Scilight

*Intelligence & Control*
https://www.sciltp.com/journals/ic

*Article*

Check for updates

# Look-Ahead-Look-Back Hindsight Experience Replay for Motion Planning of Spherical Multi-Telescopic-Legged Robots

Xinyun Liu, Fengde Xu and Xudong Zhao *

Key Laboratory of Intelligent Control and Optimization for Industrial Equipment of Minstry of Education, Dalian University of Technology, Dalian 116081, China
* Correspondence: xudongzhao@dlut.edu.cn

**How To Cite**: Liu, X.; Xu, F.; Zhao, X. Look-Ahead-Look-Back Hindsight Experience Replay for Motion Planning of Spherical Multi-Telescopic-Legged Robots. *Intelligence & Control* **2026**, *2*(2), 1. https://doi.org/10.53941/ic.2026.100004

Received: 28 February 2026
Revised: 3 April 2026
Accepted: 13 April 2026
Published: 21 April 2026

**Abstract:** To address the issues of sparse rewards, low sample efficiency, and limited policy generalization in offline motion planning of spherical multi-telescopic-legged robots in complex three-dimensional environments, this paper proposes a Look-Ahead-Look-Back Hindsight Experience Replay (LALB-HER) algorithm. Building upon the future goal sampling strategy of the conventional Hindsight Experience Replay (HER) framework, the proposed method introduces a historical trajectory reinforcement mechanism to enhance the utilization of historical experiences, thereby mitigating the degradation of model generalization performance caused by the diminishing influence of past samples. On this basis, the Soft Actor–Critic (SAC) algorithm is adopted as the underlying reinforcement learning framework, into which the proposed LALB-HER mechanism is integrated. In addition, a task-oriented reward function is designed to facilitate stable convergence and efficient policy learning. Simulation results in complex environments demonstrate that the proposed method not only significantly accelerates policy convergence but also effectively improves the generalization performance of the learned policy under varying task conditions.

**Keywords:** spherical multi-telescopic-legged robot; motion planning; Hindsight Experience Replay; Soft Actor–Critic

## 1. Introduction

In the field of mobile robot motion planning, particularly in complex and dynamic three-dimensional environments, how to efficiently learn robust and adaptable policies from limited interaction experience remains a critical challenge [1–7]. In scenarios such as Mars exploration and disaster rescue, where environments are highly unstructured and terrains are variable, spherical multi-telescopic-legged robots exhibit significant locomotion potential due to their unique structure and mobility. Consequently, motion planning for such robots has attracted considerable research attention. These robots typically rely on deep reinforcement learning (DRL) to perform offline policy learning in simulation environments [8–11]. However, conventional approaches often suffer from slow convergence due to the high complexity of environmental states and insufficient sampling efficiency. Moreover, they tend to exhibit poor generalization when encountering unseen target positions or obstacle configurations [12–14].

In recent years, Hindsight Experience Replay (HER) has been proposed as an effective technique to improve sample efficiency and policy generalization [15–17]. By relabeling goals in hindsight for a single trajectory, HER enables experience reuse and significantly enhances data utilization. While HER has demonstrated promising performance in motion planning tasks with low-dimensional state spaces, such as robotic manipulators, it remains insufficient for spherical multi-telescopic-legged robots, which operate in high-dimensional state spaces and unstructured environments.

To address the challenges posed by high-dimensional motion planning, reinforcement learning algorithms have continuously evolved. Advanced methods such as Twin Delayed Deep Deterministic Policy Gradient (TD3) [18] and Soft Actor-Critic (SAC) [19] have achieved remarkable success in complex control tasks. Motivated by these advancements, this paper adopts SAC as the base learning framework due to its strong exploration capability and stable learning performance. Furthermore, to overcome the limitations of HER in cross-episode experience utilization, an

**Copyright:** © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).
**Publisher's Note:** Scilight stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

improved mechanism termed Look-Ahead-Look-Back Hindsight Experience Replay (LALB-HER) is proposed. By deeply integrating LALB-HER with SAC, the proposed method aims to enhance motion planning performance and improve generalization across different goals and environments for spherical multi-telescopic-legged robots in complex 3D settings.

