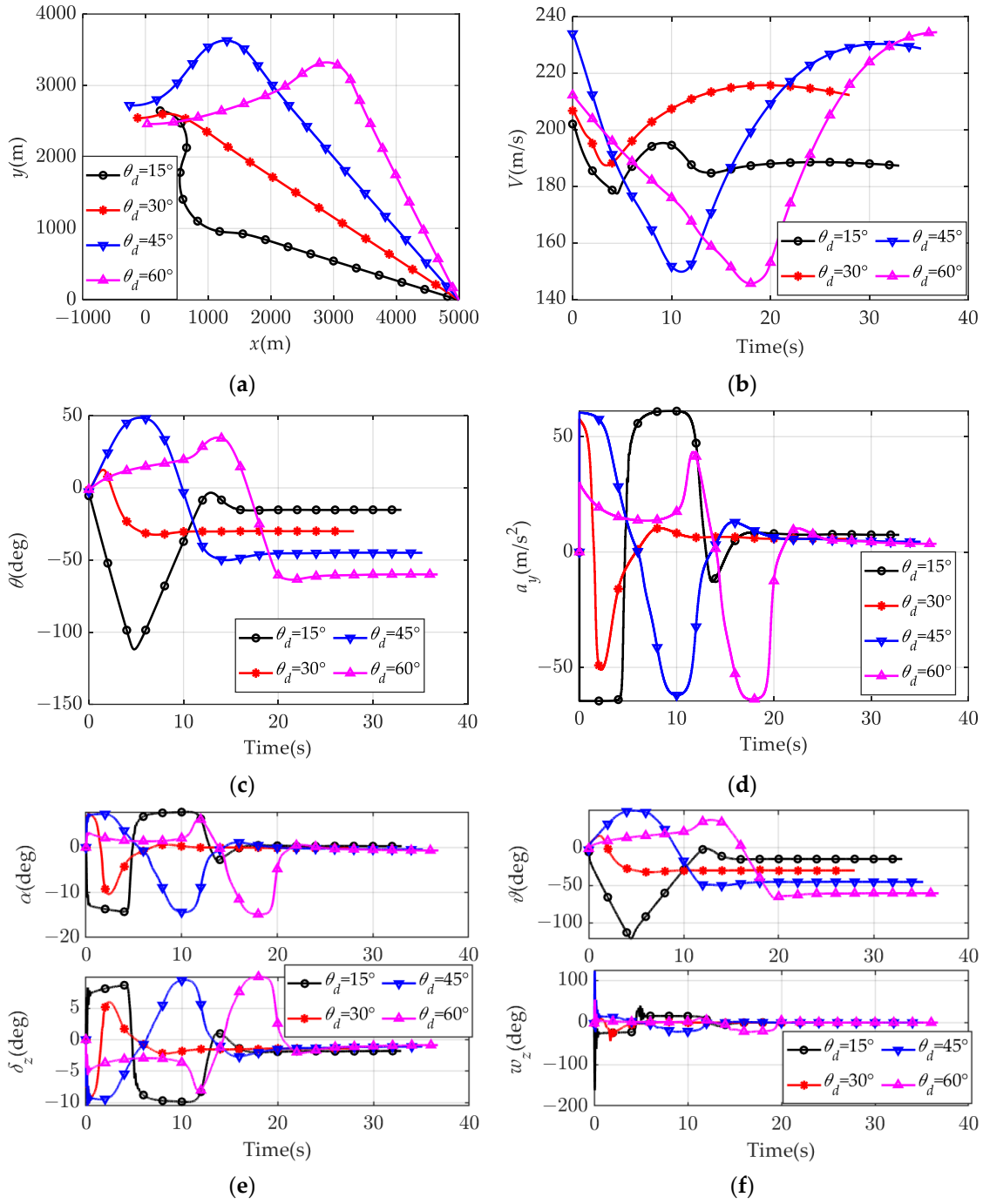


Figure 14. Comparison of simulation results under different expected impact angles using the DLGCIAC-DRL strategy: (**a**) ballistic trajectory; (**b**) speed variation; (**c**) trajectory inclination; (**d**) acceleration; (**e**) AOA and elevator deflection angle; and (**f**) pitch angle and pitch angular rate.

