Received: 6 December 2023 | Revised: 25 March 2024 | Accepted: 11 April 2024

DOI: 10.1049/cth2.12677

ORIGINAL RESEARCH

Enhanced LSTM-DQN algorithm for a two-player zero-sum game in three-dimensional space

Bo Lu[1] | Le Ru[1] | Maolong Lv[2] | Shiguang Hu[1] | Hongguo Zhang[3] | Zilong Zhao[4]

[1]National Key Lab of Unmanned Aerial Vehicle Technology, Equipment Management and UAV Engineering College, Air Force Engineering University, Xi'an, Shaanxi, China

[2]Air Traffic Control and Navigation College, Air Force Engineering University, Xi'an, Shaanxi, China

[3]School of Aeronautics, Northwestern Polytechnical University, Xi'an, Shaanxi, China

[4]Social Sciences and Technology, Technical University of Munich, Munich, Germany

Correspondence
Le Ru, National Key Lab of Unmanned Aerial Vehicle Technology, Equipment Management and UAV Engineering College, Air Force Engineering University, Xi'an, Shaanxi, China.
Email: ru-le@163.com

Funding information
Project for Science and Technology, Grant/Award Number: 2022-JCJQ-QT-018; Young Talent Fund of Association for Science and Technology in Shaanxi, Grant/Award Number: 20220101; Postdoctoral International Exchange Project, Grant/Award Number: YJ20220347; National Natural Science Foundation of China, Grant/Award Number: 62303489; Postdoctoral Science Foundation General Program, Grant/Award Number: 2022M723877; Postdoctoral Science Foundation Special Funding, Grant/Award Number: 2023T160790

Abstract
To tackle the challenges presented by the two-player zero sum game (TZSG) in three-dimensional space, this study introduces an enhanced deep Q-learning (DQN) algorithm that utilizes long short term memory (LSTM) network. The primary objective of this algorithm is to enhance the temporal correlation of the TZSG in three-dimensional space. Additionally, it incorporates the hindsight experience replay (HER) mechanism to improve the learning efficiency of the network and mitigate the issue of the "sparse reward" that arises from prolonged training of intelligence in solving the TZSG in the three-dimensional. Furthermore, this method enhances the convergence and stability of the overall solution.An intelligent training environment centred around an airborne agent and its mutual pursuit interaction scenario was designed to proposed approach's effectiveness. The algorithm training and comparison results show that the LSTM-DQN-HER algorithm outperforms similar algorithm in solving the TZSG in three-dimensional space. In conclusion, this paper presents an improved DQN algorithm based on LSTM and incorporates the HER mechanism to address the challenges posed by the TZSG in three-dimensional space. The proposed algorithm enhances the solution's temporal correlation, learning efficiency, convergence, and stability. The simulation results confirm its superior performance in solving the TZSG in three-dimensional space.

1 | INTRODUCTION

A two-player zero sum game (TZSG) occurs when two agents compete against each other in a shared environment. Depending on the actions taken by the agents, they will receive a payoff corresponding to the current state and the transitions that occur in the environment due to their behaviour, leading to the subsequent state. The objective of the first agent (agent 1) is to determine a sequence of actions, starting from a given state that maximizes the total discounted payoff. Similarly, the second agent (agent 2) aims to find a sequence of actions that maximizes the total discounted gain, ultimately seeking to win the game for both parties. This problem is formulated as a Markov game.

In terms of solving TZSG problems, researchers in this field have proposed numerous solution methods. The conventional approaches include the influence diagram method [1], the expert system method [2], the differential game method [3], and the matrix game method [4], among others. While these methods offer practical solutions for two-player zero-sum game problem decision-making to some extent, they also exhibit certain limitations. For instance, the manoeuvres decision of the genetic algorithm tend to be subjective. The expert system method lacks adaptability. The differential game method involves high computational complexity and challenges in finding solutions. Additionally, the matrix game method encounters difficulties in ensuring real-time performance.

Deep reinforcement learning is a distinctive machine learning approach that utilizes feedback from the environment to adapt to and interact with it. By employing trial-and-error methods, it explores the optimal behavioural strategy to maximize the cumulative reward value received from the environment [5]. In recent years, significant advancements have been made in solving various classical problem domains [6–14].

For the specific problem of a TZSG game, Littman proposed the 'extensive minimal Q-learning' algorithm [15]. In a two-player general sum game, the agents' payoffs are usually uncorrelated. However, if one actor's payoffs are detrimental to the other actor's payoffs, the game becomes a zero-sum game. To tackle general sum games, a Nash Q-learning algorithm was proposed [16]. Additionally, FF Q-learning, introduced in [17] for available sum games, demonstrated more robust convergence compared to Nash Q-learning. The concept of correlation Q-learning, which is a generalization of Nash Q-learning and FF Q-learning, was discussed in [18]

In [19], the study focused on exploring the desirable properties of intelligent learning in a scenario involving multiple intelligent agents. A novel learning algorithm, namely the "WoLF" Strategy Hill Climbing, was proposed. In a groundbreaking work, Narasimhan et al. [20] introduced the long short term memory (LSTM) unit into the structure of recurrent neural networks for the first time. They proposed a deep Q-learning (DQN) model called LSTM-DQN, which incorporates LSTM units. This LSTM-DQN network model was successfully applied to a text-based game, yielding promising results. However, none of the aforementioned methods adequately address the problem of a two-player zero-sum game in a three-dimensional space.

For a DRL-controlled agent, the three-dimensional space TZSG problem results in a huge state space due to the large environment space and the continuous change of the game opponent state, which results in an even more sparse state where the intelligences are able to obtain rewards in the dynamic environment. Therefore, this is also a "sparse reward" problem. Reward sparsity can cause failure in the training of agent and thus dealing with "sparse reward" is one of the biggest challenges in reinforcement learning [21].

This work encompasses three contributions: (1) Expanding the TZSG problem from the two-dimensional plane to the three-dimensional space, thereby making the research findings. (2) A Markov game model is developed for theTZSG game problem in three-dimensional space. Additionally, a discrete action space is designed for the TZSG game problem in three-dimensional space. A discrete action space is designed for the movement of an intelligent agent in three-dimensional space, enabling the agent to formulate an effective strategy. (3) The paper introduces and enhances the DQN algorithm, a well-known deep reinforcement learning algorithm based on value functions. The algorithm incorporates the LSTM neural network and the hindsight experience replay (HER) mechanism to accelerate the convergence efficiency of deep reinforcement learning. Finally, a simulation training environment is constructed using a typical three-dimensional space intelligence aircraft and a two-aircraft aerial vehicle. Finally, a simulation training environment is constructed using a typical three-dimensional space intelligence aircraft and a two-aircraft aerial chase scenario. Through training and simulation experiments, it is demonstrated that the LSTM-DQN-HER algorithm exhibits superior convergence and effectiveness in solving the two-player zero-sum game problem in three-dimensional space.

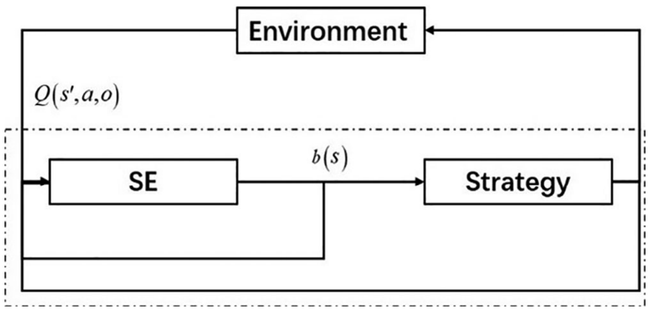


FIGURE 1 Agent1's interaction with the environment under POMDP.

2 | TWO-PLAYER ZERO-SUM MARKOV GAMES WITH DEEP REINFORCEMENT LEARNING

2.1 | Two- player zero-sum Markov game

The two-player zero-sum Markov game (TZMG) can be played with a sextuple (S, U, V, P, r, γ). Description: Where, 'S' denotes the set of finite state spaces observed by the two agents; the number of states is $|S|$; 'U' and 'V' denote the finite action spaces of agent1 and agent2, the number of actions are $|U|$ and $|V|$, respectively; 'P' denotes the transfer probability, that is $P(s', r|s, u, v) : \delta \times U \times V \to \Delta(S \times R)$ denotes the probability of transferring from state $i \in S$ to state $j \in S$ when the actions, $u \in U$, $v \in V$ are chosen by agent1 and two respectively; 'r' denotes the single-stage payoff obtained by agent1 when agent1 and agent2 are interacting with the environment at the same time; denotes the single-stage payoff obtained by agent2 when agent1 is interacting with the environment at the same time, the single-stage gain obtained by agent1; $\gamma \in (0, 1]$ denotes the discount factor.

The interaction between agent and environment under TZMG is shown in Figure 1; at the beginning of a round of the game, the environment is in the initial state, each time step t, agent1 selects an action under the current state S_t according to the random strategy $\pi(\cdot|S_t) : S \to \Delta(U)$ and the obtained action is noted as U_t, meanwhile, agent2 selects an action under the current state S_t according to its strategy and the obtained action is indicated as V_t. The joint action (U_t, V_t) is sent to the environment for execution, and the environment transfers the state of the environment S_t, S_{t+1} according to the probability distribution of the state transfer and reward generation P and gives the reward r_t, that is, $S_{t+1}, r_t \sim P(\cdot, \cdot | S_t, U_t, V_t)$. This is done until the termination moment of the current round of the game T. Each round of the game generates a trajectory $\tau := (S_0, U_0, V_0, r_0, S_1, \dots, S_{T-1}, U_{T-1}, V_{T-1}, r_{T-1}, S_T)$.