The main contributions of this paper are summarized as follows:

(1) Based on the SAC framework, the state space and action space are designed to match the structural characteristics of the spherical multi-telescopic-legged robot. A corresponding reward function is also constructed to facilitate effective policy learning and convergence.

(2) Building upon the future-goal relabeling mechanism of HER, a historical experience retrieval and reuse strategy based on state similarity is introduced, leading to the proposed LALB-HER algorithm. This approach enables dual-dimensional experience reuse across both current and past episodes, thereby improving policy generalization and accelerating the training process.

The remainder of this paper is organized as follows. Section 2 introduces the fundamentals of the SAC algorithm. Section 3 presents the design of the state and action spaces for the spherical multi-telescopic-legged robot. Section 4 describes the construction of the reward function. Section 5 details the proposed LALB-HER algorithm and its theoretical foundation. Section 6 validates the effectiveness of the proposed method through simulation experiments. Finally, Section 7 concludes the paper.

## 2. SAC Algorithm

This paper improves the data-driven SAC reinforcement learning algorithm by taking into account the structural and motion characteristics of the spherical multi-telescopic-legged robot, as well as the requirements of its motion planning scenarios, so as to enhance the adaptability, stability, and effectiveness of the planning strategy in complex environments.

*Principle of the SAC Algorithm*

The SAC algorithm is a deep reinforcement learning method based on the maximum entropy, which has garnered significant attention for its excellent balance between policy exploration and task performance.

The objective function of the SAC algorithm is as follows:

$$J(\pi)=\sum_{t=0}^{T} E_{\left(s_t, a_t\right) \sim \rho_\pi}\left[r\left(s_t, a_t\right)+\alpha H\left(\pi\left(\cdot \mid s_t\right)\right)\right] \tag{1}$$

where $\rho_\pi$ represents the policy, $r\left(s_t, a_t\right)$ indicates the reward associated with the current state $s_t$ and the action $a_t$ , $\alpha$ represents the temperature coefficient that governs the extent of entropy's influence during the optimization process, $\pi\left(\cdot \mid s_t\right)$ signifies the policy under the current state $s_t$ which defines the probability distribution of all possible actions given the state $s_t$ , and $H\left(\pi\left(\cdot \mid s_t\right)\right)$ represents the entropy of the policy.

The calculation formula for $H\left(\pi\left(\cdot \mid s_t\right)\right)$ is given by:

$$\begin{aligned} H\left(\pi\left(\cdot \mid s_t\right)\right) &= -\sum_{t=0}^{T} \pi\left(a_t^* \mid s_t\right) \log \pi\left(a_t^* \mid s_t\right) \\ &= E\left(-\log \pi\left(a_t^* \mid s_t\right)\right) \end{aligned} \tag{2}$$

where $a_t^*$ denotes the action sampled from the probability distribution $\pi\left(\cdot \mid s_t\right)$ , which has the highest execution probability, and $\pi\left(a_t^* \mid s_t\right)$ represents the probability of the action $a_t^*$ .

The SAC algorithm used in this paper consists of one Actor network and four Q Critic networks . The four Q Critic networks are subdivided into two Q Critic networks and two Q Critic target networks. The agent continuously obtains its state information $s_t$ from the environment, feeding $s_t$ into the Actor network, which outputs the action $a_t$ . The agent then executes the action $a_t$, transitioning to the next state $s_{t+1}$ . Based on the predefined reward function, the reward $r\left(s_t, a_t\right)$ corresponding to the state-action pair is obtained. Additionally, a mask flag is set to regulate the actual value estimation of the state $s_t$ . The aforementioned parameters are assembled into a tuple $\left(s_t, a_t, r\left(s_t, a_t\right), s_{t+1},\right.$ mask $)$ , which is stored in the experience replay buffer. Once the number of tuples stored in the replay buffer reaches a specified threshold, a mini-batch of tuples is randomly sampled from the buffer and used to update the parameters of both the Actor and Critic networks, thereby enabling the agent to learn the policy and value functions more effectively and improving the overall stability and efficiency of the training process.

