Article

High-Speed Three-Dimensional Aerial Vehicle Evasion Based on a Multi-Stage Dueling Deep Q-Network

Yefeng Yang [1,2,*,†], Tao Huang [1,2,†], Xinxin Wang [1,†], Chih-Yung Wen [2,†] and Xianlin Huang [1,†]

[1] Center for Control Theory and Guidance Technology, Harbin Institute of Technology, Harbin 150001, China
[2] Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, China
[*] Correspondence: yefeng.yang@connect.polyu.hk
[†] These authors contributed equally to this work.

Abstract: This paper proposes a multi-stage dueling deep Q-network (MS-DDQN) algorithm to address the high-speed aerial vehicle evasion problem. High-speed aerial vehicle pursuit and evasion are an ongoing game attracting significant research attention in the field of autonomous aerial vehicle decision making. However, traditional maneuvering methods are usually not applicable in high-speed scenarios. Independent of the aerial vehicle model, the implemented MS-DDQN-based method searches for an approximate optimal maneuvering policy by iteratively interacting with the environment. Furthermore, the multi-stage learning mechanism was introduced to improve the training data quality. Simulation experiments were conducted to compare the proposed method with several typical evasion maneuvering policies and to reveal the effectiveness and robustness of the proposed MS-DDQN algorithm.

Keywords: aerial vehicle evasion; deep reinforcement learning; dueling deep Q-network; multi-stage training

check for updates

Citation: Yang, Y.; Huang, T.; Wang, X.; Wen, C.-Y.; Huang, X. High-Speed Three-Dimensional Aerial Vehicle Evasion Based on a Multi-Stage Dueling Deep Q-Network. *Aerospace* **2022**, *9*, 673. https://doi.org/10.3390/aerospace9110673

Academic Editor: Domenico Accardo

Received: 23 September 2022
Accepted: 28 October 2022
Published: 31 October 2022

1. Introduction

The pursuit–evasion game is a challenging problem involving non-cooperative confrontation, which is currently an active research topic in aerial vehicle guidance, navigation, and control. In a pursuit–evasion problem, non-partners aerial vehicles have conflicts of interest while the trackers in the game are committed to adjusting their strategy to address a smaller miss. On the contrary, an evasion problem aims to find an optimal strategy for the tracked aerial vehicle to maximize the miss between the two. Most traditional aerial vehicle evasion strategies adopt typical evasion strategies to avoid the attack of the pursuit aerial vehicle [1], including sinusoidal [2], step [3], square [4], and spiral [5] maneuvers. Conventional maneuvering policies have proven effective at aircraft evasion when pursuers motorize with a low velocity and weak maneuverability [6]. Nevertheless, it is challenging to achieve a successful evasion when pursuers have strong maneuverability.

Differential game theory is particularly effective for solving problems with conflicting interests. Considering the one-pursuer one-evader (P1N1) scenario, transforming it into a two-player zero-sum game problem [6], the differential game attains the optimal evasion strategy for the evading aerial vehicle through iteratively solving the Hamilton function based on the confrontation model. Several optimal evasion strategies have been designed to employ differential game theory. In [7], the authors investigated the evasion differential game problem of an infinite-evader infinite-pursuer in the Hilbert Space l_2 and provided conditions for a successful escape. Liang et al. [8] discussed the game problem between an attacker and the target with an active defense function and utilized differential game theory to produce the evader's winning regions. A sufficient condition of an M-pursuer N-evader differential game problem was given by Ibragimov et al. [9]. Rilwan et al. [10] improved the dynamic function of the differential game problem and solved the evasion problem of the one-pursuer-one-evader in the Hilbert space. Based on nonlinear control and game theories, [11] proposed a state-dependent Riccati equation method in a two-player zero-sum differential game scenario. Asadi et al. [12] developed an analytical closed-form solution to deal with the vehicle-target assignment in a pursuit–evasion problem. Nevertheless, a differential game-based evading strategy has not been implemented in realistic scenarios due to low stability, model dependence, weak robustness, and poor adaptability.

With the fast development of computing science and deep learning (DL) theory, artificial intelligence (AI) technology has drawn increasing attention. As an essential branch of AI, deep reinforcement learning (DRL) pushes the field of AI to a higher level of autonomous systems understanding. Combining DL and reinforcement learning (RL), DRL establishes a direct connection between the perception and decision making of autonomous systems employing deep neural networks (DNNs). Unlike supervised learning, DRL iteratively learns the optimal policy by interacting with the environment instead of developing a regression model utilizing predefined labeled data [13]. The application of DRL technology is growing exponentially since the Alpha Go agent trained by Google DeepMind won the Go competition in 2016 [14]. DRL has several unique characteristics. First, DRL can generate an optimal policy for agents without recognizing the accurate mathematical model of both the environment and the agents. In particular, DRL shows significant performance when dealing with high model uncertainty and parameter fluctuation scenarios. Second, DRL shows adaptability in different environments since it actively interacts with the environment when generating a policy and affords a small exploration probability to guarantee sufficient environmental exploration even when the current policy is good enough.

Research on introducing DRL technology into pursuit–evasion problems has increased dramatically during the last five years. Sun et al. [15] introduced an adaptive dynamic programming (ADP) framework as an equation solver for differential games that successfully approximates the guidance law's optimal solution. Gaudet et al. [16] developed an RL-based method for the interception problem that only depends on the measurement of line-of-sight angles and angular rates. Zhu et al. [17] proposed an aerial vehicle evasion strategy based on DQN, while Li et al. [18] introduced a deep deterministic policy gradient (DDPG) framework that considers small vehicle evasion strategies. Shalumov et al. [19] presented an optimal launch time and a guidance law via DRL in a target–missile–defender scenario. The work of Souza et al. [20] proposed a DRL method for decentralized multi-agent pursuit problems. In [21], Tipaldi et al. provided a detailed literature review of RL-based methods for aircraft pursuit and evasion.

Nevertheless, only a few methods for solving aircraft pursuit–evasion problems have been specifically designed for aircraft evasion, leading to aircraft evasion technology not meeting the current practical application requirements. Therefore, this paper proposes a multi-stage dueling DQN (MS-DDQN) algorithm to address the evasion problem of a vehicle equipped with ignition pulse engines. Furthermore, the integration of multi-stage learning technology significantly speeds up the training process. The simulation results indicate that the evasion strategy generated by the proposed MS-DDQN can maximize the miss and escape from the pursuer successfully.

The main contributions of this paper are the following:

- An MS-DDQN algorithm is proposed to improve the typical DQN algorithm to accelerate the convergence process and modify the quality of training data;
- An adaptive iterative learning framework is implemented for high-speed aerial vehicle evasion problems;
- Adequate comparative simulation experiments are implemented to verify the effectiveness and robustness of the proposed MS-DDQN algorithm.

The remainder of this paper is organized as follows: Section 2 presents the fundamental and problem formation. Section 3 introduces the MS-DDQN model and proposes the learning process to obtain the optimal evasion strategy. Section 4 presents the simulation experiments, and finally, Section 5 concludes this work.

2. Fundamental and Problem Formulation

2.1. Aerial Vehicle Pursuit–Evasion Model

This section proposes the pursuit–evasion model of two vehicles to provide an interactive simulation environment for the problem.