The reward for agent1 is defined as the cumulative discount reward, $G(\tau) := \sum_{t=0}^{T-1} \gamma^t r_t$, and for TZMG, the expected reward when the initial state is S is defined as $V_s(\pi, \mu) := E_{\pi,\mu}[G(\tau)|S_0 = s]$, and since the initial state S_0 obeys the probability distribution P, the expected reward can be defined as:

$$V_P(\pi, \mu) := E_{s\sim P}\left[V_s(\pi, \mu)\right] = E_{\pi,\mu}\left[G(\tau)\right] \tag{1}$$

From Equation (1), it can be seen that the expected return $V_P(\pi, \mu)$ is not only related to your strategy but also to the other player's strategy, that is, the joint strategy of both parties (π, μ) jointly determines the value of $V_P(\pi, \mu)$.

2.2 | Deep reinforcement learning based on POMDP

2.2.1 | Model

Markov decision process (MDP) serves as the fundamental framework for deep reinforcement learning. MDP refers to a decision-making framework in which the decision maker periodically or continuously observes a stochastic dynamic system with Markovian properties. Based on the current observation state, the decision maker selects an action to execute, with the goal of transitioning to the next state. Importantly, the next state is determined solely by the current state and the action taken [22].

Partially observable markov decision process (POMDP) is a type of decision-making framework [23, 24] that simulates the decision-making process of an intelligent agent. In POMDP, the system's dynamics are determined by the underlying MDP. However, the agent is unable to fully observe the current state of the entire domain, leading to incomplete information.

Typically, POMDP arises from two scenarios: (1) when multiple states produce identical sensor readings due to limited perception of the environment by the intelligent agent and 2) when sensor- readings are noisy, resulting in different sensor data for the same state. This paper constructs the training environment based on scenario (1). Specifically, the observation data is transmitted to the agent not at the simulation intrinsic frequency, but at a fixed simulation frequency interval. As a result, the agent does not receive the other's information in a timely manner, which resembles the real-world situation faced by two agents in a three-dimensional game.

In a partially observable POMDP, intelligent agents utilize partially observable information from the environment to make decisions. These agents exist in a random state at any given moment. However, because the partially observable information is incomplete, they are unaware of their current state. Therefore, they must rely on observable information, historical sequences, and reward values available to them when making decisions.

The POMDP relies on both current observations and historical state actions to make optimal decisions. These decisions are based on the belief state b, behaviour a, and observations o, which are used to estimate the belief state b' within the current state s'. The specific formula is as follows:

$$b'[s'] = \Pr(s'|o, a, b) = \frac{O(s', a, o)\sum_{s\in S} T(s, a, s')b(s)}{\Pr(o|a, b)} \tag{2}$$

When solving the POMDP problem, a non-Markov chain approach necessitates access to historical action information in order to make decisions based on the current state. However, with the incorporation of the belief state space, the POMDP problem can be tackled using a Markov chain approach, enabling decisions to be made in the present without relying on historical action information. The optimal strategy and value function are as follows:

$$\pi_t^*(\mathrm{b}) = \arg\max_a \left[\sum b(s)R(s, a) + \gamma \sum \Pr(o|b, a)V_t^*\left(b'\right)\right] \tag{3}$$

$$V_t^*(b) = \max_a \left[\sum b(s)R(s, a) + \gamma \sum \Pr(o|b, a)V_{t-1}^*\left(b'\right)\right] \tag{4}$$

MDP and POMDP serve as the fundamental building blocks of reinforcement learning. Deep reinforcement learning combines deep learning with reinforcement learning and utilizes neural networks to approximate the value function and strategy function in reinforcement learning. This approach overcomes the limitation of the small state space in traditional reinforcement learning methods [25]. In the context of a two-player zero-sum game problem that involves continuous multidimensional state features that are partially observable in a three-dimensional space, this paper proposes the utilization of a deep Q-network model called LSTM-DQN (Deep Q-Network with long-short term memory). LSTM-DQN is a type of deep Q-network that incorporates long-short term memory units [20].

2.2.2 | POMDP in three-dimensional TZSG game

The overall state of the TZSG under the realisation of the POMDP condition is the same as the state shown in Figure 2 in Section 3.1 among the agents.The difference is that, according to the implementation conditions of POMDP, it is necessary to obtain non-real-time and delayed state information when the controlled agent is observing the state. The exact implementation of this paper is given in detail in the experimental setup.

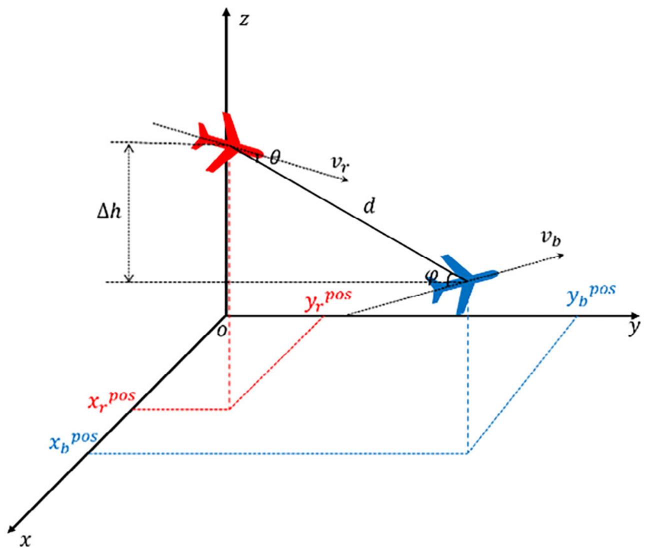


FIGURE 2 Modelling of TZMG dynamics in three-dimensional space.

3 | MODELLING OF THREE-DIMENSIONAL SPACE TWO-PLAYER ZERO-SUM MARKOV GAMES

3.1 | Three-dimensional space two-player zero-sum Markov game problem Mathematical description

This paper presents the problem of a three-dimensional space two-player zero-sum Markov game (TZMG) as a process of mutual pursuit between moving objects in the air. The motion model of a three-dimensional spatial agent in the Cartesian coordinate system is constructed using the dynamic equations of the simulated vehicle developed in the literature [26]. The game situation is mainly defined by the position of agent1 and agent2 $(x^{\mathrm{pos}}, y^{\mathrm{pos}}, z^{\mathrm{pos}})$, the trajectory deflection angle ψ, the roll angle ϕ, the rate of change of the roll angle ϕ', the velocity v and the amount of time change δt:

$$\begin{cases} x^{pos} = x^{pos} + v \cdot \cos(\phi)\cos\left(\psi'\right)\delta t \\ y^{pos} = y^{pos} + v\sin(\phi)\,\delta t \\ z^{pos} = z^{pos} + v \cdot \cos(\phi)\sin\left(\psi'\right)\delta t \\ v = v' + dv \cdot \delta t \\ \phi = \phi + \phi' \cdot \delta t \\ \phi = \max(\phi, -\phi_{\max}) \\ \phi = \min(\phi, \phi_{\max}) \\ \psi' = \dfrac{g}{v}\tan\phi \end{cases} \tag{5}$$

The three-dimensional space two-player zero-sum Markov game dynamics of agent1 and agent2 are set according to the agent's aerial motion model as shown in Figure 2:

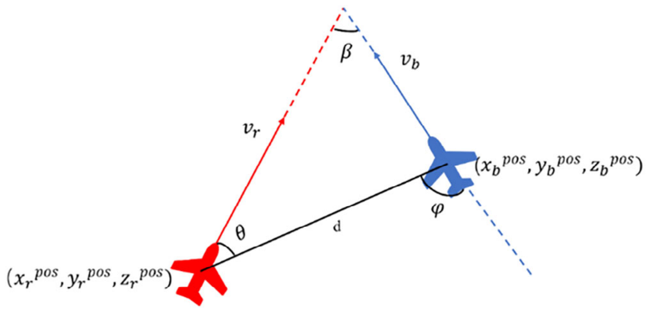

β
v_b
v_r
$(x_b{}^{pos}, y_b{}^{pos}, z_b{}^{pos})$
φ
d
θ
$(x_r{}^{pos}, y_r{}^{pos}, z_r{}^{pos})$

FIGURE 3 Relative geometry between agent1 (r) and agent2 (b).

In Figure 2, we assume that both the angle of attack and the side-slip angle are zero. This means that the velocity coordinate system and the body coordinate system align when the agent is moving in three-dimensional space. The axis of the agent represents the direction of velocity, and the coordinate system used is the spatial Cartesian coordinate system of the environment. Under this coordinate system, the three-dimensional space state is primarily represented by the following states: d is shown as the distance between agent1(r) and agent2(b), v_r and v_b represents the speeds of agent1(r) and agent2(r), θ is the heading angle of agent1 (ATA), φ is the angle of the target of agent1 (OTA), Δh is the altitude difference, the angle of speed β_i, and thus the state variables of the two sides of the game are as follows:

$$\varphi_i = \arccos\left\{\left[\left|x_i - x_{\bar{i}}\right| cos\psi_i cos\varphi_i + \left|y_i - y_{\bar{i}}\right| sin\psi_i cos\varphi_i + \left|h_i - h_{\bar{i}}\right|\right]/d\right\} \tag{6}$$

$$\theta_i = \arccos\left\{\left[\left|x_i - x_{\bar{i}}\right| cos\psi_{\bar{i}} cos\varphi_{\bar{i}} + \left|y_i - y_{\bar{i}}\right| sin\psi_{\bar{i}} cos\varphi_{\bar{i}} + \left|h_i - h_{\bar{i}}\right| sin\varphi_{\bar{i}}\right]/d\right\} \tag{7}$$

$$d = \sqrt{(x_i - x_{\bar{i}})^2 + (y_i - y_{\bar{i}})^2 + (h_i - h_{\bar{i}})^2} \tag{8}$$

$$\Delta h = \left|h_i - h_{\bar{i}}\right| \tag{9}$$

$$\beta_i = \arccos\left(cos\psi_i cos\varphi_i cos\psi_{\bar{i}} cos\varphi_{\bar{i}} + cos\psi_i sin\varphi_{\bar{i}} cos\psi_{\bar{i}} cos\varphi_{\bar{i}} + sin\varphi_i sin\varphi_{\bar{i}}\right) \tag{10}$$

If i is agent1 (r), $\bar{i}$ it is agent2 (b).