The update of parameters in the Q Critic network is performed by minimizing the following loss function:

$$\text{Loss} = \frac{1}{|B|} \sum_{(s_t, a_t, r_{t+1}, s_{t+1,\text{mask}}) \in \mathcal{D}} \left[\mathrm{q}_i\left(s_t, a_t\right) - \mathrm{Q_t}\right]^2 \tag{3}$$

where $\mathrm{q}_i\left(s_t, a_t\right)$ (i=1,2) denotes the predicted value of state $s_t$, $Q_t$ represents the true value of state $s_t$, $\mathcal{D}$ signifies the experience replay buffer, and $|B|$ indicates the number of tuples sampled from the experience replay buffer $\mathcal{D}$.

The derivation formula for $Q_t$ is as follows:

$$a_{t+1}^* = \arg\max_{a_{t+1}} \pi\left(a_{t+1} \mid s_{t+1}\right) \tag{4}$$

$$\begin{aligned} Q_{\text{target}}\left(s_{\mathrm{t}+1}\right) = & \min\left(q_{\text{target 1}}\left(s_{\mathrm{t}+1}, a_{t+1}\right), q_{\text{target 2}}\left(s_{\mathrm{t}+1}, a_{t+1}\right)\right) \\ & - \alpha \log \pi\left(a_{t+1}^* \mid s_{t+1}\right) \end{aligned} \tag{5}$$

$$\mathrm{Q_t} = r\left(s_t, a_t\right) + \text{mask} * \gamma * Q_{\text{target}}\left(s_{\mathrm{t}+1}\right) \tag{6}$$

Update the parameters of the Q Critic target network through soft updates:

$$\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i \quad (i = 1, 2) \tag{7}$$

where $\theta_i$ represents the parameters in the Q Critic network, $\bar{\theta}_i$ denotes the parameters in the Q Critic target network, and $\tau$ is the proportional coefficient with a range of [0, 1].

The parameters of the Actor network are updated by minimizing the following loss function:

$$\begin{aligned} \text{Loss}^* = & \frac{1}{|B|} \sum_{(s_t, a_t, r_{t+1}, s_{t+1}, \text{ mask }) \in \mathcal{D}} \\ & \left[\alpha \log \pi\left(a_t^{\text{new}} \mid s_t\right) - Q\left(s_t, a_t^{\text{new}}\right)\right] \end{aligned} \tag{8}$$

The calculation formula for $Q\left(s_{\mathrm{t}}, a_t^{\text{new}}\right)$ is given by:

$$Q\left(s_{\mathrm{t}}, a_t^{\text{new}}\right) = \min\left(q_1\left(s_{\mathrm{t}}, a_t^{\text{new}}\right), q_2\left(s_{\mathrm{t}}, a_t^{\text{new}}\right)\right) \tag{9}$$

where $a_t^{new}$ represents the action obtained through the policy network.

The update of the temperature coefficient $\alpha$ is achieved by minimizing the following loss function:

$$\text{Loss}_\alpha = -\frac{1}{|B|} \sum_{t=0}^{T} \alpha\left(\log \pi\left(a_t^{new} \mid s_t\right) + H^*\right) \tag{10}$$

$$H^* = -\prod_{i=1}^{n} \text{Dimension of the action space } [i] \tag{11}$$

## 3. Input and Output of the Actor Network

Motion planning for spherical multi-telescopic-legged robots in three-dimensional space is significantly more complex than in planar environments. Due to their omnidirectional locomotion capability and deformable structure, the state representation of such robots is considerably more intricate than that of conventional mobile robots. To enable the agent to make appropriate action decisions, the input state is designed to comprehensively capture both environmental and robot-specific information. Specifically, the state space $S$ is defined as

$$\mathrm{S} = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\} \tag{12}$$

where

$$s_1 = \{\Delta x_1, \Delta y_1, \ldots, \Delta x_i, \Delta y_i\} \tag{13}$$

denotes the relative positions of obstacles with respect to the robot, with $\Delta x_i$ and $\Delta y_i$ representing the coordinate differences in the $x$ and $y$-directions between the $i$-th obstacle and the robot center;

$$s_2 = \{\Delta x_t, \Delta y_t\} \tag{14}$$

denotes the relative position of the target with respect to the robot, with where $\Delta x_t$ and $\Delta y_t$ denote the coordinate differences in the $x$- and $y$-directions, respectively, between the target position and the robot center.