It is a one-pursuer one-evader scenario where two agents are moving toward one another, whose projection distance in the X-axis of the velocity–turn–climb coordinate system is $r_0 = 150$ km. The pursuer uses a proportional navigation guidance (PNG) law to attack the evader with a maneuver period of $T_P = 15$ ms and the maximum acceleration is $a_{maxP} = 4.5$ g, where $g = 9.8\ \mathrm{m/s^2}$. The evader has four typical ignition pulse engines with a maneuver period of $T_E = 50$ ms to avoid the pursuer's attack with a maximum acceleration of $a_{maxE} = 7.5$ g. The orientation of the acceleration generated by the engines is perpendicular to the direction of the body. To make our testing scenario more challenging, we chose a larger velocity. The pursuer's velocity was $V_P = 1.875$ km/s, and the evader's velocity was $V_E = 4.5$ km/s. In this scenario, the evader is the agent that needs to be trained, while the pursuer is part of the simulation environment for the DRL problem described in this scenario. The agent learns the optimal evasion strategy by interacting with the pursuer to maximize the minimal miss, thoroughly discussed in Section 3.

The inertial reference frame, velocity–turn–climb coordinate frame, and line-of-sight (LOS) coordinate frame are denoted as $OXYZ(S_0)$, $O_2X_2Y_2Z_2(S_2)$, and $O_4X_4Y_4Z_4(S_4)$, respectively. For simplicity, the pursuer is denoted as P and the evader as E. Let $\theta_i|_{i=P,E}$ be the flight path angle and $\psi_{vi}|_{i=P,E}$ the heading angle. q_ϵ and q_β are the horizontal and vertical LOS angles, and the horizontal and vertical LOS rotational rates are represented by $\dot{q}_\epsilon$ and $\dot{q}_\beta$, respectively. As presented in Equations (1) and (2), $\boldsymbol{C_{2-0}}$ and $\boldsymbol{C_{4-0}}$ are the transformation matrices from S_2 to S_0 and from S_4 to S_0, respectively.

$$\boldsymbol{C_{2-0}} = \left.\begin{bmatrix} \cos\theta_i \cos\psi_{vi} & -\sin\theta_i \sin\psi_{vi} & \sin\psi_{vi} \\ \sin\theta_i & \cos\theta_i & 0 \\ -\cos\theta_i \sin\psi_{vi} & \sin\theta_i \sin\psi_{vi} & \cos\psi_{vi} \end{bmatrix}\right|_{i=P,E}, \tag{1}$$

$$\boldsymbol{C_{4-0}} = \begin{bmatrix} \cos q_\epsilon \cos q_\beta & -\sin q_\epsilon \sin q_\beta & \sin q_\beta \\ \sin q_\epsilon & \cos q_\epsilon & 0 \\ -\cos q_\epsilon \sin q_\beta & \sin q_\epsilon \sin q_\beta & \cos q_\beta \end{bmatrix}. \tag{2}$$

Figure 1 Left illustrates the conversion relationship between S_0 and S_2, where θ is the flight path angle and ψ_v is the heading angle. Figure 1 Right depicts the relationship between S_0 and S_4 defined by the LOS angles q_ϵ and q_β.

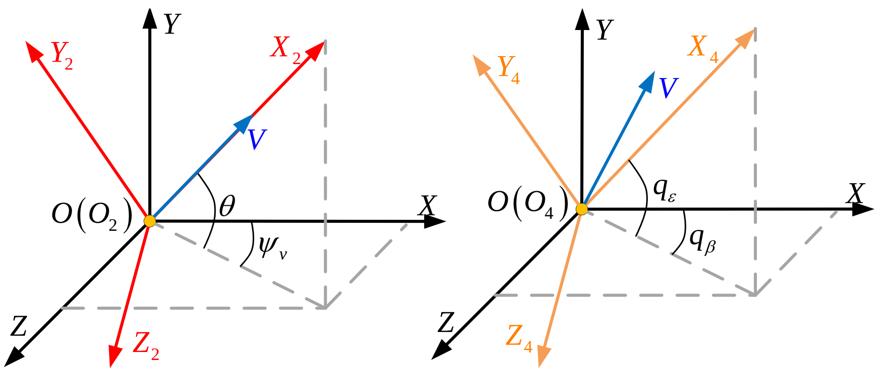

Figure 1. Conversion relationship between S_0, S_2, and S_4 (**Left**: S_0 and S_2. **Right**: S_0 and S_4). V is the velocity vector of the aircraft, θ is the flight path angle, ψ_v is the heading angle, and q_ϵ and q_β are the LOS angles.

Let $\boldsymbol{X_i} = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}|_{i=P,E}$ be the agent's location in S_0 and let $\boldsymbol{V_{2i}} = \begin{bmatrix} V_i & 0 & 0 \end{bmatrix}|_{i=P,E}$ be the vehicle's velocity in S_2. Then, we have:

$$\boldsymbol{V_{0i}} = \boldsymbol{C_{2-0}}\boldsymbol{V_{2i}} = \begin{bmatrix} v_{0xi} & v_{0yi} & v_{0zi} \end{bmatrix}|_{i=P,E}, \tag{3}$$

where $\boldsymbol{V_{0i}}, i = P, E$ is the vehicle's velocity in S_0. The location's derivative can be depicted as:

$$\begin{cases} \dot{x}_i = V_i \cos\theta_i \cos\psi_{vi}|_{i=P,E} \\ \dot{y}_i = V_i \cos\theta_i|_{i=P,E} \\ \dot{z}_i = -V_i \cos\theta_i \sin\psi_{vi}|_{i=P,E} \end{cases}. \tag{4}$$

Let $\boldsymbol{a_{4i}} = \begin{bmatrix} 0 & a_{iy} & a_{iz} \end{bmatrix}|_{i=P,E}$ be the aerial vehicle's acceleration in S_4; then, we have:

$$\boldsymbol{a_{0i}} = \boldsymbol{C_{4-0}}\boldsymbol{a_{4i}}|_{i=P,E}, \tag{5}$$

where $\boldsymbol{a_{0i}}|_{i=P,E}$ is the vehicle's acceleration in S_0. $\dot{\boldsymbol{V}}_{0i}$ is also reformulated as:

$$\begin{cases} \dot{v}_{0xi} = V_i \cos q_\epsilon \cos q_\beta|_{i=P,E} \\ \dot{v}_{0yi} = V_i \cos q_\epsilon|_{i=P,E} \\ \dot{v}_{0zi} = -V_i \cos q_\epsilon \sin q_\beta|_{i=P,E} \end{cases}. \tag{6}$$

Equations (7)–(10) are the formulas of q_ϵ, q_β, $\dot{q}_\epsilon$, and $\dot{q}_\beta$ in S_4, respectively.

$$q_\epsilon = \arctan\frac{r_y}{\sqrt{r_x^2 + r_z^2}}, \tag{7}$$

$$q_\beta = -\arctan\frac{r_z}{r_x}, \tag{8}$$

$$\dot{q}_\epsilon = \frac{(r_x^2 + r_z^2)\dot{r}_y - r_y(r_x\dot{r}_x + r_z\dot{r}_z)}{\left(r_x^2 + r_y^2 + r_z^2\right)\sqrt{r_x^2 + r_z^2}}, \tag{9}$$

$$\dot{q}_\beta = \frac{r_z\dot{r}_x - r_x\dot{r}_z}{r_x^2 + r_z^2}, \tag{10}$$

where $r_x = x_E - x_P$, $r_y = y_E - y_P$, $r_z = z_E - z_P$, $\dot{r}_x = \dot{x}_E - \dot{x}_P$, $\dot{r}_y = \dot{y}_E - \dot{y}_P$, and $\dot{r}_z = \dot{z}_E - \dot{z}_P$.