Specifically, as shown in Figure 3. The goal of agent1 (R) is to gain and maintain a dominant position behind agent2 (B), and this dominant position can be quantified using the heading angle θ (ATA) and the objective angle φ (OTA). In addition, the heading cross angle β (HCA) is also used to describe the difference in orientation between agent1 and agent2.

3.2 | Modelling of two-player zero-sum Markov games in three-dimensional space

To achieve deep reinforcement learning for solving two-player zero-sum Markov games in three-dimensional space, modelling of the Markov decision process is required. The machine determines a tuple (S, U, V, P, γ, r) based on which a stochastic game model in air combat is constructed.

3.2.1 | POMDP modelling of a three-dimensional space two-player zero-sum Markov game

In this paper, we introduce the concept of a partially observable Markov decision process (POMDP). A POMDP is a tuple consisting of (S, A, T, R, Ω, O, γ), where S represents the state space, A represents the action space, T represents the transition model, R represents the reward function, Ω represents the observation space, O represents the observation model, and γ represents the discount factor. At each time period, an Agent takes an action in a certain state s and transitions to a new state, s' with a probability determined by the transition model $\mathrm{T}(s, a, s') = \mathrm{P}(\mathrm{s}'|\mathrm{s}', \mathrm{a})$. The agent then receives an observation $o \in O$ of the environment with a probability determined by the observation model $\mathrm{O}(o, s', a) = \mathrm{P}(o|s', a)$, which is dependent on the new state of the environment. This paper implements partial observables using a method based on a similar approach to the literature [20], which involves partially selecting the state space.

(1) State space

According to the factors affecting the agent tracking three-dimensional space target posture, the state characteristics of the agent can be determined, now selected to control the agent1, thus mainly by the two sides of the spatial coordinates $(x_i^{pos}, y_i^{pos}, z_i^{pos}), i \in a\text{gent1}(\mathrm{r}), a\text{gent2}(\mathrm{b})$, agent1 (r) and agent2 (b) distance between the two players of the 'd.' agent1 (r) and agent2 (b) speed v_r, v_b, the angle of the agent is heading in the three-dimensional space when flying (ATA) $\theta_i, i \in a\text{gent1}(\mathrm{r}), a\text{gent2}(\mathrm{b})$, the agent's angle of target (OTA) $\varphi_i, i \in a\text{gent1}(\mathrm{r}), a\text{gent2}(\mathrm{b})$, the angle of velocity $\beta_i, i \in a\text{gent1}(\mathrm{r}), a\text{gent2}(\mathrm{b})$, Δh is the height difference between the two intelligent bodies, from which the state space can be expressed as: $S = (v_i, x_i^{pos}, y_i^{pos}, z_i^{pos}, \varphi_i, \theta_i, \Delta h, \beta_i) i \in a\text{gent1}(\mathrm{r}), \text{agent2}(\mathrm{b})$.

The DRL environment sends the elements of the state space to the agent every second. In order to create a two-player zero-sum PO Markov game, we developed an observation mechanism that relies on the state space. This involved adding an observation mechanism to the state inputs of the neural network and extracting states at intervals of ≈5–10 HZ during the state inputs. As a result, we were able to transform the original continuous state space into a discontinuous observation space.

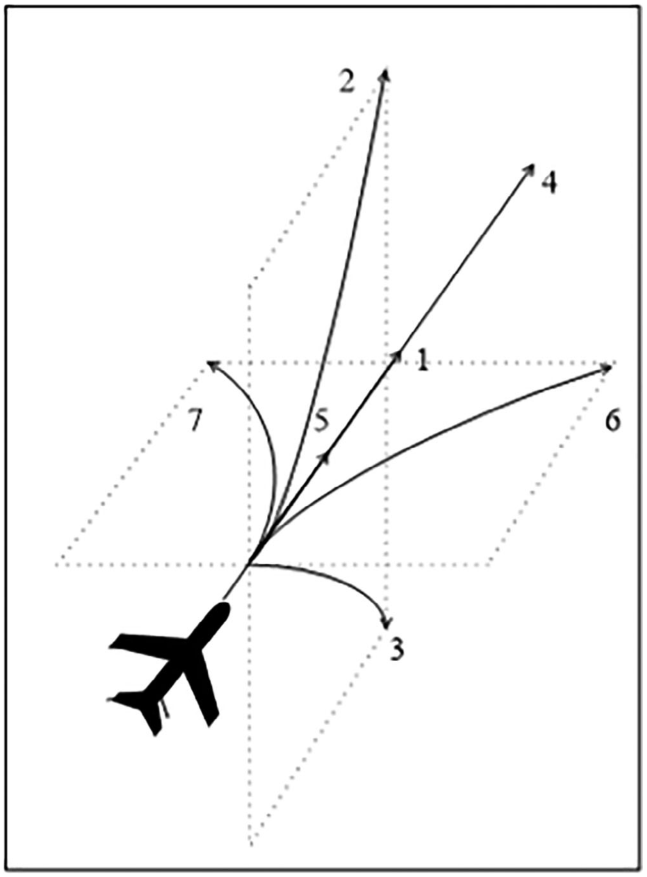


FIGURE 4 Basic manoeuvres.

(2) Action space

Figure 4 depicts the seven basic actions that the Agent can perform within a time step Δt.

(1) Achieve stable flight by maintaining constant control. (2) Increase the state to the maximum overload. (3) Decrease the state to the maximum overload. (4) Reach the maximum increase in axial acceleration. (5) Reach the maximum decrease in axial acceleration. (6) Attain the maximum increase in roll rate. (7) Attain the maximum decrease in roll rate.

By utilizing these seven combinations of fundamental manoeuvres, the description of the manoeuvre decision space is simplified, resulting in a reduction in computational complexity. Moreover, this approach offers the flexibility to generate a wide range of actual manoeuvre postures. The fundamental principle of manoeuvre decision making entails selecting the appropriate basic manoeuvre that maximizes the desired discount return within the current state environment. Notably, among these manoeuvres, $U = V = \{1, 2, 3, 4, 5, 6, 7\}$.

(3) Return function 'r'

In a three-dimensional space two-player zero-sum Markov game, the Q-value of the MDP is used to represent the immediate reward. Let's denote $Q(s, u, v)$ as the expected reward for each state, given the action taken by your side (u) and the action taken by the blue side (v). Although this paper discretizes the action space of the three-dimensional two-player zero-sum Markov game to limit the range of possible actions, the state space becomes vast due to the extension into three dimensions. Using only the end-of-round state as the reward leads to sparse reward signals and low learning efficiency, hence the need for reward shaping.

3.2.2 | Game reward function shaping

The reward design in this paper utilizes a combination of sparse and dense rewards to accelerate convergence by providing agent1 (r) with guidance throughout the training process, aiming to surpass the performance of agent2 (b). More specifically, this paper incorporates a single sparse reward and two dense rewards.

(1) Sparse rewards

The sparse reward is the great reward given by the environment at the final moment S_t when the agent interacts with the environment for a complete episode; in this paper, the sparse reward is set as follows:

$$r_{\text{env}} = \begin{cases} r_0, & d < 1\,\text{km}, \theta_r < 2°, \varphi_r > 2°, \beta_r < 30° \\ -r_0, & d < 1\,\text{km}, \theta_b < 2° \\ 0, & \text{others} \end{cases} \tag{11}$$

which sets the size r_0 to 100. The function return means that the intelligences must beat each other while avoiding being beaten by each other.

(2) Heuristic thick rewards

In a two-player three-dimensional zero-sum Markov game scenario, agent1(r) and agent2(b) strategically choose their manoeuvres based on their individual strategies, calculating their aerial positions to gain a favourable advantage. In this three-dimensional manoeuvre space, the player who occupies a superior relative position generally holds a greater advantage in the confrontation. By leveraging this understanding, situational rewards or advantages can be fine-tuned based on the relative positional advantages that the intelligences have in the aerial game. These positional advantages encompass angular, distance, speed, and altitude advantages.

Angular advantage reward: In a game of manoeuvring decisions between two parties in three-dimensional space, agent1 can only gain an advantage and improve their chances of winning if agent2 is situated within a specific section of airspace known as the "tailgating posture" ahead of them. Conversely, agent1 is at a disadvantage when being tailgated. When both parties are moving in opposite directions or backwards, they are in a neutral position. To measure this advantage, we use the following equation: the smaller the deviation angle (q_r) and the larger the disengagement angle (q_b), the closer Agent1 is to the tail pursuit posture, resulting in a greater angular advantage for agent1.

$$r_a = 1 - \frac{|ATA| + |OTA|}{2\pi} = 1 - \frac{|\theta_r| + |\varphi_r|}{2\pi} \tag{12}$$

where θ_r is the deviation angle of agent1 (r) and φ_r detachment angle.

Distance advantage reward: The presence of two-player zero-sum Markov game is a crucial factor that affects the effectiveness of the strategy. When agent2 is closer to our side, the chances of agent1 winning over agent2 increase. Therefore, the distance advantage function is defined as follows: To reach the target as quickly as possible, the distance between the enemy and us in the next moment should be shorter than the distance in the current moment. As a result, the distance reward is defined as:

$$r_d = \frac{v_r \Delta t}{|S| - |S'|} \tag{13}$$

Where: 'S' is the position at the current moment, S' is the position at the next moment v_r is the agent1 (r) velocity; $v_r\Delta t$ is used to normalize the distance reward.

Speed advantage function: The speed advantage primarily considers the current velocities of both agents, along with the distance between them and the conditions for obtaining sparse rewards.

$$r_v = \begin{cases} 0.1 & (v_b < 0.6 v_r) \\ -0.5 + v_b / v_r & (0.6 v_r \le v_b \le 1.5 v_r) \\ 1 & (\text{v}_b > 1.5 v_r) \end{cases} \tag{14}$$

In the above equation, v_r, v_b denote the flight speed of agent1 (r) and agent1 (b), respectively.