$$s_3 = \{l_1, l_3, \ldots, l_{23}\} \tag{15}$$

represents the extension lengths of the telescopic joints. The robot has 24 joints, of which the odd-numbered joints are motor-driven, and the elongations of these joints are recorded as part of the system state.

$$s_4 = \{v_1, v_2, \ldots, v_{12}\} \tag{16}$$

represents the velocities of the telescopic joints;

$$s_5 = \{x_0, y_0, z_0\} \tag{17}$$

denotes the position coordinates of the robot body center;

$$s_6 = \{\alpha, \beta, \gamma\} \tag{18}$$

denotes the orientation of the robot body center expressed in Euler angles (yaw, pitch, and roll);

$$s_7 = \{v_x, v_y, v_z\} \tag{19}$$

represents the linear velocity of the robot body center.

Based on these inputs, the Actor network is responsible for selecting the corresponding actions. The spherical multi-telescopic-legged robot is equipped with a total of 12 legs, each featuring two sliding joints: a motor-driven joint and a belt-driven joint. The actions output by the Actor network represent the extension lengths for all 24 joints, with the extension lengths of the belt-driven joints being consistent with those of the motor-driven joints. Therefore, the output action A is defined as follows:

$$A = \{l_1, l_2, \ldots, l_{24}\} \tag{20}$$

## 4. Reward Function Setting

A reasonable reward function can guide the agent to learn the desired policy. The reward function used in this paper primarily consists of the following six components: out-of-bounds penalty $R_1$ , ground contact penalty for legs $R_2$ , penalties for approaching obstacles $R_3$ , rewards for approaching the target point $R_4$ , reward for reaching the target point $R_5$ , and penalty for colliding with obstacles $R_6$ .

(1) Out-of-bounds penalty $R_1$

If any foot-end apex of the 12 legs of the robot exceeds the defined boundary range, a negative reward value is obtained; if the robot remains within the boundary, the reward value is set to zero.

$$R_1 = \begin{cases} r_1, & \text{if exceeding the boundary} \\ 0 & \text{else} \end{cases} \tag{21}$$

where $r_1$ is an adjustable parameter.

(2) Ground contact penalty for legs $R_2$

We design a reward function based on the robot's ground contact legs to ensure that the motions in the simulation experiments are consistent with the actual motions of the prototype robot. There are two types of ground contact for robot legs: when the robot is in a stable state, three legs are in contact with the ground; whereas during rolling actions, two legs are in contact. Therefore, it is stipulated in the simulation experiments that the robot must keep two or three legs touching the ground during its motion in order to be realistic.

Specifically, if the robot does not meet the condition that two or three legs touch the ground, then it will receive a negative reward value. Conversely, if the robot satisfies this condition, it will receive a reward value of zero.

$$R_2 = \begin{cases} r_2, & \text{if it is neither 2 nor 3} \\ 0 & \text{else} \end{cases} \tag{22}$$

where $r_2$ is an adjustable parameter.

(3) Penalties for approaching obstacles $R_3$

The design of this part of the reward function is important for the robot to achieve precise obstacle avoidance. Inspired by the artificial potential field method, the potential energy acting on the robot is designed such that it increases as the robot approaches an obstacle and decreases as it moves away.

Within a single time step, let the position of the robot center at time $t$ be denoted as $(x_t, y_t, z_t)$, and at time $t+1$ as $(x_{t+1}, y_{t+1}, z_{t+1})$. For a given obstacle, let its centroid be $(x_{obs}, y_{obs}, z_{obs})$. Then, the potential energy acting on the robot at time $t$ and $t+1$ can be computed as:

$$U_{\text{obs}}(t) = M\sqrt{(x_t - x_{obs})^2 + (y_t - y_{obs})^2 + (z_t - z_{obs})^2} \tag{23}$$

$$U_{\text{obs}}(t+1) = M\sqrt{(x_{t+1} - x_{obs})^2 + (y_{t+1} - y_{obs})^2 + (z_{t+1} - z_{obs})^2} \tag{24}$$

where $M$ is a negative adjustable parameter.