The statistics of the miss distance and terminal impact-angle error using the DLGCIAC-DRL strategy are shown in Figure 15.

The comparison of simulation results under different expected impact angles using the IGCIAC-DRL strategy is shown in Figure 16.

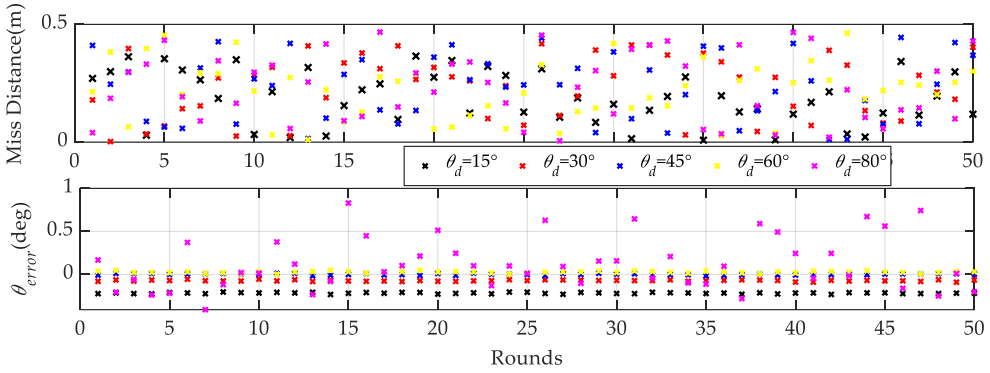


Figure 15. The statistics of the miss distance and terminal impact-angle error using the DLGCIAC-DRL strategy.