In order to further promote agent1's ability to gain an advantage and win the end game, a special relative health advantage reward is added.

$$r_{\text{blood}} = \text{blood}_{r,t} - \text{blood}_{b,t} \tag{15}$$

where $\text{blood}_{r,t}$ and $\text{blood}_{b,t}$ denote the remaining health of the two agents at time t.

Based on the analysis of the air posture advantage function above, the total advantageous reward shaping is obtained by weighting the advantage functions of angle, distance and speed r_{shape}; in this paper, the red and blue agents have the same manoeuvre capability, so the relative advantage brought by Agent performance is not taken into account. The total reward shaping is calculated as follows:

$$r_{shape} = \omega_a r_a + \omega_d r_d + \omega_v r_v + r_{\text{blood}} \tag{16}$$

The sum of the three parameters $\omega_a, \omega_d, \omega_v$ is equal to 1, that is, $\omega_a + \omega_d + \omega_v = 1$, ω_a, ω_d, ω_v the weights of angle advantage reward, distance advantage reward, and speed advantage reward, which have different weights in different game postures, respectively. As the current discussion is, the game problem in the case of two agents with closer distance, the weight of the angle advantage reward of the Agent is more significant.

3.2.3 | Composite return function

Combining the above reward designs, the integrated reward function obtained by the intelligences is:

$$r = r_{\mathrm{env}} + r_{shape} \tag{17}$$

4 | BASED ON THE TZMG LSTM-DQN-HER ALGORITHM

4.1 | DQN algorithm

The core of the Q-learning algorithm involves constructing a Q-table, which is continuously updated and enhanced through the ongoing interaction between an intelligent agent and its environment, allowing the agent to learn improved behavioural strategies. The Q-table serves as a state-action table, where the values represent the maximum expected rewards achievable by taking a specific action in a given state. However, when the agent operates in an environment with a high number of dimensions and a large number of states, relying solely on the Q-table to store state-action functions can lead to a phenomenon known as the 'dimensionality catastrophe.'

To tackle the problem, the DeepMind team combined deep neural networks with Q-learning algorithms, leading to the creation of the DQN algorithm. DQN is an improved version of Q-learning that employs a deep neural network to dynamically generate a table of Q-values. It approximates the state-action-value function by continuously and iteratively updating the parameters of the neural network f. This algorithm extends the capabilities of Q-learning by utilizing deep neural networks for the dynamic generation of the Q-value table.

The neural network structure of the DQN algorithm consists of two parts: the estimation network and the target network, which differ only in their network parameters. The algorithm utilizes the estimation network to estimate the Q-values of all actions in the action space based on the current state. It collaborates with the output values of the target network to calculate the loss. The estimation network learns and updates the network parameters in real-time and copies the parameters to the target network every specific round to update them.

During training, a batch of samples from the empirical replay buffer needs to be randomly and uniformly selected. These samples are mixed with the training data to destroy the correlation among them. The loss function of the DQN algorithm is defined as the square of the difference between the output Q-values of the target network and the estimation network, as shown in Equation (18).

$$\mathrm{Loss} = \left(Q\left(s_t, a_t; \theta\right) - \mathrm{target}Q\right)^2 \tag{18}$$

where target Q is calculated as shown in Equation (19).

$$\mathrm{target}Q = r_{t+1} + max Q'\left(s_{t+1}, a; \theta_t^-\right) \tag{19}$$

After calculating the loss value, the DQN updates the network parameters using the gradient descent method, and the gradient descent formula is shown in Equation (20).

$$\theta_{t+1} = \theta_t + E\left[\mathrm{target}Q - Q\left(s_t, a_t; \theta_t\right)\right] \nabla Q\left(s_t, a_t; \theta_t\right) \tag{20}$$

4.2 | LSTM-DQN-HER

4.2.1 | LSTM-based DQN deep network improvement

The benchmark DQN algorithm [27] utilizes two neural networks, namely the Q network and the target Q network, to approximate the original Q-table. This approach tackles the issues of the 'dimensional catastrophe' and the instability that arises when a single neural network is used to fit the Q-table. These challenges stem from the vast and intricate state space. Moreover, it also resolves the problem of instability when fitting a single neural network.

In the benchmark DQN algorithm, the two neural networks are composed of an input layer, two fully connected (FC) layers, and an output layer. For the specific problem of a three-dimensional two-player zero-sum game, each fully connected layer consists of 256 nodes. The input layer has 128 nodes, and the output layer has 49 nodes. These nodes represent the state variables of the UAV at each moment and the $Q(s, a)$ value of each action in the action space at that moment. Afterwards, the AI selects an action 'a' based on the $Q(s, a)$ value using a ε-greedy approach.

However, due to the problem of the three-dimensional two-player zero-sum game, the state eigenquantities of the Agent are partially observable and multi-dimensional. The discretized state space is extensive, and when using a connected neural network to fit $Q(s, a)$, the training process may encounter the issues of gradient explosion and gradient vanishing as the network size increases. This can lead to a training collapse or invalidation. Additionally, the fully connected neural network can only handle inputs at a single point in time, and the inputs at two points in time before and after are completely unrelated, as illustrated in Figure 5. However, in the three-dimensional two-player zero-sum game problem, the state at each moment exhibits a strong temporal correlation with the action performed by the Agent.

The long short-term memory (LSTM) network, proposed in the literature [28], is an advanced variant of the recurrent neural network (RNN) specifically designed for sequential data processing. It tackles the challenges of gradient explosion and gradient vanishing that often arise during training, which are common issues in conventional RNNs. Additionally, LSTM has demonstrated remarkable effectiveness in handling tasks involving highly correlated time series data. The LSTM architecture comprises an input gate, a forgetting gate, and an output gate, as illustrated in Figure 6.

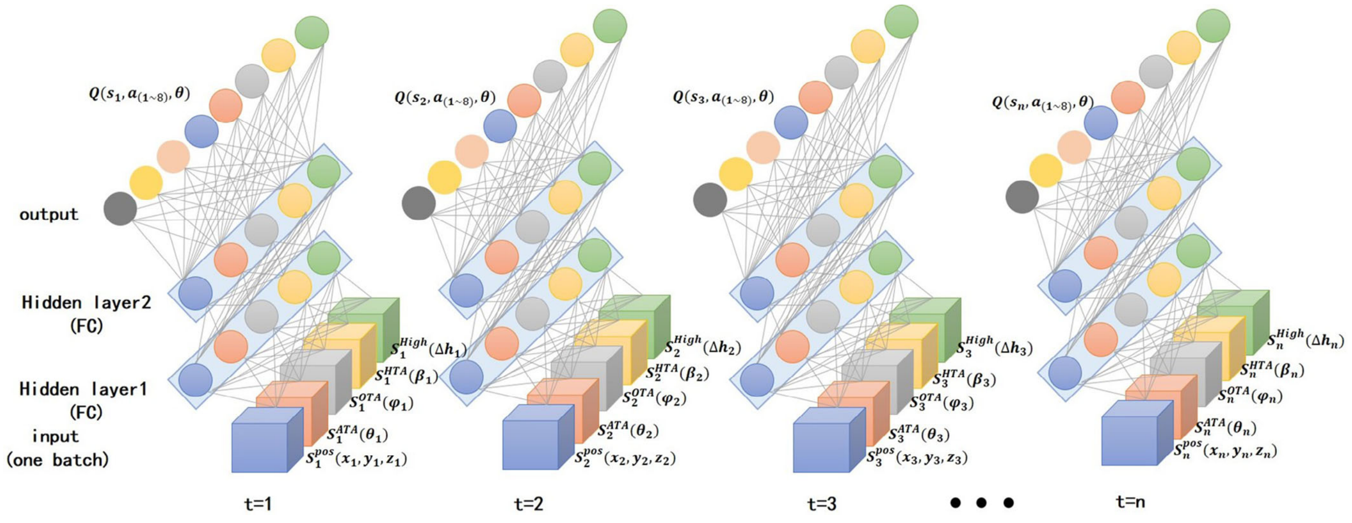


FIGURE 5 Benchmark DQN using a fully connected network structure diagram.

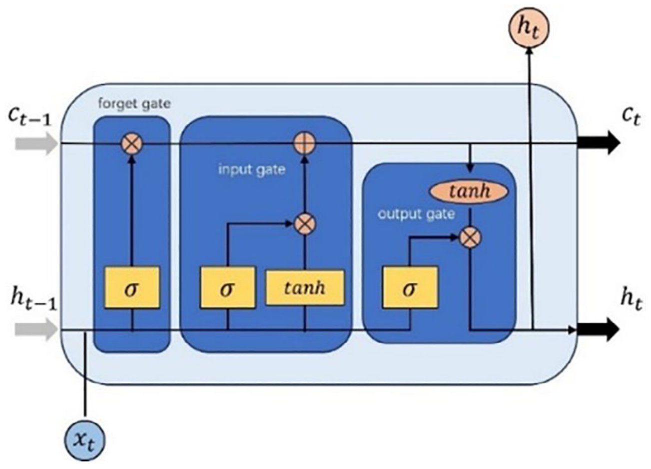


FIGURE 6 Basic structure inside LSTM.

The 14-dimensional vector of game states is fed directly into the LSTM layer. Its internal structure is shown in Equation (21):

$$\begin{aligned} i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ c'_t &= tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \\ c_t &= f_t * c_{t-1} + i_t * c'_t \\ h_t &= o_t * tanh\left(c_t\right) \end{aligned} \tag{21}$$

x_t is the input of the current moment, c_{t-1}, h_{t-1} is the cell state and output of the previous moment, c_t, h_t represents the cell state and output of the current moment, σ is the sigmoid activation function, and *tanh* is the *tanh* activation function. The unique gate structure makes its output not only related to the current moment's input and the previous moment's output but also to the previous moment's cell state.

In this paper, outside the fully connected (FC) neural network layer of the deep network, an additional LSTM layer is added to the two layers, and the improved neural network is shown in Figure 7.