According to the basic principle of PNG, the pursuer's acceleration of $\boldsymbol{a_{4P}}$ in S_4 is given by:

$$\begin{cases} \boldsymbol{a_{4P}} = \begin{bmatrix} 0 & N_{4y}v\dot{q}_\epsilon & N_{4z}v\dot{q}_\beta \end{bmatrix} \\ r = \sqrt{r_x^2 + r_y^2 + r_z^2} \\ vs. = \dfrac{r_x\dot{r}_x + r_y\dot{r}_y + r_z\dot{r}_z}{r} \end{cases}, \tag{11}$$

where $N_{4y} = 4$ and $N_{4z} = 5$ are the PNG's navigation ratios. Then, the aerial vehicle pursuit-evasion model can be defined by combining Equations (3)–(11)

$$\begin{cases} \boldsymbol{X_i} = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix} |_{i=P,E} \\ \dot{x}_i = V_i \cos\theta_i \cos\psi_{vi} |_{i=P,E} \\ \dot{y}_i = V_i \cos\theta_i |_{i=P,E} \\ \dot{z}_i = -V_i \cos\theta_i \sin\psi_{vi} |_{i=P,E} \\ \boldsymbol{V_{0i}} = \begin{bmatrix} v_{0xi} & v_{0yi} & v_{0zi} \end{bmatrix} |_{i=P,E} \\ \boldsymbol{a_{0i}} = \boldsymbol{C_{4-0}} \boldsymbol{a_{4i}} |_{i=P,E} \\ \boldsymbol{a_{4P}} = \begin{bmatrix} 0 & N_{4y} v \dot{q}_\epsilon & N_{4z} v \dot{q}_\beta \end{bmatrix} \\ \boldsymbol{a_{4E}} = \begin{bmatrix} 0 & a_{4Ey} & a_{4Ez} \end{bmatrix} \end{cases}, \tag{12}$$

where a_{4ey} and a_{4ez} are the evader's accelerations in S_4.

2.2. Value-Based Reinforcement Learning

An RL problem can be uniquely defined by a Markov decision process (MDP) [22] tuple $\langle \mathcal{S}, P, \mathcal{R}, \gamma, \mathcal{A} \rangle$, where $\mathcal{S}$ is the state space of the agent, P is the state transition probability matrix, $\mathcal{R}$ is the reward function, γ is the discount factor of the cumulative reward, and $\mathcal{A}$ is the action space. MDP is the fundamental framework of RL, where each episode starts with the initial condition and ends with the termination condition. When the agent takes action, it gets an immediate reward R_t and transfers to another state s_{t+1}. An RL problem intends to search for an optimal policy for a given MDP to maximize the cumulative reward G_t, introduced in Equation (13), and the optimal policy π^* is a distribution of an action space $\mathcal{A}$ for a fixed state s, introduced in Equation (14).

$$G_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} = r_{t+1} + \gamma G_{t+1}, \tag{13}$$

$$\pi^*(a|s) = P[A_t = a | S_t = s]. \tag{14}$$

In this aerial vehicle evasion problem, the state is the line-of-sight angles (q_ε and q_β) and the distance (r) of the two aerial vehicles, while the action is the operational status of the four pulse engines on or off. The designs of γ and $\mathcal{R}$ are implemented in Section 3.3, and the termination conditions of this problem are as follows:

- The evader is successfully intercepted by the pursuer, which is called failure evasion;
- The evader escapes from the attack of the pursuer successfully, which is called a successful evasion.

The RL algorithms comprise value-based [23], policy-based [24], and actor-critic [25]. For a value-based discrete time MDP (DT-MDP), to maximize the cumulative reward G_t, the agent chooses an action according to the state-value function (Q-Function; Equation (15)) to iteratively update the policy until π converges to the optimal solution π^*.

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_\pi[G_t | S_t = s_t, A_t = a] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{T} \gamma^k r^{t+k+1} | S_t = s_t, A_t = a\right]. \end{aligned} \tag{15}$$

More specifically, a neural network is chosen as a value function approximator of Equation (15). In the optimal scenario, the agent chooses a greedy policy according to π^*:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi^*}(s, a, \omega), \tag{16}$$

where ω is the parameter of the neural network.

3. MS-DDQN Algorithm Design

This section introduces the basic structure of the dueling DQN framework, proposes the MS-DDQN algorithm, and provides details on the learning process.

3.1. Dueling DQN Framework

The MS-DDQN framework is a multi-stage value-based DRL method, which improves the traditional DQN. The DQN establishes a direct connection between perception and the decision-making level of an autonomous system [26], as illustrated in Figure 2, where the replay memory collects data generated from the interaction between the agent (Evaluation Net) and the environment.

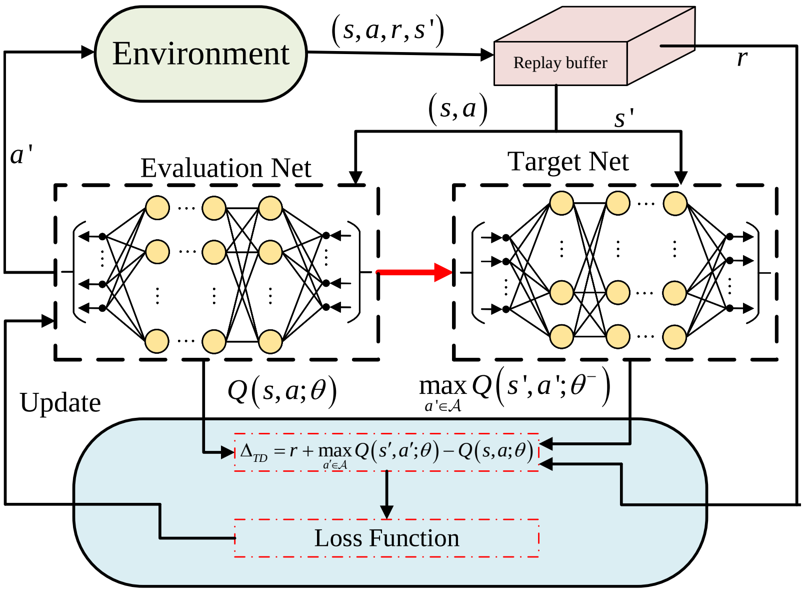
Environment
(s,a,r,s')
Replay buffer
r
(s,a)
s'
a'
Evaluation Net
Target Net
Update
$Q(s,a;\theta)$
$\max_{a'\in\mathcal{A}} Q(s',a';\theta^-)$
$\Delta_{TD} = r + \max_{a'\in\mathcal{A}} Q(s',a';\theta) - Q(s,a;\theta)$
Loss Function

Figure 2. Schematic diagram of DQN.

The sign of the convergence of the algorithm TD error converges to 0, and the TD error is calculated in the loss function module employing the formula $\Delta_{TD} = r + \max_{a'\in\mathcal{A}} Q(s',a';\theta) - Q(s,a;\theta)$. The θ parameters in the evaluation net are updated by the loss function utilizing a suitable gradient descent algorithm. Moreover, the θ^- parameters in the target net are copied from θ every C step, where C is a constant. The learning process ends when $\Delta_{TD} \to 0$.

However, DQN overfits the neural network approximator, overestimating the state's value function. To overcome this drawback, Wang et al. [27] proposed a dueling network architecture that weakens the Q-network's overestimation. Specifically, the dueling Q-network defines an advantage function to decouple the state-value function and the state-action-value function, as shown in Equation (17):

$$Q(\mathcal{S},\mathcal{A},\omega,\alpha,\beta) = V(\mathcal{S},\omega,\alpha) + A(\mathcal{S},\mathcal{A},\omega,\beta), \tag{17}$$

where ω, α, and β are the parameters that need to be optimized. Specifically, α is the parameter of the value function $V(\mathcal{S},\omega,\alpha)$, β is the parameter of the advantage function $A(\mathcal{S},\mathcal{A},\omega,\beta)$, and ω is the parameter that two functions have in common. Equation (17) is often written in the form of Equation (18) to reflect the identifiability of simulated physical experiments.

$$Q(\mathcal{S},\mathcal{A},\omega,\alpha,\beta) = V(\mathcal{S},\omega,\alpha) + \left[A(\mathcal{S},\mathcal{A},\omega,\beta) - \frac{1}{\mathcal{A}} \sum_{a' \in \mathcal{A}} A(\mathcal{S},a',\omega,\beta) \right]. \tag{18}$$

Figure 3 illustrates the difference between DQN and dueling DQN. Both networks involve m hidden layers and the same input–output dimension. The last hidden layer in the bottom figure of Figure 3 is divided into two parts: $V(\mathcal{S},\omega,\alpha)$ and $A(\mathcal{S},\mathcal{A},\omega,\beta)$, while the dueling DQN output is the sum of $V(\cdot)$ and $A(\cdot)$.