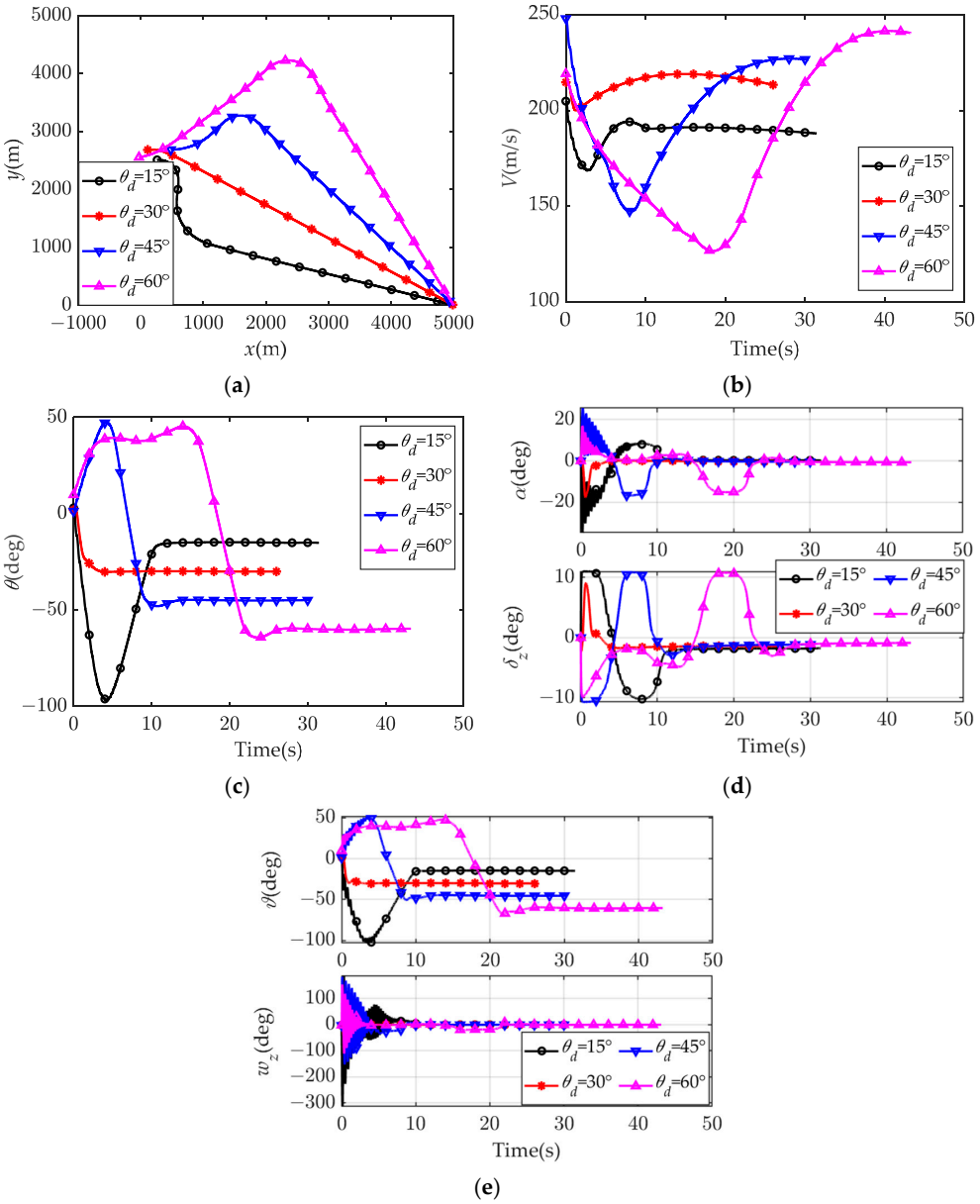


Figure 16. The simulation results of the IGCIAC-DRL strategy: (**a**) ballistic trajectory; (**b**) speed variation; (**c**) trajectory inclination; (**d**) AOA and elevator deflection angle; and (**e**) pitch angle and pitch angular rate.

The statistics of the miss distance and terminal impact-angle error using the IGCIAC-DRL strategy are shown in Figure 17.

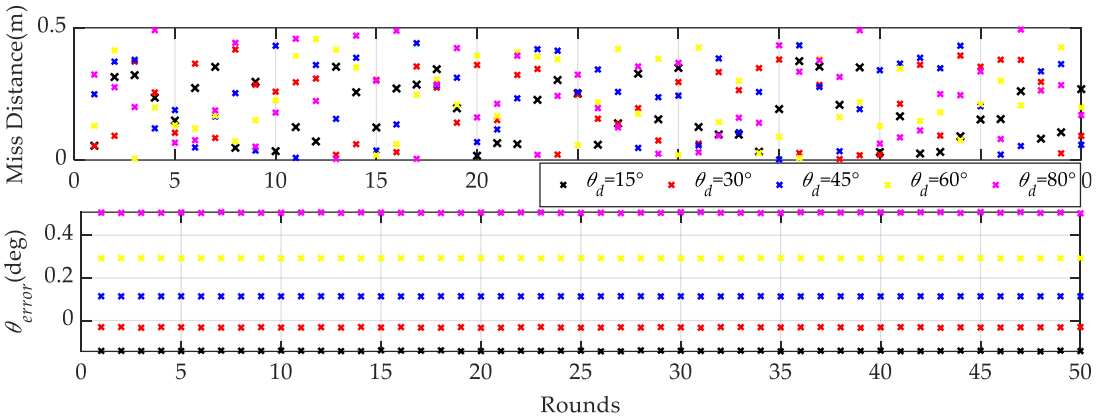


Figure 17. The statistics of the miss distance and terminal impact-angle error using the IGCIAC-DRL strategy.

The statistical data on the impact-angle error θ_{error} and miss distance of the two strategies are given in Table 6.

Table 6. The statistical data on the impact-angle error and miss distance of the DLGCIAC-DRL and IGCIAC-DRL strategies.