Thus, the calculation method for $R_3$ during the time step from $t$ to $t+1$ is as follows:

$$R_3 = U_{\text{obs}}(t) - U_{\text{obs}}(t+1) \tag{25}$$

(4) Rewards for approaching the target point $R_4$

Based on the concept of the artificial potential field method, the target point is modeled as an attractive source that exerts gravitational potential energy on the robot. Specifically, the closer the center of the robot is to the target point, the smaller the corresponding gravitational potential energy, indicating that the robot is approaching the desired goal state. Conversely, the farther the robot's center is from the target point, the greater the gravitational potential energy, reflecting a larger deviation from the target location. In this way, the variation in gravitational potential energy can be used to characterize the relative positional relationship between the robot and the target point, thereby providing an effective basis for reward design and motion planning. The calculation formula for the gravitational potential energy at time $t$ is given as follows:

Let the coordinates of the target point $T$ be denoted by $(x_{tar}, y_{tar}, z_{tar})$, and let the position of the robot center at time $t$ be represented as $(x_t, y_t, z_t)$. Then,

$$U_{\text{p}}(t) = C\sqrt{(x_t - x_{tar})^2 + (y_t - y_{tar})^2 + (z_t - z_{tar})^2} \tag{26}$$

where $C$ is a positive adjustable parameter.

Similarly, the calculation formula for the gravitational potential energy at time $t+1$ is as follows:

After undergoing a change over one time step, let the coordinates of the robot's center be represented as $\mathrm{O}^*(x_{t+1}, y_{t+1}, z_{t+1})$. Then,

$$U_{\text{p}}(t+1) = C\sqrt{(x_{t+1} - x_{tar})^2 + (y_{t+1} - y_{tar})^2 + (z_{t+1} - z_{tar})^2} \tag{27}$$

Thus, the calculation method for $R_4$ during the time step from $t$ to $t+1$ is as follows:

$$R_4 = U_{\text{p}}(t) - U_{\text{p}}(t+1) \tag{28}$$

(5) Reward for reaching the target point $R_5$

When the Euclidean distance between the robot's center position and the target point falls below a predefined threshold, the robot is considered to have successfully reached the target. In this case, a designated reward is assigned to explicitly encourage goal-directed behavior and to reinforce successful task completion during training. Conversely, if the distance remains greater than the threshold, the robot is regarded as having not yet accomplished the task, and no terminal reward is provided, that is, the reward value is set to zero. This reward design enables the agent to clearly distinguish successful states from unsuccessful ones, thereby facilitating the learning of effective

motion planning strategies.

$$R_5 = \begin{cases} r_5, & \text{if reaching the target point} \\ 0 & \text{else} \end{cases} \tag{29}$$

(6) Penalty for colliding with obstacles $R_6$

When the robot collides with obstacles similar in size to itself or large-sized obstacles, it obtains a negative reward value; when the robot makes contact with non-threatening obstacles, the reward value is set to zero.

$$R_6 = \begin{cases} r_6, & \text{if a collision occurs} \\ 0 & \text{else} \end{cases} \tag{30}$$

The total reward value $R_{\text{total}}$ obtained by the robot from time step t to t+1 is the sum of the six types of reward values mentioned above.

$$R_{\text{total}} = R_1 + R_2 + R_3 + R_4 + R_5 + R_6 \tag{31}$$

The specific settings of the tunable parameters involved above are provided in Table 1 in the subsequent simulation experiments.

**Table 1.** Tunable Parameter Configuration

| Parameters | Value |
|---|---|
| $r_1$ | $-500$ |
| $r_2$ | $-20$ |
| $M$ | 100 |
| $C$ | 500 |
| $r_5$ | 500 |
| $r_6$ | $-50$ |

## 5. Design of the Look-Ahead-Look-Back HER (LALB-HER) Algorithm

Hindsight Experience Replay (HER) substantially enhances sample utilization during training through the mechanism of goal relabeling. However, the widely used "future state" relabeling strategy, which only relabels goals using future states from the current episode, fails to fully exploit the transferable information embedded in historical experiences. Similarly, the "episode" strategy performs random sampling solely within the current episode, without considering the potential of leveraging cross-episode historical experiences. In practice, the rich history of state transitions accumulated during reinforcement learning contains a wealth of reusable success patterns and failure lessons, providing significant value for cross-episode knowledge transfer.

To address this limitation, this paper proposes a state-similarity-based Look-Ahead-Look-Back HER (LALB-HER) algorithm, which simultaneously incorporates future states from the current episode and goal states of historically similar states as dual sources for goal sampling, thereby achieving two-fold experience augmentation.