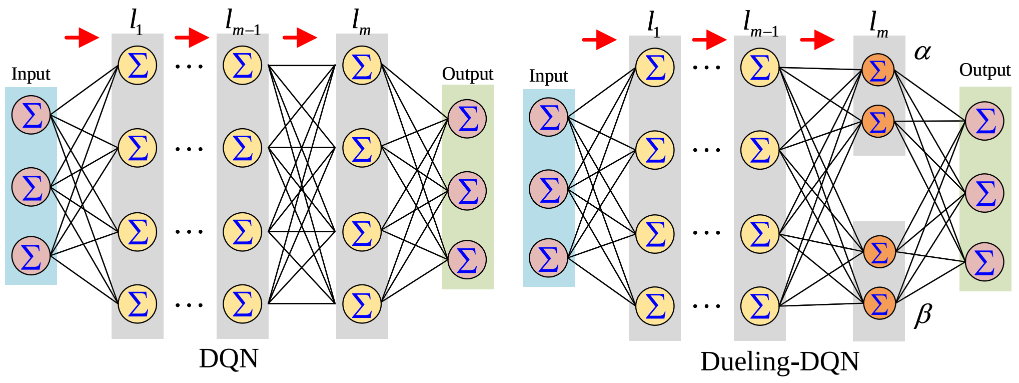

Figure 3. Difference between the networks in DQN (**left**) and dueling DQN (**right**).

Figure 4 illustrates the complete learning process. In Figure 4, *ODE* is the model state-solving module for state update, *CF* is the coordinate transformation module, and *PG* is the proportional guide law module that generates the acceleration command for the pursuer. The "policy learning" module in Figure 4 is the RL learning process shown in Figure 2. The "RL maneuver" is the training result of "policy learning", which is used to generate acceleration commands for the evader. The entire framework is a recurrent process and terminates when the evader can get rid of the pursuer's attack.

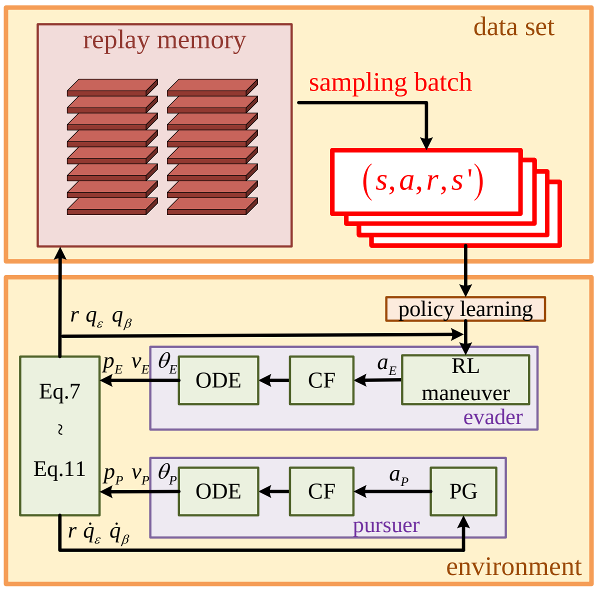

Figure 4. Complete learning framework.

3.2. Multi-Stage Learning

DRL methods iteratively learn the optimal policy through interacting with the environment. Nonetheless, it is almost impossible for an agent to collect adequate high-quality sampling data with an initial stochastic policy when the state space is vast. Therefore, a multi-stage learning scheme is proposed to speed up the agent's learning process by inserting several sub-mission nodes during the entire learning process to guide the agent step-by-step in reasonably learning the optimal policy. This idea is inspired by humans' solutions to complex problems, as we typically decompose a considerable problem into several smaller ones that are serially connected. The solution of a sub-task is the initial condition of the next one.

The entire task is divided into two sub-tasks for the evasion problem examined in this paper. In the first training phase, the agent learns an applicable policy π_{temp} with a certain stochastic exploration probability. π_{temp} is a relatively ideal initial condition for second-stage training, although not the optimal policy with a high probability. The objective of second-stage training is to find an optimal policy $\pi^\star$ that maximizes the miss based on π_{temp} by using the adequate high-quality data generated by the initial acceptable policy in the first stage of learning. Algorithm 1 presents the pseudo-code of this multi-stage training framework.

Algorithm 1 A multi-stage training framework with N sub-missions.

```
    Input: $N$, Reward function for each phase $f_i, i = 1, 2, \cdots, N$, an initial policy $\pi_0$.
 1: Output: The optimal policy $\pi^\star$
 2: Environment initialization: $i = 0$, empty the replay memory.
 3: while $i < N$ do
 4:     Initialize the neural network.
 5:     Fill the replay memory with sampling data generated by policy $\pi_i, \varepsilon_i$. ($\varepsilon_i$ is a certain
    initial exploration probability for the $ith$ phase.)
 6:     while $\pi_i$ is not converged do
 7:         Network training in Dueling-DQN framework with reward function $f_i$.
 8:         Update $\pi_i$ and replay memory.
 9:     Save $\pi_i$, $i+ = 1$.
10: return $\pi^\star$
```

3.3. Complete Learning Framework

The MS-DDQN method has four key elements: State space, action space, reward functions for each training phase, and the neural network structure. The choice of state space predominantly affects the quality of the final policy. In the vehicle evasion scenario, the distance, and LOS angles (q_ϵ and q_β) are the three crucial factors, and Equation (19) denotes the agent's state:

$$s = \left[r, q_\epsilon, q_\beta\right], \tag{19}$$

where r is the distance between the two vehicles and q_ϵ and q_β are the LOS angles. The evader in this scenario maneuvers with four ignition pulse engines, hence the evader has nine maneuvering types (Table 1); the action's dimension is one, with the agent selecting one of the nine actions according to the policy each time.

Table 1. Action space of the evader.

Action	Acceleration	Action	Acceleration	Action	Acceleration
0	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}$	3	$\begin{bmatrix}0 & 0 & +max\end{bmatrix}$	6	$\begin{bmatrix}0 & +max & -max\end{bmatrix}$
1	$\begin{bmatrix}0 & +max & 0\end{bmatrix}$	4	$\begin{bmatrix}0 & 0 & -max\end{bmatrix}$	7	$\begin{bmatrix}0 & -max & +max\end{bmatrix}$
2	$\begin{bmatrix}0 & -max & 0\end{bmatrix}$	5	$\begin{bmatrix}0 & +max & +max\end{bmatrix}$	8	$\begin{bmatrix}0 & -max & -max\end{bmatrix}$

The three elements in each action in Table 1 are the evader's acceleration in X, Y, and Z. '$+max$' means the acceleration reaches the maximum value, and '$-max$' is the minimum value (reverse maximum).