	θ_{error} (°)		Miss Distance (m)	
	DLGCIAC-DRL	**IGCIAC-DRL**	**DLGCIAC-DRL**	**IGCIAC-DRL**
Maximum value	0.8292	0.5085	0.4626	0.4932
Average value	−0.029	0.1476	0.1989	0.2199
Standard deviation	0.1762	0.2314	0.1299	0.1359
Mean squared error	0.0656	0.0751	0.0592	0.0668

As shown in Figures 14 and 16, both strategies could hit the target at the expected impact angle, and the majority of flying object maneuvers were completed prior to guidance; the guidance was achieved by an approximately linear impact method in the latter stages. In Figures 14d,e and 16d, it can be seen that both strategies could constrain the normal acceleration and elevator deflection angle within the usable range. Also, in Figures 14f and 16e, it can be seen that compared to the IGCIAC-DRL strategy, the DLGCIAC-DRL strategy had a smaller range of pitch rate oscillation prior to guidance and a more stable guidance performance due to the assistance of the control loop. Further, according to the results in Figures 15 and 17 and Table 6, the terminal impact-angle error was within 1°, and the miss distance was within 0.5 m for both strategies. Particularly, the DLGCIAC-DRL strategy had superior performance to the IGCIAC-DRL strategy, achieving a mean pitch angle error and mean miss distance of −0.029 and 0.1989, while those of the IGCIAC-DRL strategy were 0.1476 and 0.2199, respectively. Overall, in the absence of interference, both strategies could exhibit favorable performance.

4.2.2. Monte Carlo Simulations

The Monte Carlo simulation experiments were conducted to verify the feasibility of the two strategies in engineering applications, considering factors such as aerodynamic perturbation and sensor errors. Moreover, in the experiments, the TSG and BPN-ACG methods were used for comparison. Following commonly encountered engineering applications, the terminal impact angle was selected from a range between 30° and 60°, and the flying object's initial states were randomly set based on the values in Table 3. A total of

1000 Monte Carlo simulations were performed for each method, and the error models used during the simulation are presented in Table 7.

Table 7. Error models used in the Monte Carlo simulations.

Variables	Type of Error Distribution	Error Magnitude (2σ)
Resistance coefficient (c_x)	Gaussian	10%
Lift coefficient (c_y)	Gaussian	10%
Pitching moment coefficient (m_z)	Gaussian	10%
Pitching rotational inertia (J_z)	Gaussian	10%
Pitch angle (ϑ)/deg	Uniform	$(-1, 1)$
Elevator deflection error	Gaussian	10%
Thrust error	Gaussian	10%

The ballistic trajectory of the Monte Carlo simulations is shown in Figure 18a, while Figure 18b illustrates the results of the miss distance and terminal impact-angle error. The results demonstrated that upon the introduction of the error model, the guidance performance of the IGCIAC-DRL strategy decreased in comparison to the other methods, having both the miss distance and the terminal angle error significantly reduced compared to the other methods, indicating the weaker robustness of the IGCIAC-DRL strategy.