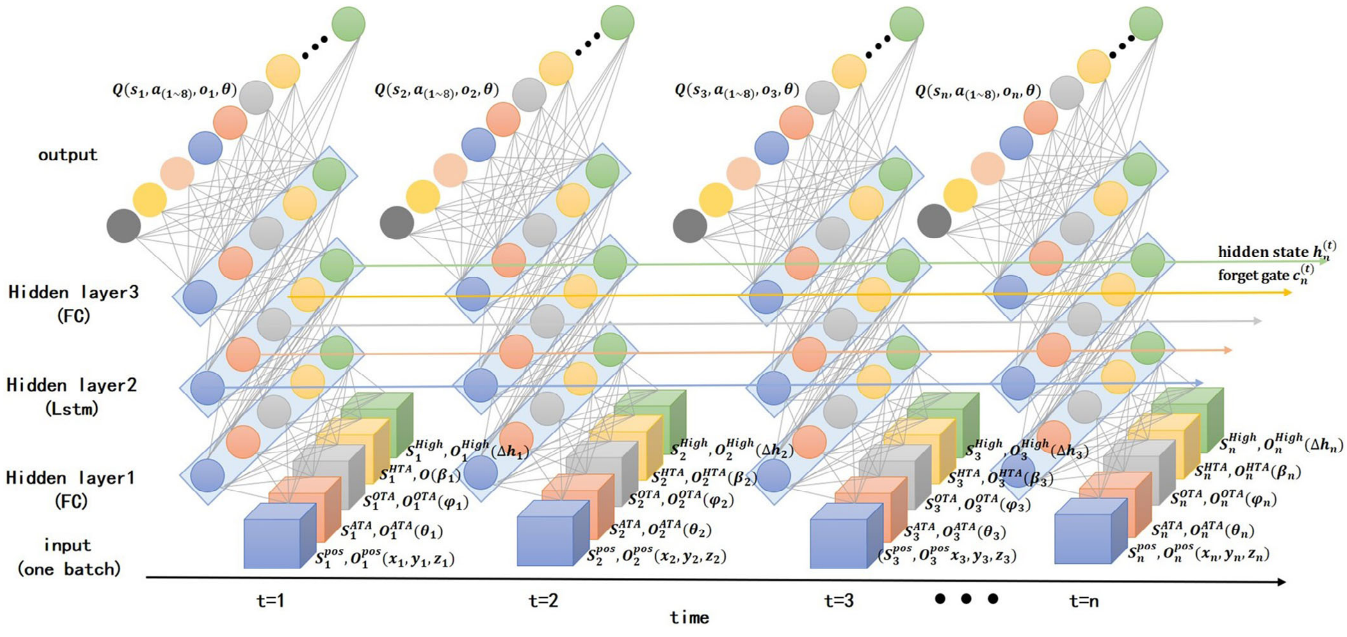
Unlike the conventional LSTM-DQN [20], the enhanced network structure for TZSG incorporates an LSTM layer between the two FC layers to handle continuous partially observable state tuples. The LSTM-DQN-HER utilizes two neural networks with identical structures but different parameters to update the network parameters. These networks consist of an evaluation network, which functions as the training network, and a target network. The parameters of the evaluation network are periodically copied to the target network after every N time steps. The objective function (i.e. loss function) of the whole network is shown in Equation (22), y is the target value calculated by the target network $Q_{\text{target}}(s', a')$ as shown in Equation (23), $Q_{\text{eval}}(s, a, \theta)$ is the predicted value obtained by the evaluation network, that is, the value to be optimised, and is the parameters of the network model. The loss function is defined as the mean square deviation of the predicted and target values, and the Adam optimiser optimises the loss.

$$\text{loss}\,(\theta) = \frac{1}{2}\left[r(s) + \gamma maxQ\left(s', a', o', \theta^-\right) - Q\left(s', a', o', \theta\right)\right]^2 \tag{22}$$

$$Q_{\text{target}} = r + \gamma \max_{a'} Q\left(s', a'; \theta^-\right) \tag{23}$$

where r is the reward value, γ and is the discount factor $\gamma \in [0, 1]$. After that, we take the partial derivatives of the network parameters θ.

$$\begin{aligned} \nabla_\mu L(\theta) = E\Bigg[\bigg(\Big(r + \gamma \max_{a'} Q\left(s', a'; \theta^-\right)\Big) \\ - Q\left(s, a, o; \theta\right) \nabla_\theta Q\left(s, a, o; \theta\right)\Bigg] \end{aligned} \tag{24}$$



$Q(s_1, a_{(1\sim8)}, o_1, \theta)$
$Q(s_2, a_{(1\sim8)}, o_2, \theta)$
$Q(s_3, a_{(1\sim8)}, o_3, \theta)$
$Q(s_n, a_{(1\sim8)}, o_n, \theta)$
output
Hidden layer3 (FC)
Hidden layer2 (Lstm)
Hidden layer1 (FC)
input (one batch)
hidden state $h_n^{(t)}$
forget gate $c_n^{(t)}$
$S_1^{High}, O_1^{High}(\Delta h_1)$
$S_1^{HTA}, O(\beta_1)$
$S_1^{OTA}, O_1^{OTA}(\varphi_1)$
$S_1^{ATA}, O_1^{ATA}(\theta_1)$
$S_1^{pos}, O_1^{pos}(x_1, y_1, z_1)$
$S_2^{High}, O_2^{High}(\Delta h_2)$
$S_2^{HTA}, O_2^{HTA}(\beta_2)$
$S_2^{OTA}, O_2^{OTA}(\varphi_2)$
$S_2^{ATA}, O_2^{ATA}(\theta_2)$
$S_2^{pos}, O_2^{pos}(x_2, y_2, z_2)$
$S_3^{High}, O_3^{High}(\Delta h_3)$
$S_3^{HTA}, O_3^{HTA}(\beta_3)$
$S_3^{OTA}, O_3^{OTA}(\varphi_3)$
$S_3^{ATA}, O_3^{ATA}(\theta_3)$
$(S_3^{pos}, O_3^{pos} x_3, y_3, z_3)$
$S_n^{High}, O_n^{High}(\Delta h_n)$
$S_n^{HTA}, O_n^{HTA}(\beta_n)$
$S_n^{OTA}, O_n^{OTA}(\varphi_n)$
$S_n^{ATA}, O_n^{ATA}(\theta_n)$
$S_n^{pos}, O_n^{pos}(x_n, y_n, z_n)$
t=1
t=2
t=3
t=n
time

FIGURE 7 Network structure of improved DQN algorithm using LSTM.

4.2.2 | Hindsight experience replay mechanism

Deep reinforcement learning (RL) algorithms have been explored in the context of multitasking. One prominent approach is the HER algorithm, which aims to mimic the learning process of human beings by leveraging failure experiences. This paper applies HER to two-player zero-sum games in a three-dimensional space. The key insight is that while there is a vast exploration space for training discrete action-based intelligences in such games, the states that lead to reward shaping or the final reward constitutes only a small fraction of the total state space. Consequently, relying solely on reward shaping for algorithm convergence can result in a lengthy or even infeasible training cycle. To address this challenge, the paper proposes processing and selecting experiences in the replay buffer, focusing on useful experiences that exhibit a gradient growth trend. It is important to note that the paper also maintains an exploratory nature in the training of intelligent agents, gradually increasing a certain proportion of experiences during the training process.

Specifically, the HER implementation in this paper is generally divided into two stages as follows:

(1) Goal state-based training generation: A goal $g \in G$ (which remains constant throughout the training process) is preset at the beginning of each round of training, such that at each time step t, the policy output of the intelligent body will depend on both the current state and the goal g, that is $\pi = \pi(\mathrm{a}|\mathrm{s}, \mathrm{o}, g)$. Similarly, the payoff function $r = r(s, a, o, g)$ and the Q function is:

$$Q^{\pi}(s, a, o, g) = E(R|s, a, o, g) \tag{25}$$

Finally, the sequence obtained after the interaction of the strategy with the environment is stored as a tuple in the experience playback cache, that is:

$$D = s, a, r(s, a, o, g), o, s', g \tag{26}$$

(2) State update based on target experience transformation: During the experience storage phase, we sample experience sequences of a specific length K to prevent excessive correlation generation of samples during the sampling process. This length is larger than the timestep size of the LSTM layer, which aims to enhance the performance of the LSTM network. Additionally, for HER (hindsight experience replay), we constantly select multiple target states and update the sequence tuples as $r = r(s, a, o, g)$ and g'. These modified tuples are then stored in the experience playback cache. The update process, based on 3D-TZSG and LSTM, is as follows: After a number of Batches, randomly select k state sequences in the same round as the new target. Furthermore, throughout the entire training process, randomly select k states as the new target.

The HER mechanism algorithm is shown in Table 1:

4.2.3 | Structure of the TZSG LSTM-DQN-HER algorithm

The algorithm structure of this paper mainly consists of TZSG LSTM-DQN-HER and a three-dimensional spatial agent body interaction environment, as illustrated in Figure 8. After successfully initializing the environment, agent1 randomly selects an action to execute by calculating the initialization in the first step. Agent2 determines an action to execute based on