The choice of the reward function is another decisive factor in the training process. It should be a continuous function of the state, with the reward function of the first and second training phases given in Equations (20) and (21), respectively.

$$\begin{cases} r_1 = k_1 \max\left(|q_\epsilon|, |q_\beta|\right)^2 \\ r_2 = \begin{cases} 10000 & if\ evasion\ succeeded \\ 0 & if\ evasion\ failed \end{cases} \\ r = r_1 + r_2 \\ k_1 = \begin{cases} 200 \tan 0.1047t - 1.5184 & if\ action = 0,1,2,3,4 \\ 50 \tan 0.1047t - 1.5184 & if\ action = 5,6,7,8 \end{cases} \end{cases}, \tag{20}$$

$$\begin{cases} r_1 = k_1 \max\left(|q_\epsilon|, |q_\beta|\right)^2 \\ r_2 = \begin{cases} 1000r^2 & if\ evasion\ succeeded \\ 0 & if\ evasion\ failed \end{cases} \\ r = r_1 + r_2 \\ k_1 = \begin{cases} 200 \tan 0.1047t - 1.5184 & if\ action = 0,1,2,3,4 \\ 50 \tan 0.1047t - 1.5184 & if\ action = 5,6,7,8 \end{cases} \end{cases}. \tag{21}$$

In Equations (20) and (21), the reward function is divided into two parts: The reward for the approaching process r_1 and the reward for the episode termination r_2. k_1 is the coefficient of r_1, which is originally designed to affect the sensitivity of r_1 to the LOS angles and increases with time. The design of k_1 formation guides the agent to consume the remaining fuel at the end of the escape to increase its line of sight with the pursuer. The introduction of the tangent function ensures that the reward is more sensitive to the agent's action as time increases. The purpose of r_1 is to maximize one of the LOS angles, considering that the Y-direction is symmetrical to the Z-direction and r_2 is different in the two training phases. Therefore, the aim is to find, for the aerial vehicle, an acceptable policy in the first training phase and to maximize the miss in the second training phase.

Additionally, r_1 still exists in the second learning stage to avoid sparse rewards in RL. A sparse reward problem, i.e., the agent can get a nonzero reward if, and only if, it completes an episode of exploration, means that the agent does not have adequate data to modify its action [28]. It is hard for RL algorithms to converge when dealing with sparse reward problems. Therefore, both r_1 and r_2 are added to the reward function to ensure the adequacy of the training data.

The MS-DDQN estimates the policy's value function through a Q-network. Table 2 presents the neural network structure, where $3V$ and $3A$ are the value and advantage parts of the Q function, respectively, and $ReLu()$ is the activation function of the neural network.

Table 2. Structure of the neural network.

Layer	Input	Output	Activation Function
1	3	20	$ReLu()$
2	20	20	
3V	20	9	
3A	20	9	

Figure 5 illustrates the flow diagram of the training process. First, the flow chart presents the left channel to start first-phase training. Then, it continues to perform the second training phase after an acceptable policy is obtained, which is the initial policy of the second phase. The capacities of the two-training phase are hyper-parameters that need to be tuned manually. Finally, the algorithm stops when the target network converges. Furthermore, this paper adopted the network-retraining technique proposed in [29–31] to strengthen the network training quality.

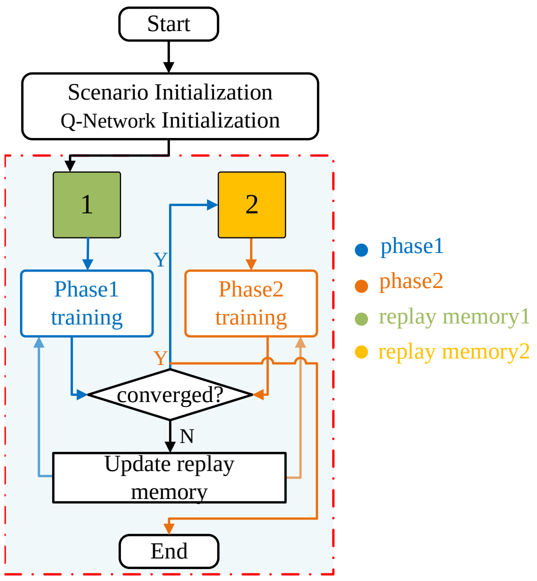

Figure 5. Flow diagram of the MS-DDQN training process.

The network training consists of procedures, initialization, and training. The data input into the replay memories during the initialization part are generated by the current temporary optimal policy with certain exploration probabilities $\epsilon_1 = 0.2$ and $\epsilon_2 = 0.2$. For the first stage of the training part, the optimal temporary policy is stochastic. In contrast, the second stage of the training part takes the policy generated by the first stage as the initial temporary optimal policy. Figure 6 depicts the relationship between the episode and exploration probability during the training process. The functional relationship between the exploration probability and episode is designed as an exponential function to make the agent's exploration more adequate. The mathematical function of each segment in Figure 6 is presented at the right top of the figure. Episodes 0–1499 belong to the first training of the first phase, while episodes 1500–2099 are the second training of the first phase, and episodes 2100–2999 belong to the training in the second phase.

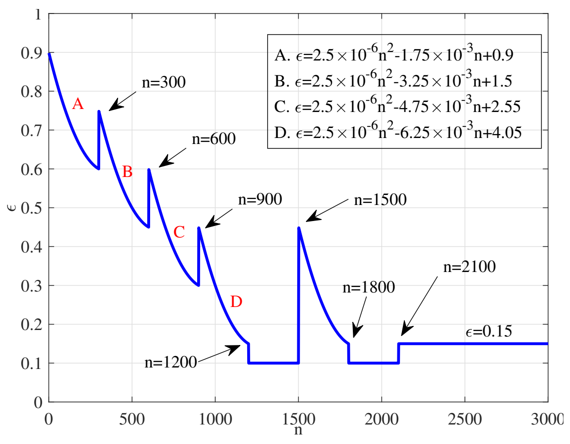

Figure 6. Relationship between episode n and exploration probability ϵ.

Table 3 lists the hyper-parameters of MS-DDQN, where γ denotes the discount factor of RL, ϵ_0 is the initial exploration probability, α is the learning rate of the Q-network, M is the capacity of the replay memory, and *Batch Size* is the number of samples extracted each time during the Q-network learning.

Table 3. Some hyper-parameters of MS-DDQN.

Parameter \ Phase	Phase1-a	Phase1-b	Phase2
γ	0.9	0.9	0.9
ϵ_0	0.9	0.9	0.9
α	0.01	0.01	0.01
M	5000	5000	2000
Batch Size	64	64	64

4. Simulation Experiments

This section evaluates the evasion performance employing the strategy generated by MS-DDQN. For clarity, we rearranged the parameters presented in Section 2.1 into Table 4.

Table 4. Parameters of the simulation scenario.

Parameter	Value	Parameter	Value
r_0	150 km	g	9.8 m/s^2
T_P	15 ms	T_E	50 ms
a_{maxP}	4.5 g	a_{maxE}	7.5 g
V_P	1.875 km/s	V_E	4.5 km/s

Here, r_0 is the initial relative distance in the LOS coordinate frame (S_4) of the two vehicles, $T_i\,|_{i=P,E}$ is the maneuver cycle of the vehicle, $a_{maxi}\,|_{i=P,E}$ is the maximum acceleration of the vehicle in the LOS coordinate frame (S_4), and $V_i\,|_{i=P,E}$ is the velocity of the vehicle in LOS coordinate frame (S_4).

Figure 7 represents the relationship between the cumulative reward and episode in both multi- and single-phase training. Simultaneously, the curves verify the effectiveness of the multi-phase training. Figure 7 highlights that the average reward increased significantly as the episode increased in phase1 training. However, the average reward decreased slightly in episodes 1500–2100 due to the increase in ϵ (see Figure 6), which means the agent explored the environment stochastically with a higher probability. Based on phase1, the average reward of phase2 was substantially higher than that of phase1, indicating a successful evasion and the effectiveness of the two-stage training technique implemented in the training process. Finally, the evader achieved a high-quality policy after 2700 training episodes. However, without multi-phase training, the single-phase training process maintained a low-level cumulative reward throughout the entire training process.