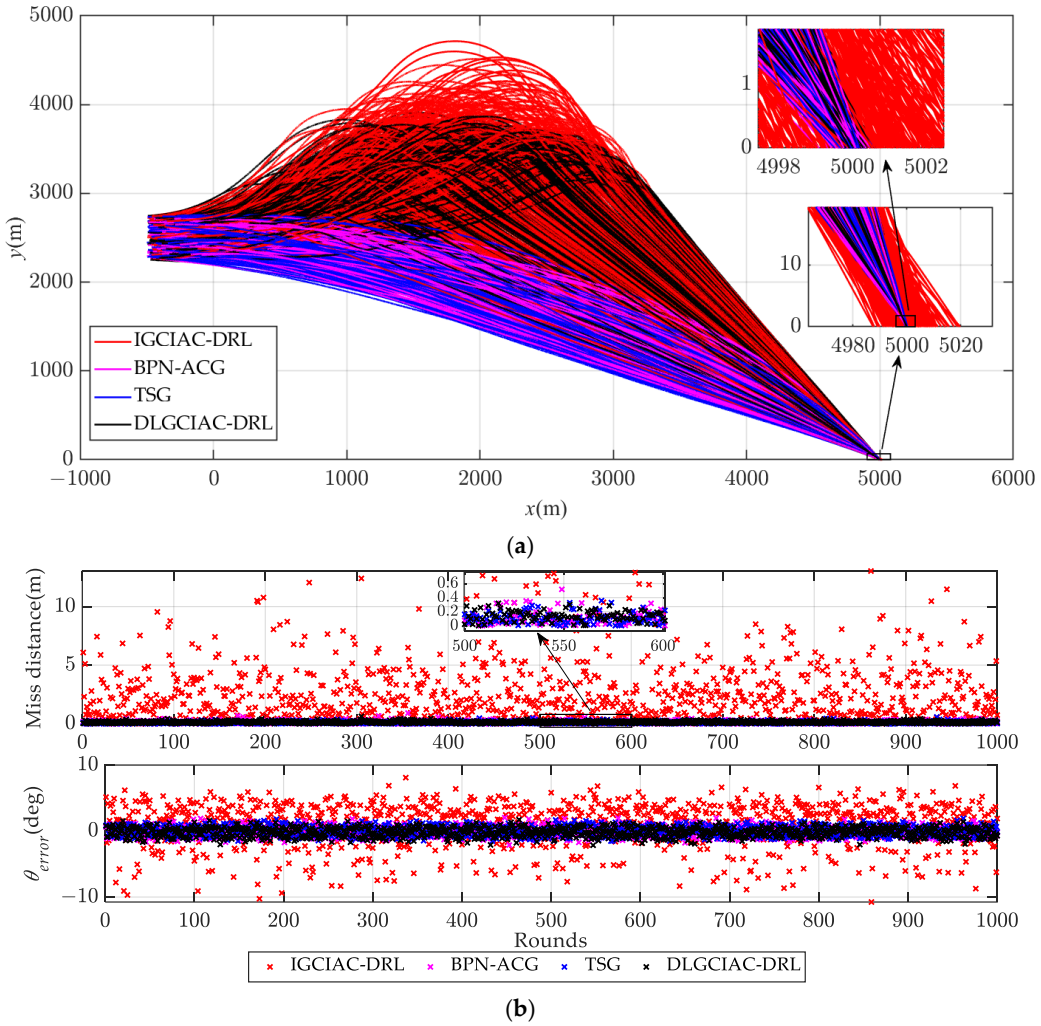


Figure 18. The Monte Carlo simulation results: (**a**) ballistic trajectory and (**b**) the miss distance and terminal impact-angle error.

Next, to analyze the guidance performance further, the Monte Carlo simulation process was performed using specific statistical data on the terminal impact-angle error θ_{error} and miss distance given in Table 8.

Table 8. The statistical results of the Monte Carlo experiment.

	θ_{error}				Miss Distance			
	DLGCIAC-DRL	**IGCIAC-DRL**	**TSG**	**BPN-ACG**	**DLGCIAC-DRL**	**IGCIAC-DRL**	**TSG**	**BPN-ACG**
Maximum value	1.7159	8.0935	1.9243	2.0290	0.5274	13.0343	0.5714	0.9058
Average value	−0.017	0.9456	0.0216	−0.103	0.1057	2.4751	0.1145	0.1281
Standard deviation	0.7149	3.3135	0.7418	0.7666	0.0775	2.1387	0.0808	0.103
Mean squared error	0.5384	11.8627	0.5500	0.5978	0.0161	10.6958	0.0174	0.027

The statistical results in Table 8 indicate that the DLGCIAC-DRL strategy had better robustness against interference compared to the other three strategies, which was reflected in its smaller values of both the maximum and average impact-angle error and miss distance in comparison to the other methods. For the purpose of better visualization of the results of the terminal impact-angle error and ballistic impact point distribution, a statistical histogram is displayed in Figure 19. The majority of terminal impact-angle errors of the DLGCIAC-DRL strategy were within a range from −1° to 1°, whereas the impact points of ballistic were mostly distributed between −0.25 m and 0.25 m, which demonstrates the high stability against disturbance and scalability and practicality of the proposed DLGCIAC-DRL strategy.