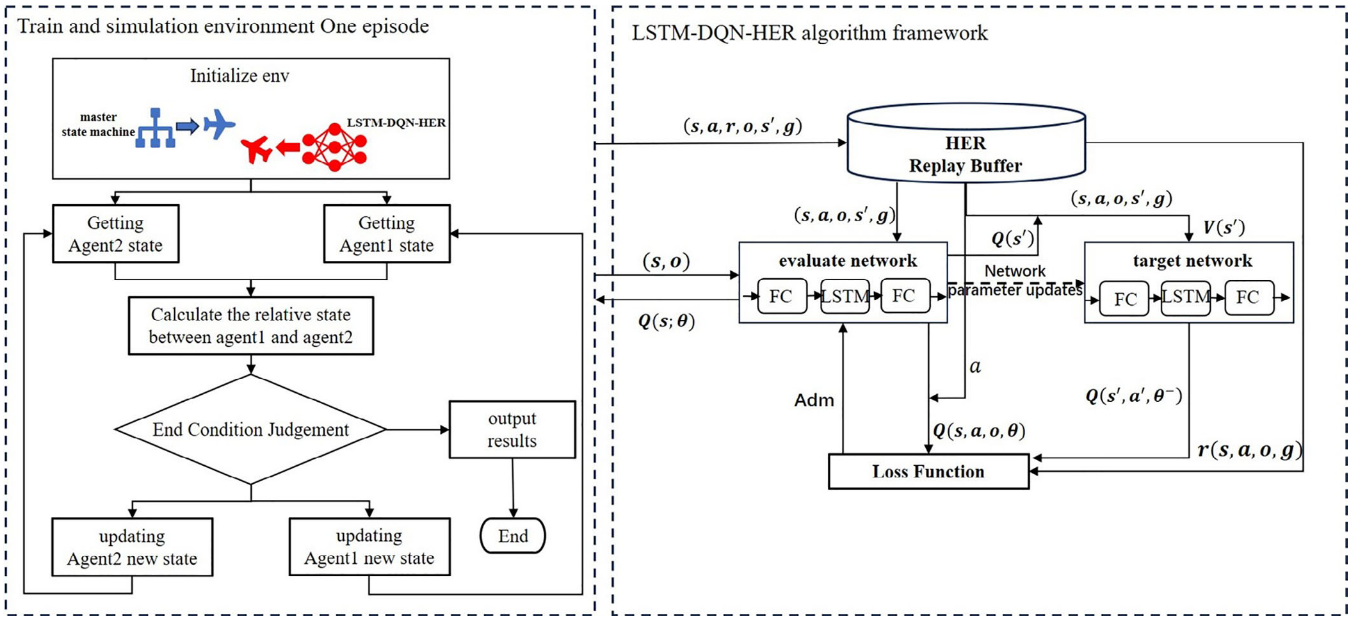


FIGURE 8 Framework of LSTM-DQN-HER algorithm.

the state machine synthesis, obtaining the reward value '*r*'. This reward value is then passed into the improved TZSG LSTM-DQN-HER. The TZSG LSTM-DQN-HER reads the state information corresponding to this reward value and processes it to form a vector tuple, as described in Equation (20). This tuple serves as an input to the evaluation network of the TZSG LSTM-DQN-HER algorithm, which computes the $Q(s, a, o, g)$ value for each set of parameters (i.e. each action) in this state. The algorithm employs an epsilon-greedy policy to output actions as tuning parameters for the next run.

The objective of deep reinforcement learning is to maximize the discounted reward. In the improved LSTM-DQN-HER parameter adaptive tuning algorithm, the goal is to identify the optimal combination of actions in each state to continuously increase the advantage. At time t, the experience generated after agent1 takes action '*a*' is denoted as (s, a, o, r, s', g), where '*s*' represents the current state at time t, '*a*' represents the action taken at time t, '*o*' represents the current observation at time t, '*r*' represents the immediate reward obtained from taking action '*a*', and '*s*'' represents the state after taking action '*a*' at time t. This experience is stored in the experience replay buffer periodically.

During the learning and training phase, a sequence is randomly selected from the experience replay buffer, and '*n*' time points are randomly chosen from the sequence. For learning and training, the algorithm randomly selects a sequence from the experience replay buffer and learns from '*n*' time steps in reverse order (n can be 1). Before each update, the LSTM hidden layer state is reset to 0. The random selection of sequences and time steps disrupts the continuity between samples, enhancing the effectiveness of training. Additionally, when the experience replay buffer reaches its capacity, new experience sequences will overwrite existing ones.

4.2.4 | LSTM-DQN-HER algorithm training steps

In the training of a three-dimensional two-player zero-sum game, the agent utilizes the LSTM-DQN-HER learning algorithm mentioned above to make manoeuvre decisions. Each episode consists of multiple game rounds, which conclude when the agent captures an enemy aircraft, escapes from an enemy aircraft, or reaches the maximum round time. At this point, a new round begins, and the simulation environment is reset. Additionally, periodic evaluations are conducted to assess the agent's decision-making ability during the training process and account for the impact of learning. During these evaluations, the random probability is reduced to 0, allowing the decision-making model to directly output the action with the highest Q-value. The dominance function value is then calculated at the end of each round, and the agent's learning efficiency is compared across different episodes, shown in Table 2.

5 | SIMULATION AND ANALYSIS OF RESULTS

5.1 | Experimental setup

As previously mentioned, the problem of a three-dimensional two-player zero-sum game involves synthesizing a class of problems with similar characteristics. A typical example of this problem is the pursuit problem, which entails two intelligent entities in the air. To effectively validate the proposed algorithm's efficacy in addressing this problem, we conducted experiments using a python-based simulation training environment. This environment utilized a nonlinear dynamic model

TABLE 1 Hindsight experience replay mechanisms.

```
Algorithm 1 Hindsight experience replay
Initialise $\mathbb{A}$
Initialise replay buffer $\mathcal{D}$
for episode = 1, M do
    Sample a goal $g$ and an initial state $s_0$.
    for $t = 0, T-1$ do
        Sample an action $a$ using the behavioural policy from: $\mathbb{A}$
        $a_t \leftarrow \pi_b(s_t \parallel g)$
        Execute the action $a$ and observe a new state $s'$
    end for
    for t = 0, T−1 do
        $r = r(s, a, o, g)$
        Store the transition $(s \parallel g, a, o \parallel g, r, s' \parallel g)$ in $\mathcal{D}$
        Sample a set of additional goals for replay $G := \mathbb{S}(\text{currentepisode})$
        for $g' \in G$ do
            $r' := r(s, a, o, g')$
            Store the transition $(s \parallel g', a, o \parallel g', r', s' \parallel g')$ in $\mathcal{D}$
        end for
end for
for $t = 1, N$ do
    Sample a minibatch $B$ from the replay buffer $\mathcal{D}$
    Perform one step of optimisation using $\mathbb{A}$ a minibatch $B$
end for
end for
```

of a fighter jet. To solve the relevant posture information, we employed the Eulerian method to handle the dynamic function with a 100 Hz action stepping period, the state is observed by the agent every 10 Hz. The algorithm proposed in this paper is a value function learning method based on discrete action space. Agent 1's decision-making rate for manoeuvres is limited to 1 Hz and it employs the TZSG LSTM-DQN-HER algorithm. On the other hand, Agent 2 executes manoeuvre actions based on a manoeuvre strategy state machine. As a control experiment, agent 2's strategy remains constant while agent 1 uses DQN, DDQN, Dueling DQN, and LSTM-DQN algorithms to select actions. The hyperparameters for each algorithm are set as shown in the table below, shown in Table 3:

The construction of the three-dimensional TZSG posture is illustrated in Figure 9. Figure 9a displays the initial unfavourable posture of the algorithm, which validates its adaptability. Figure 9b portrays the initial advantageous posture of the algorithm, which confirms its effectiveness, Tables 4 and 5 show the initial parameters of the agent.

The spatial coordinates of agent are $\boldsymbol{c}_i (i \in R, B)$, and the inclination and declination of agent's trajectory are $\gamma_i (i \in R, B)$, $\psi_i (i \in R, B)$, $v_i (i \in R, B)$ are velocity size, the initial health value, both of them which is 3. The end game condition of each round is that either side's health value goes to 0, measured by the most extended duration of a game if no one wins. The initial

TABLE 2 LSTM-DQN-HER learning algorithm based on the two-player zero-sum game in three-dimensional space.

```
Algorithm 2 LSTM-DQN-HER
1.  Initialise the HER experience playback pool $\mathcal{D}$ with a capacity of $N$
2.  Create two neural networks: LSTM Q network and target LSTM network.
      The parameters of the LSTM Q network are $\theta$ as follows:
The target network parameters are; $\theta^- = \theta$
3.  for episode = 1,2,…, M do:
4.      Initialisation: state of the environment; $S_1$
5.      for step = 1,2,…,$T$ do.
6.          Generate actions using the $\varepsilon - greedy$ strategy $a$
7.          Execute the action $a$, get the next state $s$, receive the reward $r$, and
            observe the strategy o executed by Blue;
8.          Update $Q(s,a,o)$: $Q(s,a,o) = R(s,a,o) + \gamma \sum_{s'} T(s,a,o,s')V(s')$
9.          Solve using linear programming:
            $V(s) = \max_{\pi \in \mathrm{PD}(A)} \min_{o \in O} \sum_{a \in A} Q(s,a,o)\pi_a$ using argmax and update
            $V(s)$ and $\pi_a(s)$;
10.     Sample transitions from the moment i$(s \parallel g, a, o \parallel g, r, s' \parallel g)$ are stored
        in the HER experience playback pool $\mathcal{D}$;
11.         Randomly select a sample of minibatch transitions from the HER
            experience playback pool $\mathcal{D}$. $(s \parallel g', a, o \parallel g', r', s' \parallel g')$
12.         If the step $j+1$ is the final moment, then $y_j = r_j$
If the $j+1$ step is not the final moment, then the $y_j = r_j + \gamma Q(s', a', \theta^-)$
13.         Update $\mathrm{loss}(\theta) = \frac{1}{2}[r(s) + \gamma MinmaxQ(s', a', o', \theta^-) -$
        $Q(s', a', o', \theta)]^2$ using gradient descent;
14.         LSTM units are implemented for network learning using
            Equation (21) and F.C. layer network learning using Equations (22),
            (23) and (24);
Update the network node status;
15.         Update the target network every $C$ step $\theta^- = \theta$,
16.     End for
17. End for
```

health value is 3. The endgame condition for each round is that either player's health goes to 0, and the most extended game duration is measured when no one wins. This is usually set to 300 s.

5.2 | Training results

We incorporated the LSTM-DQN-HER method with the five core algorithms of deep reinforcement learning, which rely on value functions as discussed in Section 5.1. These algorithms were utilized for training purposes in the 3D TZSG environment. We successfully obtained training results in both disadvantageous and advantageous states, as illustrated in Figure 10a,b.

By analysing the experimental results, we present here two main findings.

To enhance the convergence speed of a deep reinforcement learning algorithm based on the value function for solving the 3D TZSG problem, the addition of an LSTM network

TABLE 3 Hyperparameters of each algorithm.