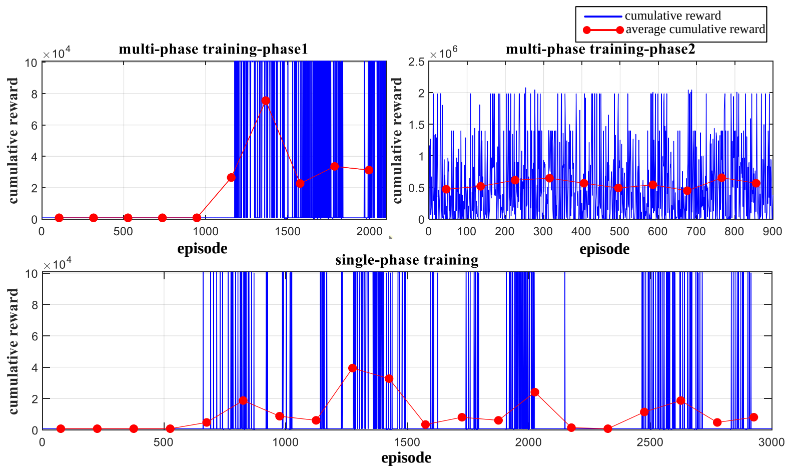

Figure 7. Comparative cumulative rewards of single- and multi-phase training.

Comparative simulation experiments were designed on the miss distances of the evader using various maneuvers. Figures 8 and 9 illustrate the simulation results when the evader implemented a random maneuvering policy and a square wave maneuver policy, respectively.

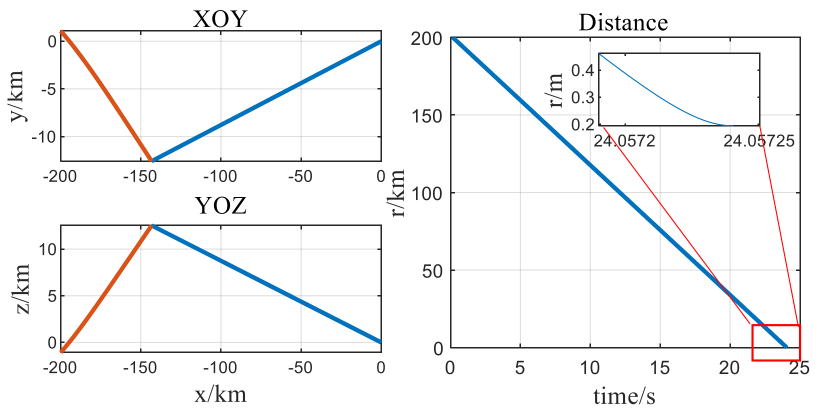

Figure 8. Evader employing a random maneuver.

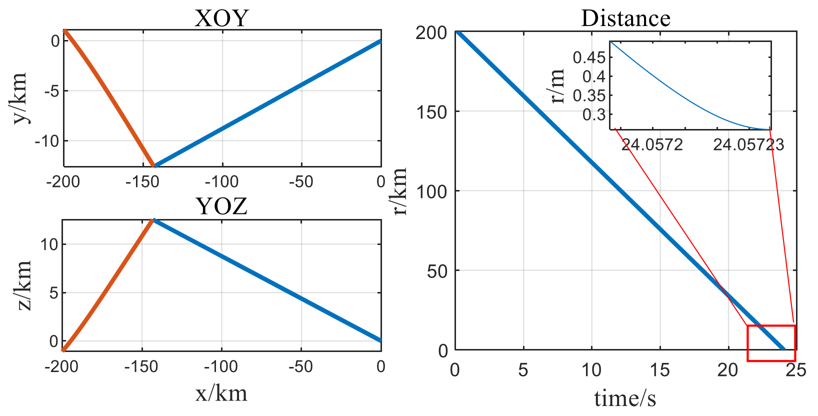

Figure 9. Evader employing a square maneuver.

The misses were 0.20 m and 0.26 m, respectively, and the simulation time is 24.057 s. The results indicate that the evader could not use traditional maneuvers to avoid the pursuer's attack. Figures 10–12 present the simulation results when the evader applied a neural network maneuver strategy generated by MS-DDQN. The scenarios involved nine relative locations with an initial flight path angle θ and a heading angle of 0°. We conclude that the miss is much more significant than the traditional maneuver.

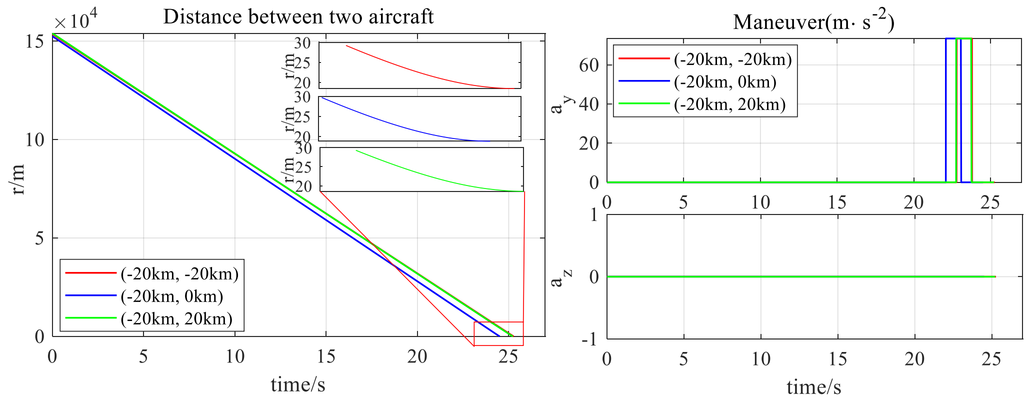

Figure 10. Evader utilizing MS-DDQN maneuvers with the three different initial relative positions $(-20\,\text{km}, -20\,\text{km})$, $(-20\,\text{km}, 0\,\text{km})$, and $(-20\,\text{km}, 20\,\text{km})$ in the initial inertial coordinate system.

Figure 10 presents the evasion process when the evader implemented the policy generated by MS-DDQN with the initial relative positions $(-20\,\text{km}, -20\,\text{km})$, $(-20\,\text{km}, 0\,\text{km})$, and $(-20\,\text{km}, 20\,\text{km})$ in the initial inertial coordinate system. The results indicate that the slightest miss of the three scenarios was 18.4174 m, satisfying the problem requirement. Additionally, the figures for $time - a_y$ and $time - a_z$ illustrate that the evader tended to consume all of the remaining fuel at the end of the interception to increase its line of sight with the pursuer, which can be found in the figure for $time - a_y$. Allocating all fuel in the Y-direction, the output of the Z-direction remained zero throughout the process due to the limited fuel. Figures 11 and 12 indicate the results from the same law as Figure 10.

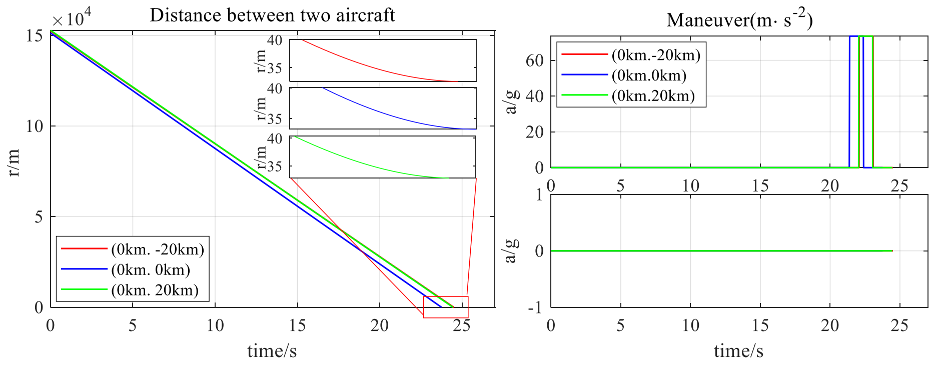

Figure 11. Evader utilizing MS-DDQN maneuvers with the three different initial relative positions $(0\,\text{km}, -20\,\text{km})$, $(0\,\text{km}, 0\,\text{km})$, and $(20\,\text{km}, 20\,\text{km})$ in the initial inertial coordinate system.