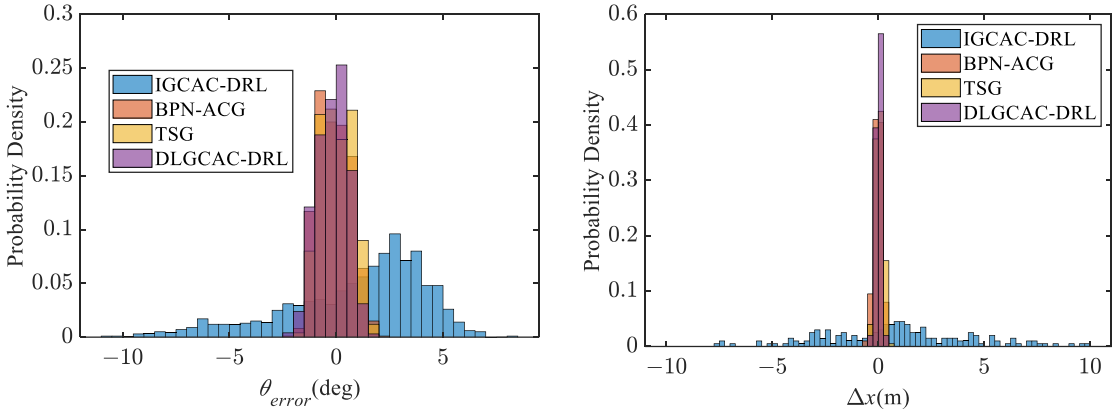


Figure 19. Statistical histogram of the terminal impact-angle error and ballistic impact point distribution: (**a**) terminal impact-angle error and (**b**) ballistic impact point distribution.

4.3. Computational Cost Analysis

The actor network generated by training can be deployed in embedded hardware as a real-time decision-maker for the flying object to generate guidance instructions in real-time. In each decision, each weight parameter in each neuron corresponds to a basic multiplication operation and an addition operation, each bias parameter corresponds to an addition operation, and each neuron performs an activation function solution operation. In addition, the output of the actor network in this paper includes the parameters a and b of the Beta distribution. After the training is completed, the mean of the Beta distribution can be used as the guidance command. The mean expression of the Beta distribution is:

$$B_{mean} = \frac{a}{a+b} \tag{33}$$

According to the actor network structure shown in Figure 9a, the number of various computational operations required to perform a decision is shown in Table 9.

Table 9. The number of calculation operations required to perform a decision.

Calculating Operation	Frequency
Multiplication operation	12,929
Addition operation	13,187
Activation function solving operation	258

Taking the commonly used STM32F407ZGT6 embedded platform as an example, its built-in floating-point arithmetic unit performs a basic floating-point addition or multiplication operation on a 32-bit single-precision floating-point number which generally requires 3~4 clock cycles. Each clock cycle is about 5.95 ns, and the time required to perform a basic operation on a single-precision floating-point number is about 17.85 ns~23.8 ns. Suppose that the operation time of solving the activation function is 10 times the operation time of performing the basic operation. Based on this, it can be estimated that the execution time of a decision is about 663,925 ns, about 0.66 ms. It can be seen that the proposed guidance control strategy based on RL can basically meet the real-time requirements of guidance tasks.

5. Conclusions

This paper proposes two DRL guidance and control strategies named DLGCIAC-DRL and IGCIAC-DRL. First, to ensure the high fidelity of the guidance and control model, it is imperative to obtain an RL-based model of the guidance and control system by incorporating comprehensive flying object dynamics and a control mechanism. Additionally, to enhance the model's applicability, the real-time impact-angle error is incorporated into the state vector. In addition, an improved reward mechanism is developed to decrease both the flying object–target distance and the impact-angle error, while alleviating the problem of sparse rewards in RL. The proposed mechanism facilitates the convergence of the reward function. The study also uses the Beta distribution instead of the Gaussian distribution to sample actions in the actor network. This substitution helps to alleviate the negative effect of unbounded distributions on the bounded action spaces. Finally, simulations are conducted in various scenarios, and a comparative analysis of the two strategies is performed. Additionally, a Monte Carlo experiment is designed to assess their robustness against interference.

The results indicate that the two proposed strategies can achieve impact-angle control, but that the DLGCIAC-DRL strategy exhibited a stronger anti-interference ability and greater applicability than the IGCIAC-DRL. They also show that the proposed impact-angle constraint guidance and control strategies based on DRL have engineering application value. In the future, the proposed method should be considered and extended to a three-dimensional scenario and then transplanted into an embedded OS to further verify its engineering practicability.