The core idea of the proposed method is to extend the standard two-stage processing framework of the future strategy by introducing a cross-episode historical experience mining module. Specifically, after each episode, the algorithm not only performs conventional future-state goal relabeling on the current trajectory, but also retrieves historically similar states from the experience replay buffer. Furthermore, the goal information embedded in these similar states is leveraged to relabel goals in the reconstructed experiences, thereby generating additional training samples. This approach enables dual-dimensional experience reuse—horizontally within the current episode and vertically across historical episodes. Consequently, it significantly improves data utilization efficiency and enhances policy learning performance.

The sampling principle is schematically illustrated in Figure 1, while the complete algorithmic pseudocode is detailed in Algorithm 1.

The similarity criterion $\text{Sim}(s_t, s_h)$ is defined as follows. Given the current state $s_t$ during the backtracking process and an initial state $s_h$ sampled from historical experience, the three-dimensional coordinates of the robot body center are extracted from both states. The Euclidean distance $d$ between these two positions is then computed. If $d$ is less than or equal to a predefined threshold $\tau$, the two states are considered similar. Based on the overall scale of the environment and the structural dimensions of the robot, the threshold $\tau$ is set to 0.05 m.

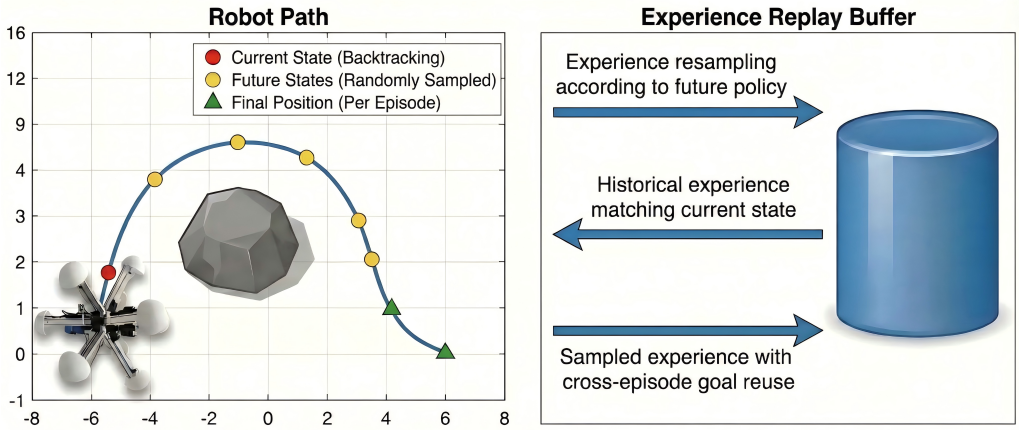



**Figure 1.** Dual-dimensional sampling principle.

**Algorithm 1** LALB-HER-SAC

Initialize Actor network parameters $\phi$, Q critic network parameters $\theta$, Q critic target network parameters $\bar{\theta}$, and temperature coefficient $\alpha$
Initialize an empty replay buffer $\mathcal{D}$
**for** episode = 1 to $M$ **do**
    Receive initial state $s_t$
    Initialize an empty historical database $\mathcal{H}$
    **for** step = 1 to $T$ **do**
        Take $a_t = \pi_\phi(s_t)$
        Execute actions $a_t$ and observe a new state $s_{t+1}$
        Compute reward $r_t = R(s_t, a_t, s_{t+1})$
        Store transitions $(s_t, a_t, r_t, s_{t+1}, mask)$ in $\mathcal{D}$
        Store transitions $(s_t, a_t, r_t, s_{t+1}, mask)$ in $\mathcal{H}$
        **if** $|\mathcal{D}| > batchsize$ **then**
            Randomly sample a minibatch B from the replay buffer $\mathcal{D}$
            Perform one step of optimization using minibatch B
        **end if**
        Set $s_t = s_{t+1}$
    **end for**
    **for** t = 1 to $T-1$ **do**
        Sample $k' = \min(k, |T| - t)$ future indices: $I \sim \{t+1, \ldots, |T|\}$ with size $k'$
        **for** each index $j$ in $I$ **do**
            Computer $g_{\text{future}} \in H[j]$ ▷ $g_{\text{future}}$ represents the initial state in $H[j]$
            $s_t, s_{t+1} \leftarrow g_{\text{future}}$ ▷ Reshape the target point
            $r_{\text{new}} = R(s_t, a_t, s_{t+1})$
            Store transitions $(s_t, a_t, r_{\text{new}}, s_{t+1}, mask)$ in $\mathcal{D}$
        **end for**
        **if** $|\mathcal{D}| > k_h$ **then**
            Random sample $k_h$ historical data from $\mathcal{D}$ : $D_h$
            Initialize counter $cnt = 0$
            **for** each $s_h \in D_h$ **do**
                **if** $\text{Sim}(s_t, s_h) \leq \tau$ and $cnt < k$ **then**
                    Computer $g_h \in s_h$ ▷ $g_h$ denotes the target point positions
                    $s_t, s_{t+1} \leftarrow g_{\text{h}}$ ▷ Reshape the target point
                    $r_h = R(s_t, a_t, s_{t+1})$
                    Store transitions $(s_t, a_t, r_h, s_{t+1}, mask)$ in $\mathcal{D}$
                    $cnt = cnt + 1$
                **else if** $cnt \geq k$ **then**
                    break
                **end if**
            **end for**
        **end if**
    **end for**
**end for**