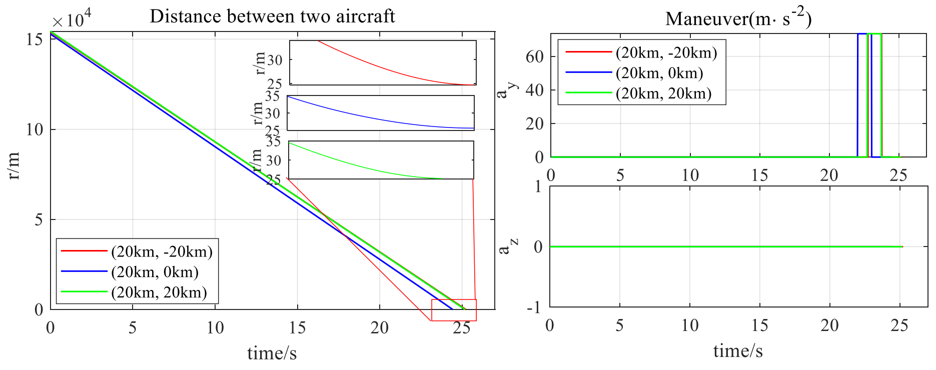

Figure 12. Evader utilizing MS-DDQN maneuvers with the three different initial relative positions $(20\,\text{km}, -20\,\text{km})$, $(20\,\text{km}, 0\,\text{km})$, and $(20\,\text{km}, 20\,\text{km})$ in the initial inertial coordinate system.

To further demonstrate the feasibility of the MS-DDQN maneuver, more simulation experiments with different relative initial positions, flight path angles θ, and heading angles Ψ_v are presented. Tables 5–9 present the simulation results with an initial θ of 10°, 5°, 0°, −5°, and −10°, respectively.

Table 5. Miss distance (m) with initial flight path angle $\theta = 10°$.

Initial Relative Distance (km)	**Heading Angle Ψ_v (°)**				
	10	**5**	**0**	**−5**	**−10**
$(20, 20)$	19.81	23.25	25.17	25.36	25.45
$(20, 0)$	23.87	25.41	25.61	25.43	24.21
$(20, -20)$	25.69	25.53	24.90	23.12	18.80
$(0, 20)$	24.11	28.97	29.90	30.50	30.69
$(0, 0)$	30.29	30.71	30.75	30.48	29.71
$(0, -20)$	30.93	30.49	29.76	28.32	23.92
$(-20, 20)$	13.71	9.79	13.15	13.79	14.19
$(-20, 0)$	13.08	13.82	13.82	13.5	12.50
$(-20, -20)$	14.38	13.71	12.56	9.79	6.67

Table 6. Miss distance (m) with initial flight path angle ($\theta = 5°$).

Initial Relative Distance (km)	**Heading Angle Ψ_v(°)**				
	10	**5**	**0**	**−5**	**−10**
$(20, 20)$	20.17	23.30	25.63	26.21	26.13
$(20, 0)$	24.64	26.02	26.25	25.56	24.02
$(20, -20)$	25.83	25.98	25.26	23.60	19.21
$(0, 20)$	28.93	31.25	31.93	32.21	32.42
$(0, 0)$	32.44	32.56	32.44	32.22	31.72
$(0, -20)$	32.67	32.22	31.64	30.49	28.12
$(-20, 20)$	14.14	17.39	16.93	17.36	17.90
$(-20, 0)$	17.97	17.34	17.18	17.20	16.87
$(-20, -20)$	17.99	17.21	16.69	15.18	13.66

Table 7. Miss distance (m) with initial flight path angle $\theta = 0°$.

Initial Relative Distance (km)	**Heading Angle Ψ_v°**				
	10	**5**	**0**	**−5**	**−10**
$(20, 20)$	16.61	22.12	25.19	25.47	25.37
$(20, 0)$	23.82	25.14	25.63	25.36	23.65
$(20, -20)$	25.59	25.72	24.68	22.12	17.41
$(0, 20)$	30.45	32.10	32.80	33.01	33.24
$(0, 0)$	33.33	33.39	33.23	33.07	32.54
$(0, -20)$	33.48	33.02	32.53	31.22	29.00
$(-20, 20)$	16.69	19.12	18.57	19.06	19.64
$(-20, 0)$	19.22	19.18	18.81	18.83	19.49
$(-20, -20)$	19.68	18.96	18.42	18.07	18.27

Table 8. Miss distance (m) with initial flight path angle $\theta = -5°$.

Initial Relative Distance (km)	**Heading Angle Ψ_v°**				
	10	**5**	**0**	**−5**	**−10**
$(20, 20)$	11.14	19.92	23.16	24.21	24.17
$(20, 0)$	21.31	23.43	23.98	23.69	21.08
$(20, -20)$	23.91	24.24	22.84	19.07	10.31
$(0, 20)$	29.11	32.27	32.85	33.12	33.36
$(0, 0)$	33.25	33.49	33.35	33.15	32.52
$(0, -20)$	33.55	33.12	32.59	31.56	29.19
$(-20, 20)$	18.55	19.65	19.73	19.85	20.47
$(-20, 0)$	20.68	19.68	19.52	19.70	19.96
$(-20, -20)$	20.45	19.77	19.47	18.48	18.74

Table 9. Miss distance (m) with initial flight path angle $\theta = -10°$.

Initial Relative Distance (km)	**Heading Angle Ψ_v°**				
	10	**5**	**0**	**−5**	**−10**
$(20, 20)$	8.70	15.98	20.88	22.21	22.31
$(20, 0)$	20.50	21.60	22.12	21.51	18.47
$(20, -20)$	22.51	22.22	20.22	15.94	6.58
$(0, 20)$	26.72	30.83	31.92	32.34	32.54
$(0, 0)$	32.15	32.53	32.59	32.34	31.37
$(0, -20)$	32.79	32.32	31.61	29.97	25.76
$(-20, 20)$	19.89	19.79	19.68	20.15	20.72
$(-20, 0)$	20.69	20.15	19.81	19.90	20.54
$(-20, -20)$	20.72	20.10	19.54	19.46	17.85

Tables 5–9 illustrate that the slightest miss of these 225 scenarios was 6.58 m, satisfying the evasion problem requirement.

The simulation experiments indicate that the evading policy generated by MS-DDQN can avoid the pursuer's attack in a large domain. For clarity, we also conducted a generic simulation experiment by uniformly sampling 10,000 points in a range of 20×20 km with initial $\theta = 5°$ and $\Psi_v = 5°$ (Figure 13).

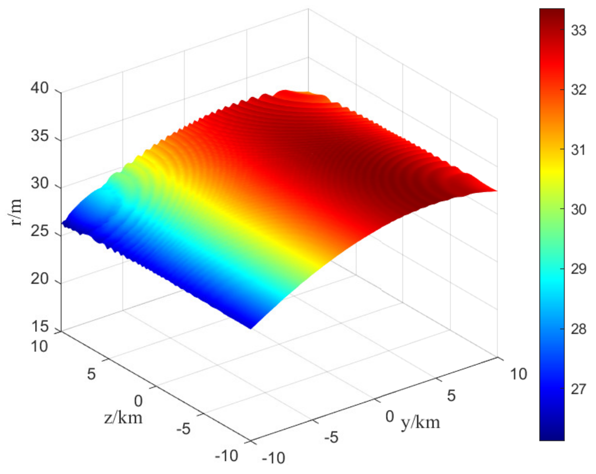

Figure 13. The curved surface of the miss distance and 10,000 different initial positions.