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

Funding: This work was supported in part by the National Key Research and Development Program (Grant No. 2020YFC1511705), the National Natural Science Foundation of China (Grant No. 52272358), the Project of Construction and Support for High-level Innovative Teams of Beijing Municipal Institutions (Grant No. BPHR20220123), and the Young Elite Scientist Sponsorship Program by BAST (Grant No. BYESS2023310).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors report no conflict of interest. All authors are responsible for the contents of and the writing of this article.

References

1. Lee, C.H.; Seo, M.G. New insights into guidance laws with terminal angle constraints. *J. Guid. Control. Dyn.* **2018**, *41*, 1832–1837. [CrossRef]
2. Tsalik, R.; Shima, T. Optimal guidance around circular trajectories for impact-angle interception. *J. Guid. Control. Dyn.* **2016**, *39*, 1278–1291. [CrossRef]
3. Park, B.G.; Kim, T.H.; Tahk, M.J. Range-to-go weighted optimal guidance with impact angle constraint and seeker's look angle limits. *IEEE Trans. Aerosp. Electron. Syst.* **2016**, *52*, 1241–1256. [CrossRef]
4. Kim, M.; Grider, K.V. Terminal guidance for impact attitude angle constrained flight trajectories. *IEEE Trans. Aerosp. Electron. Syst.* **1973**, *1*, 852–859. [CrossRef]
5. Zhang, Y.; Ma, G.; Wu, L. A biased proportional navigation guidance law with large impact-angle constraint and the time-to-go estimation. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2014**, *228*, 1725–1734. [CrossRef]
6. Erer, K.S.; Ozgoren, M.K. Control of impact-angle using biased proportional navigation. In Proceedings of the AIAA Guidance, Navigation, and Control (GNC) Conference, Boston, MA, USA, 19–22 August 2013.
7. Biswas, B.; Maity, A.; Kumar, S.R. Finite-time convergent three-dimensional nonlinear intercept angle guidance. *J. Guid. Control. Dyn.* **2020**, *43*, 146–153. [CrossRef]
8. Majumder, K.; Kumar, S.R. Finite-time convergent impact angle constrained sliding mode guidance. *IFAC-Pap.* **2020**, *53*, 87–92. [CrossRef]
9. Cho, H.; Ryoo, C.K.; Tsourdos, A.; White, B. Optimal impact angle control guidance law based on linearization about collision triangle. *J. Guid. Control. Dyn.* **2014**, *37*, 958–964. [CrossRef]
10. Lee, C.H.; Tahk, M.J.; Lee, J.I. Generalized formulation of weighted optimal guidance laws with impact angle constraint. *IEEE Trans. Aerosp. Electron. Syst.* **2013**, *49*, 1317–1322. [CrossRef]
11. Ibarz, J.; Tan, J.; Finn, C.; Kalakrishnan, M.; Pastor, P.; Levine, S. How to train your robot with deep reinforcement learning: Lessons we have learned. *Int. J. Robot. Res.* **2021**, *40*, 698–721. [CrossRef]
12. Piccinin, M.; Lavagna, M.R. Deep reinforcement learning approach for small bodies shape reconstruction enhancement. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020.
13. Graesser, L.; Keng, W.L. *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*; Pearson Education: London, UK, 2019.
14. Brandonisio, A.; Capra, L.; Lavagna, M. Spacecraft adaptive deep reinforcement learning guidance with input state uncertainties in relative motion scenario. In Proceedings of the AIAA SCITECH 2023 Forum, National Harbor, MD, USA, 23–27 January 2023.
15. LaFarge, N.B.; Miller, D.; Howell, K.C.; Linares, R. Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment. *Acta Astronaut.* **2021**, *186*, 1–23. [CrossRef]
16. Hovell, K.; Ulrich, S.; Bronz, M. Acceleration-based quadrotor guidance under time delays using deep reinforcement learning. In Proceedings of the AIAA Scitech 2021 Forum, Reston, VA, USA, 11–15 & 19–21 January 2021.
17. Hua, H.; Fang, Y. A novel reinforcement learning-based robust control strategy for a quadrotor. *IEEE Trans. Ind. Electron.* **2022**, *70*, 2812–2821. [CrossRef]
18. Zhou, W.; Li, J.; Liu, Z.; Shen, L. Improving multi-target cooperative tracking guidance for UAV swarms using multi-agent reinforcement learning. *Chin. J. Aeronaut.* **2022**, *35*, 100–112. [CrossRef]
19. Zhang, Q.; Ao, B.; Zhang, Q. Reinforcement learning guidance law of Q-learning. *Syst. Eng. Electron.* **2020**, *42*, 414–419.
20. Gaudet, B.; Furfaro, R.; Linares, R. Reinforcement learning for angle-only intercept guidance of maneuvering targets. *Aerosp. Sci. Technol.* **2020**, *99*, 105746. [CrossRef]
21. Wang, Z.; Li, H.; Wu, Z.; Wu, H. A pretrained proximal policy optimization algorithm with reward shaping for aircraft guidance to a moving destination in three-dimensional continuous space. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 1729881421989546. [CrossRef]
22. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
23. Qiu, X.; Gao, C.; Jing, W. Deep reinforcement learning guidance law for intercepting endo-atmospheric maneuvering targets. *J. Astronaut.* **2022**, *43*, 685–695.
24. Peng, C.; Zhang, H.; He, Y.; Ma, J. State-following-kernel-based online reinforcement learning guidance law against maneuvering target. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 5784–5797. [CrossRef]
25. Jiang, L.; Nan, Y.; Zhang, Y.; Li, Z. Anti-Interception Guidance for hypersonic glide vehicle: A deep reinforcement learning approach. *Aerospace* **2022**, *9*, 424. [CrossRef]
26. Hui, J.; Wang, R.; Guo, J. Research of intelligent guidance for no-fly zone avoidance based on reinforcement learning. *Acta Aeronaut. Astronaut. Sin.* **2023**, *44*, 240–252.
27. Luo, W.; Chen, L.; Liu, K.; Gu, H.; Lü, J. Optimizing constrained guidance policy with minimum overload regularization. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 2994–3005. [CrossRef]