For similar states, the target information contained in $s_h$ can be reused as the relabeled goal for both $s_t$ and $s_{t+1}$. By recomputing the state representations of $s_t$ and $s_{t+1}$ with respect to the relabeled goal, a new transition is constructed together with the remaining elements of the original experience and then stored in the replay buffer. Compared with directly reusing historical experiences, this approach effectively mitigates the suppression of exploration caused by excessive duplication of experiences and prevents overfitting, thereby improving the generalization capability of the model. Moreover, the equivalent experiences generated based on the state similarity criterion provide essential historical exploration data to support reinforcement learning.

It should be noted that only the three-dimensional position variables are used to compute the Euclidean distance in the similarity measurement. This design aims to reduce computational complexity, minimize the time overhead introduced by similarity screening, and avoid imposing additional computational burden on the overall training process.

## 6. Experiments

### *6.1. Simulation Platform Setup and Parameter Configuration*

The simulation experiments in this study were conducted on the PyBullet platform, an open-source real-time physics engine based on Bullet and widely used in robot control, reinforcement learning, multibody dynamics, and contact mechanics. PyBullet provides efficient collision detection, dynamics solving, and rigid-body constraint handling, while supporting seamless integration with mainstream deep learning frameworks such as PyTorch and TensorFlow through a concise Python interface. It also supports GPU-accelerated simulation and offers OpenAI Gym-compatible interfaces, enabling reinforcement learning agents to directly perform state acquisition, action execution, and reward computation.

The parameters involved in the experiment are listed in Table 2.

**Table 2.** Simulation Experiment Parameter Settings.

| Parameter | Value |
|---|---|
| Learning rate | 0.0001 |
| Discount factor | 0.99 |
| Optimizer | Adam |
| Batch size | 256 |
| Replay buffer size | 1,000,000 |
| Soft update coefficient | 0.005 |
| Total episodes | 40,000 |
| Steps per episode | 500 |
| Friction type | Sliding friction |
| Joint force | 30 N |

### *6.2. Training Experimental*

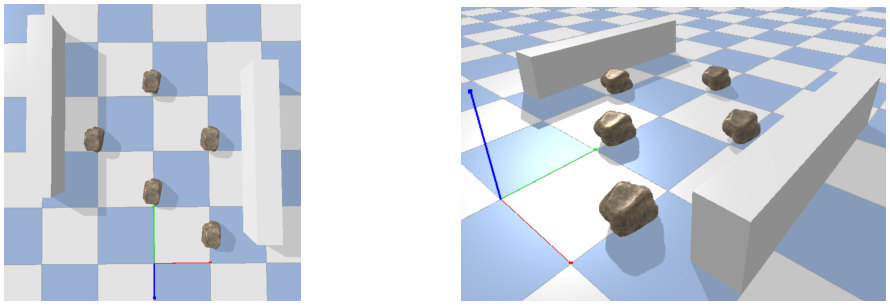
As shown in Figure 2, the training environment consists of five irregular stones comparable in size to the robot and two large obstacles measuring 2.9 m × 0.4 m × 0.6 m. On this basis, a multi-obstacle environment was constructed to evaluate the algorithm's motion planning performance and the trained model's generalization across different goals and environments.