	α	γ	$\varepsilon - greedy$	tau	Timestep	Batch size	Capacity
LSTM-DQN-HER	0.001	0.9	0.99	0.005	100	512	1,000,000.
LSTM-DQN	0.001	0.9	0.99	0.005	100	512	1,000,000.
DDQN	0.002	0.95	0.99	0.005	None	1024	1,000,000.
DQN	0.001	0.91	0.99	None	None	2048	1,000,000.
Dueling DQN	0.003	0.93	0.99	0.005	None	1024	1,000,000.

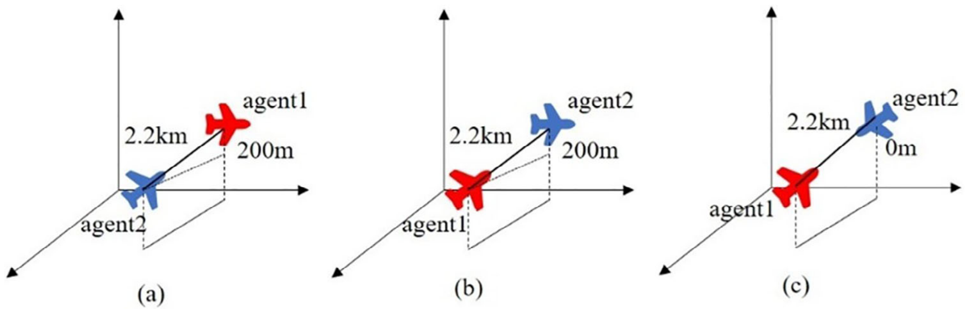


FIGURE 9 Initial posture.

TABLE 4 Agent1 initial settings.

	c_R	γ_R	ψ_R	v_R	Blood
Figure 9a	(−1997.48, 0.04, −6000.19)	0°	0°	350 m/s	3
Figure 9b	(−0.04, 2.50, −7000.19)	0°	270°	350 m/s	3
Figure 9c	(−0.04, 2.50, −7000.19)	0°	0°	350 m/s	3

TABLE 5 Agent2 initial settings.

	c_B	γ_B	ψ_B	v_B	Blood
Figure 9a	(−0.04, 2.50, −7000.19)	0°	270°	350 m/s	3
Figure 9b	(−1997.48, 0.04, −6000.19)	0°	0°	350 m/s	3
Figure 9c	(−1997.48, 0.04, −6000.19)	0°	180°	350 m/s	3

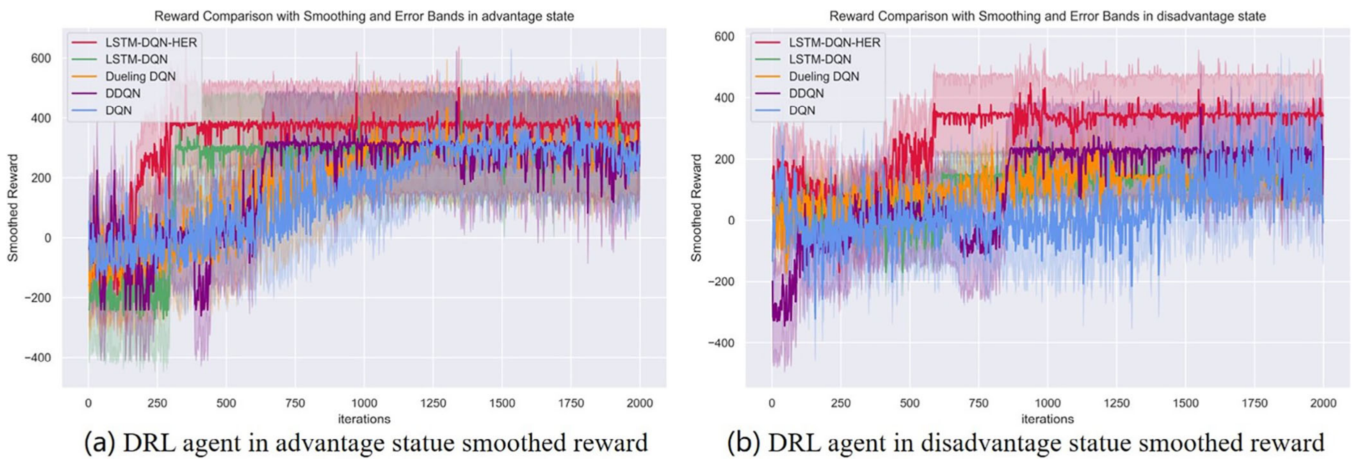


FIGURE 10 Average normalised episodic rewards for the adversary training experiment using state machines. Using one machine (Intel Core i9-13900K CPU (5.0GHz, 24 cores), 1 NVIDIA GeForce RTX 4080 16G GPU, 64GB RAM).

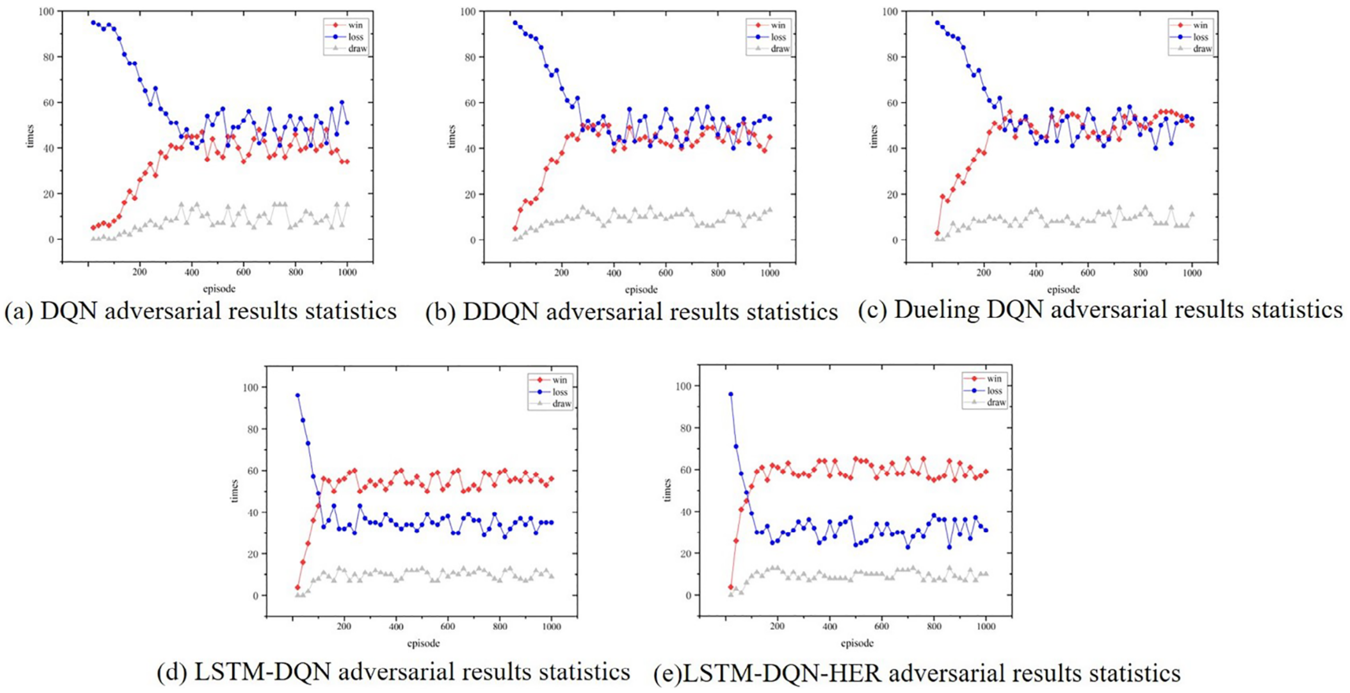


FIGURE 11 Confrontation curve of DRL algorithm with gaming expert system.

(shown in Figure 10a,b) effectively improves convergence. Both algorithms that incorporate the LSTM achieve early convergence, with the LSTM-DQN-HER converging at the 250th iteration and the LSTM-DQN converging at the 300th iteration in the advantageous case. In the disadvantageous case, the LSTM-DQN-HER converges at the 600th iteration, and the LSTM-DQN converges at the 700th iteration. This demonstrates that the 3D TZSG problem is influenced by the timing of actions and states.

When solving the 3D TZSG problem using the same DRL algorithm, the introduction of the empirical processing mechanism of HER (as shown in Figure 10a,b) leads to better algorithmic performance metrics. Specifically, the rewards obtained by LSTM-DQN-HER are higher than those of the LSTM-DQN algorithm, regardless of the game situation. Furthermore, the LSTM-DQN-HER algorithm, with the HER mechanism in place, is the only one capable of achieving the final algorithmic metrics in the dominant state, even when starting from an inferior initial state.

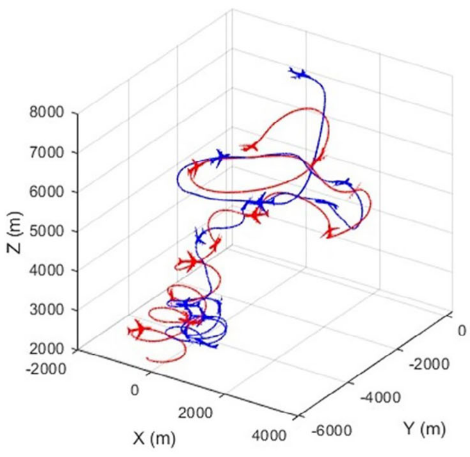
5.3 | Adversarial game experiments with expert systems

In addition to algorithm convergence, evaluating the effectiveness of the algorithm in the two-player zero-sum game problem also takes into account the winners and losers of the game. In the experiments conducted in this paper, the neural network saves the model every 1000 iterations of learning. After the training is completed, the model is loaded sequentially according to the chronological order shown in Figure 9c. Different expert system opponents are then used to play the game for 100 rounds, and the win/loss outcomes of each confrontation are statistically calculated. The experimental results are presented in Figure 11a–e.