28. Wang, W.; Wu, M.; Chen, Z.; Liu, X. Integrated Guidance-and-Control Design for Three-Dimensional Interception Based on Deep-Reinforcement Learning. *Aerospace* **2023**, *10*, 167. [CrossRef]
29. Liang, C.; Wang, W.; Liu, Z.; Lai, C.; Zhou, B. Learning to guide: Guidance law based on deep meta-learning and model predictive path integral control. *IEEE Access* **2019**, *7*, 47353–47365. [CrossRef]
30. Liang, C.; Wang, W.; Liu, Z.; Lai, C.; Wang, S. Range-aware impact-angle guidance law with deep reinforcement meta-learning. *IEEE Access* **2020**, *8*, 152093–152104. [CrossRef]
31. Li, B.; An, X.; Yang, X.; Wu, Y.; Li, G. A distributed reinforcement learning guidance method under impact-angle constraints. *J. Astronaut.* **2022**, *43*, 1061–1069.
32. Liu, Z.; Wang, J.; He, S.; Li, Y. A computational guidance algorithm for impact-angle control based on predictor-corrector concept. *Acta Aeronaut. Astronaut. Sin.* **2022**, *43*, 521–536.
33. Lee, S.; Lee, Y.; Kim, Y.; Han, Y.; Kwon, H.; Hong, D. Impact Angle Control Guidance Considering Seeker's Field-of-View Limit Based on Reinforcement Learning. *J. Guid. Control. Dyn.* **2023**, *46*, 2168–2182. [CrossRef]
34. Ji, Y.; Lin, D.; Pei, P.; Shi, X.; Wang, W. Robust partial integrated guidance and control approaches for maneuvering targets. *Int. J. Robust Nonlinear Control.* **2019**, *29*, 6522–6541. [CrossRef]
35. Wang, X.; Wang, J. Partial integrated guidance and control with impact angle constraints. *J. Guid. Control. Dyn.* **2015**, *38*, 925–936. [CrossRef]
36. Fang, J.; Lin, D.; Qi, Z.; Zhang, H. Design and analysis of a two-loop autopilot. *Syst. Eng. Electron.* **2008**, *30*, 2447–2450.