This figure illustrates a successful evasion of the evader where the missing range is $[26.12 \text{ m}, 33.34 \text{ m}]$.

5. Conclusions

This paper proposed an MS-DDQN algorithm to address the problem of high-speed aircraft evading. The evader was treated as the agent for DRL, and the optimal policy was generated through iteratively interacting with the simulation environment by the MS-DDQN framework. Moreover, the proposed two-stage learning method significantly improved the data quality and sped up the training process. As a model-free RL algorithm, MS-DDQN is an adaptive value-based DRL framework that does not require the vehicle's mathematical model.

The effectiveness and robustness of this method were verified through various simulation experiments, illustrating a successful evasion in a large domain when utilizing the policy generated by MS-DDQN. Future research on autonomous decisions for evasion will combine policy-based DRL methods and multi-vehicle autonomous decision-making systems.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no conflict of interest or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Zeng, X.; Yang, L.; Zhu, Y.; Yang, F. Comparison of Two Optimal Guidance Methods for the Long-Distance Orbital Pursuit-Evasion Game. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 521–539. [CrossRef]
2. Lee, J.; Ryoo, C. Impact Angle Control Law with Sinusoidal Evasive Maneuver for Survivability Enhancement. *Int. J. Aeronaut. Space Sci.* **2018**, *19*, 433–442. [CrossRef]
3. Si, Y.; Song, S. Three-dimensional adaptive finite-time guidance law for intercepting maneuvering targets. *Chin. J. Aeronaut.* **2017**, *30*, 1985–2003. [CrossRef]
4. Song, J.; Song, S. Three-dimensional guidance law based on adaptive integral sliding mode control. *Chin. J. Aeronaut.* **2016**, *29*, 202–214. [CrossRef]
5. He, L.; Yan, X. Adaptive terminal guidance law for spiral-diving maneuver based on virtual sliding targets. *J. Guid. Control Dynam.* **2018**, *41*, 1591–1601. [CrossRef]
6. Xu, X.; Cai, Y. Design and numerical simulation of a differential game guidance law. In Proceedings of the 2016 IEEE International Conference on Information and Automation (ICIA), Ningbo, China, 31 July–4 August 2016; pp. 314–318.
7. Alias, I.; Ibragimov, G.; Rakhmanov, A. Evasion differential game of infinitely many evaders from infinitely many pursuers in Hilbert space. *Dyn. Games Appl.* **2017**, *7*, 347–359. [CrossRef]
8. Liang, L.; Deng, F.; Peng, Z.; Li, X.; Zha, W. A differential game for cooperative target defense. *Automatica.* **2019**, *102*, 58–71. [CrossRef]
9. Ibragimov, G.; Ferrara, M.; Kuchkarov, A.; Pansera, B.A. Simple motion evasion differential game of many pursuers and evaders with integral constraints. *Dyn. Games Appl.* **2018**, *8*, 352-378. [CrossRef]
10. Rilwan, J.; Kumam, P.; Badakaya, A.J.; Ahmed, I. A Modified Dynamic Equation of Evasion Differential Game Problem in a Hilbert space. *Thai. J. Math.* **2020**, *18*, 199–211.
11. Jagat, A.; Sinclair, A.J. Nonlinear Control for Spacecraft Pursuit-Evasion Game Using the State-Dependent Riccati Equation Method. *IEEE Trans. on Aerosp. and Electron. Syst.* **2017**, *53*, 3032–3042. [CrossRef]
12. Asadi, M.M.; Gianoli, L.G.; Saussie, D. Optimal Vehicle-Target Assignment: A Swarm of Pursuers to Intercept Maneuvering Evaders based on Ideal Proportional Navigation. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *58*, 1316–1332. [CrossRef]
13. Waxenegger-Wilfing, G.; Dresia, K.; Deeken, J.; Oschwald, M. A Reinforcement Learning Approach for Transient Control of Liquid Rocket Engines. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 2938–2952. [CrossRef]
14. Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, L.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
15. Sun, J.; Liu, C.; Ye,, Q. Robust differential game guidance laws design for uncertain interceptor-target engagement via adaptive dynamic programming. *Int. J. Control* **2016**, *90*, 990–1004. [CrossRef]
16. Gaudet, B.; Furfaro, R.; Linares, R. Reinforcement learning for angle-only intercept guidance of maneuvering targets. *Aerosp. Sci. Technol.* **2020**, *99*, 105746. [CrossRef]
17. Zhu, J.; Zou, W.; Zhu, Z. Learning Evasion Strategy in Pursuit-Evasion by Deep Q-network. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; pp. 67–72.
18. Li, C.; Deng, B.; Zhang, T. Terminal guidance law of small anti-ship missile based on DDPG. In Proceedings of the International Conference on Image, Video Processing and Artificial Intelligence, Shanghai, China, 21 August 2020; Volume 11584.
19. Shalumov, V. Cooperative online Guide-Launch-Guide policy in a target-missile-defender engagement using deep reinforcement learning. *Aerosp. Sci. Technol.* **2020**, *104*, 105996. [CrossRef]
20. Souza, C.; Nwebury, R.; Cosgun, A.; Castillo, P.; Vidolov, B.; Kulić, D. Decentralized Multi-Agent Pursuit Using Deep Reinforcement Learning. *IEEE Robot. Autom. Let.* **2021**, *6*, 4552–4559. [CrossRef]
21. Tipaldi, M.; Iervoline, R.; Massenio, P.R.; Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges. *Annu. Rev. Control* **2022**, *in press*. [CrossRef]
22. Selvi, E.; Buehrer, R.M.; Martone, A.; Sherbondy, K. Reinforcement Learning for Adaptable Bandwidth Tracking Radars. *IEEE Trans. Aerosp. Electron. Syst.* **2020**, *56*, 3904–3921. [CrossRef]
23. Ahmed, A.M.; Ahmad, A.A.; Fortunati, S.; Sezgin, A.; Greco, M.S.; Gini, F. A Reinforcement Learning Based Approach for Multitarget Detection in Massive MIMO Radar. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 2622–2636. [CrossRef]
24. Hu, Q.; Yang, H.; Dong, H.; Zhao, X. Learning-Based 6-DOF Control for Autonomous Proximity Operations Under Motion Constraints. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 4097–4109. [CrossRef]
25. Elhaki, O.; Shojaei, K. A novel model-free robust saturated reinforcement learning-based controller for quadrotors guaranteeing prescribed transient and steady state performance. *Aerosp. Sci. Technol.* **2021**, *119*, 107128. [CrossRef]
26. Volodymyr, M.; Koray, K.; David, S.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.
27. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, San Francisco, CA, USA, 14 August 2016; Volume 48; pp. 1995–2003.
28. Wang, C.; Wang, J.; Wang, J.; Zhang, X. Deep-Reinforcement-Learning-Based Autonomous UAV Navigation with Sparse Rewards. *IEEE Internet Things J.* **2020**, *7*, 6180–6190. [CrossRef]

29. Huang, T.; Liang, Y.; Ban, X.; Zhang, J.; Huang, X. The Control of Magnetic Levitation System Based on Improved Q-network. In Proceedings of the Symposium Series on Computational Intelligence, Xiamen, China, 6–9 December 2019; pp. 191–197.
30. Fan, J.; Wang, Z.; Xie, Y.; Yang, Z. A Theoretical Analysis of Deep Q-Learning. In Proceedings of the Learning for Dynamics and Control, PMLR , Online, 11–12 June 2020; pp. 486–489.
31. Razzaghi, P.; Khatib, E.A.; Bakhtiari, S.; Hurmuzlu, Y. Real time control of tethered satellite systems to de-orbit space debris. *Aerosp. Sci. Technol.* **2021**, *109*, 106379. [CrossRef]