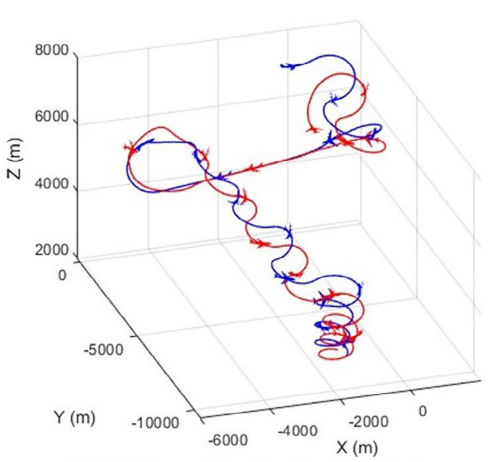
In Figure 11a, the DQN algorithm demonstrates an average win rate of approximately 40%. The wins do not exhibit significant improvement beyond the 400th episode, indicating convergence of the algorithm. Moving on to Figure 11b, the DDQN algorithm achieves an average win rate of around 43% and converges at approximately the 300th episode. Similarly, the Dueling DQN algorithm in Figure 11c attains an average win rate of about 45% and converges at around the 300th episode. In Figure 11d, the LSTM-DQN algorithm showcases an average win rate of about 52% and converges at about the 180th episode. Lastly, in Figure 11e, the LSTM-DQN-HER algorithm achieves an average win rate of approximately 60% and converges at around the 150th episode. These results demonstrate that the LSTM-DQN-HER algorithm exhibits strong performance and robustness in solving the 3D TZSG problem.

5.4 | Behavioural emergent analysis

In the context of deep reinforcement learning, it is important to not only analyse the quantitative data but also incorporate visual analysis to gain a more comprehensive understanding of the training results. By designing the training environment appropriately, we can visually assess the outcomes. During our observations and analysis of various test simulations, we have identified a common dogfighting pattern that occurs when the DRL Agent holds a dominant position. This pattern involves



(a) DRL agent in advantage statue



(b) DRL agent in disadvantage statue

FIGURE 12 Manoeuvring trajectories of the R.L. Agent game at different initial conditions.

the effective use of a "vertical manoeuvre" tactic by the DRL Agent, as shown in Figure 12a. In this particular scenario, both sides engage in an intense struggle and pursuit, leading to a decrease in altitude. Subsequently, a risky "low-altitude dogfighting process" takes place, where the RL Agent aims to maintain a position directly behind the opponent, thereby reducing the opponent's health or causing mistakes such as poorly controlled altitude that may result in near-ground collisions or forced pull-ups.

In our review and analysis, we found that when at a disadvantage, the DRL Agent learns a manoeuvre strategy (shown in Figure 12b) that allows it to turn its disadvantageous position into a balanced one. This strategy involves a process called "vertical manoeuvre," where the DRL Agent ascends vertically, performs a rolling inversion manoeuvre, and then quickly follows it with a continuous turning manoeuvre to outmanoeuvre the adversary and gain an advantage in the process.

6 | CONCLUSIONS

In this study, we have proposed an innovative deep reinforcement learning solution, named LSTM-DQN-HER, for a two-player zero-sum game conducted in three-dimensional space. Our solution enhances the existing LSTM-DQN algorithm by integrating the LSTM layer of recurrent neural networks and the HER mechanism. Through our training experiments, we demonstrate the effectiveness of the LSTM-DQN algorithm enhanced with LSTM and HER. In comparison to other deep reinforcement learning methods that rely on value functions, our approach significantly improves learning efficiency and algorithm performance. Moreover, our test results reveal that the strategies learned using the LSTM-DQN-HER algorithm exhibit strong emergence and generalization capabilities. These strategies empower the agent to consistently and swiftly navigate through combinatorial actions to achieve a favourable position. This study not only provides a viable approach for solving two-player zero-sum game problems in a three-dimensional space using deep reinforcement learning based on value functions but also offers valuable insights for enhancing the performance of DRL algorithms.

AUTHOR CONTRIBUTIONS

Bo Lu: Conceptualization; formal analysis; funding acquisition; investigation; methodology; resources; software; visualization; writing—original draft; writing—review and editing. **Le Ru**: Conceptualization; funding acquisition; methodology; project administration; resources; software; supervision; writing—review and editing. **Maolong Lv**: Data curation; methodology; supervision; writing—review and editing. **Shiguang Hu**: Software; supervision; validation; writing—review and editing. **Hongguo Zhang**: Resources; software; validation. **Zilong Zhao**: Supervision; writing—review and editing.

ACKNOWLEDGEMENTS

The authors would like to express our sincere gratitude to all the individuals who have supported us in conducting this work, as well as the reviewing committee for their valuable feedback.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Bo Lu https://orcid.org/0009-0007-8285-776X
Shiguang Hu https://orcid.org/0000-0001-6819-4700

REFERENCES

1. Virtanen, K., et al.: Modeling air combat by a moving horizon influence diagram game. J. Guid. Control Dyn. 29(5), 1080–1091 (2006). https://doi.org/10.2514/1.17168

2. Chappell, A.R.: Knowledge-based reasoning in the paladin tactical decision generation system. In: Proceedings of the IEEE/AIAA 11th Digital Avionics Systems Conference, pp. 155–160. IEEE, Piscataway, NJ. https://doi.org/10.1109/dasc.1992.282166
3. Horie, K., Conway, B.A.: Optimal fighter pursuit-evasion maneuvers found via two-sided optimization. J. Guid. Control Dyn. 29(1), 105–112 (2006). https://doi.org/10.2514/1.3960
4. Su, M.-C., et al.: A new approach to multi-aircraft air combat assignments. Swarm Evol. Comput. 6, 39–46 (2012). https://doi.org/10.1016/j.swevo.2012.03.003
5. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. IEEE Trans. Neural Networks 9(5), 1054–1054 (1998). https://doi.org/10.1109/tnn.1998.712192
6. Xia, Z.G., Fang, Z.: A preliminary study of the U.S. Army's artificial intelligence air combat system Alpha. In: Proceedings of the Sixth China Command and Control Conference (Upper Volume), (2018)
7. Silver, D., et al.: Mastering the game of Go without human knowledge. Nature 550(7676), 354–359 (2017). https://doi.org/10.1038/nature24270
8. Franois-Lavet, V., et al.: An introduction to deep reinforcement learning. Found. Trends Mach. Learn. 11(3-4), 219–354 (2018)
9. Ma, Y., et al.: A case study on air combat decision using approximated dynamic programming. Math. Probl. Eng. 2014, 1–10 (2014). https://doi.org/10.1155/2014/183401
10. Silver, D., Hubert, T., et al.: A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. Science 362(6419), 1140–1144 (2018). https://doi.org/10.1126/science.aar6404.
11. Zhao, E., et al.: AlphaHoldem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning. Proc. AAAI Conf. Artif. Intell. 36(4), 4689–4697 (2022). https://doi.org/10.1609/aaai.v36i4.20394
12. Vinyals, O., et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575, 350–354 (2019).
13. Ye, D., et al.: Mastering complex control in MOBA games with deep reinforcement learning. Proc. AAAI Conf. Artif. Intell. 34(04), 6672–6679 (2020). https://doi.org/10.1609/aaai.v34i04.6144
14. Ye, D., et al.: Towards playing full MOBA games with deep reinforcement learning. In: Proceedings of the 14th International Conference on Interfaces and Human Computer Interaction 2020 and 13th International Conference on Game and Entertainment Technologies 2020. IADIS Press (2020). https://doi.org/10.33965/ihci_get2020_202010l016
15. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. Mach. Learning Proc. 1994, 157–163 (1994). https://doi.org/10.1016/b978-1-55860-335-6.50027-1
16. Hu, J., Wellman, M.P.: Nash Q-learning for general-sum stochastic games. J. Mach. Learn Res. 4, 1039–1069 (2003)
17. Greenwald, A., Hall, K., Serrano, R.: Correlated Q-learning. ICML 3, 242–249 (2003)
18. Bowling, M., Veloso, M.: Rational and convergent learning in stochastic games. In: International Joint Conference on Artificial Intelligence, vol. 17(1), Lawrence Erlbaum Associates Ltd, pp. 1021–1026 (2001)
19. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multi-agent reinforcement learning. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 38(2), 156–172 (2008)
20. Narasimhan, K., et al.: Language understanding for text-based games using deep reinforcement learning. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1–11. Association for Computational Linguistics, Stroudsburg, Pennsylvania (2015). https://doi.org/10.18653/v1/d15-1001
21. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., et al.: Indsight experience replay. arXiv:1707.01495 (2017). https://doi.org/10.48550/arXiv.1707.01495.
22. White, C.C. III: A survey of solution techniques for the partially observed Markov decision process. Ann. Oper. Res. 32(1), 215–230 (1991). https://doi.org/10.1007/bf02204836
23. Spaan, M.T.J.: Partially Observable Markov Decision Processes. Springer-Verlag, Berlin. https://doi.org/10.1007/springerreference_179341
24. Nguyen, T., et al.: Multi-agent deep reinforcement learning with human strategies. In: Proceedings of the 2019 IEEE International Conference on Industrial Technology (ICIT), (2019). IEEE, Piscataway, NJ. https://doi.org/10.1109/icit.2019.8755032
25. McGrew, J., et al.: Air combat strategy using approximate dynamic programming. In: Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit, American Institute of Aeronautics and Astronautics, (2008). https://doi.org/10.2514/6.2008-6796
26. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
27. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
28. Andrychowicz, M., et al.: Hindsight experience replay. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 5055–5065. Curran Associates Inc., Red Hook, NY (2017)

How to cite this article: Lu, B., Ru, L., Lv, M., Hu, S., Zhang, H., Zhao, Z.: Enhanced LSTM-DQN Algorithm for a Two-Player Zero-Sum Game in Three-Dimensional Space. IET Control Theory Appl. 18, 2798–2812 (2024). https://doi.org/10.1049/cth2.12677