During training, the target position for the agent is fixed at $(-1.02, 3.5, 0)$, which serves as the primary exploration objective. This setting is intended to provide a stable reward signal in the early stage of training, thereby guiding the learning process toward convergence. If the target position were changed at every episode, the experience collected in previous episodes would quickly become invalid, making it difficult for the model to converge.

To enhance exploration in the initial stage, the agent performs random actions during the first 30 episodes. In this phase, interaction data are collected and stored in the replay buffer without updating the model parameters, and the LALB-HER mechanism is not activated for resampling.

After the exploration phase, the standard training procedure is initiated. The LALB-HER mechanism is enabled, and mini-batches are sampled from the replay buffer to update the model parameters.

We conducted training using different combinations of $k$ and $k_h$, and for each combination, we performed three independent runs to evaluate performance. The average training convergence time and



(a) (b)

**Figure 2.** Training experimental environment. (**a**) Top view of the simulation environment. (**b**) 45-degree forward-tilted view of the simulation environment.

standard deviation for each combination are shown in Table 3. As the table indicates, increasing $k_h$ leads to a significant rise in overall training time. This is because a larger amount of data sampled from the replay buffer increases the worst-case time complexity during similar-state matching, resulting in longer training durations.

Interestingly, adjusting $k$ does not lead to a proportional increase in training time; in particular, when $k = 10$, the training duration is shorter compared to other values. The reason is that $k = 10$ provides a suitable balance for extracting historical experiences, allowing the model to achieve a proper ratio between reinforced historical data and normal training experience. When $k$ is too small, the proportion of effective historical experience is insufficient, limiting the model's generalization ability and leading to poorer per-episode performance. Conversely, when $k$ is too large, both the computational cost increases and an excess of historical experience may cause the model to deviate from the initially intended training objectives.

Therefore, setting $k = 10$ is a reasonable choice. For $k_h$, it should be kept as small as possible while remaining larger than $k$, leading us to select $k_h = 30$. Based on these results, we conclude that the combination of $k = 10$ and $k_h = 30$ represents an optimal parameter setting. All subsequent experiments are conducted using this parameter configuration.

**Table 3.** Training Performance with Different Parameter Combinations.

| **Parameters** $(k, k_h)$ | **Training Time (h)** |
|---|---|
| (5, 30) | $35.37 \pm 0.55$ |
| (10, 30) | $32.67 \pm 1.24$ |
| (15, 30) | $42.13 \pm 1.06$ |
| (5, 40) | $43.07 \pm 1.29$ |
| (10, 40) | $38.31 \pm 0.61$ |
| (15, 40) | $49.66 \pm 0.59$ |
| (5, 50) | $52.93 \pm 1.63$ |
| (10, 50) | $46.77 \pm 1.15$ |
| (15, 50) | $61.33 \pm 1.16$ |

### *6.3. Test Experiment*

In the simulation-based training experiments, the agent's generalization performance was evaluated by introducing additional randomly generated target locations that avoided obstacles, beyond the predefined initial targets. Specifically, the agent was tested on 100, 1000, and 2000 sampled target points. The results show that it successfully reached most targets, except for a small number located far from its initial position or common trajectories. These unreachable targets were generally distributed in regions that received limited exploration during training, resulting in weaker local generalization. Overall, the model demonstrated strong generalization across most target locations in the environment.

During testing, an episode terminated immediately when the agent reached the target, and the number of steps taken was recorded, enabling the calculation of the average and maximum steps required to reach the targets. The motion planning success rates and corresponding step statistics for different target set sizes are summarized in Table 4. Representative motion processes for two distinct target tasks are illustrated in Figure 3 and Figure 4, respectively.

**Table 4.** Large-scale Generalization Test Results.

| Test Scale | Success Rate (%) | Average Steps | Max Steps |
|---|---|---|---|
| 100 targets | 97.0 | 40.1 | 76 |
| 1000 targets | 95.5 | 42.3 | 92 |
| 2000 targets | 94.2 | 45.6 | 101 |

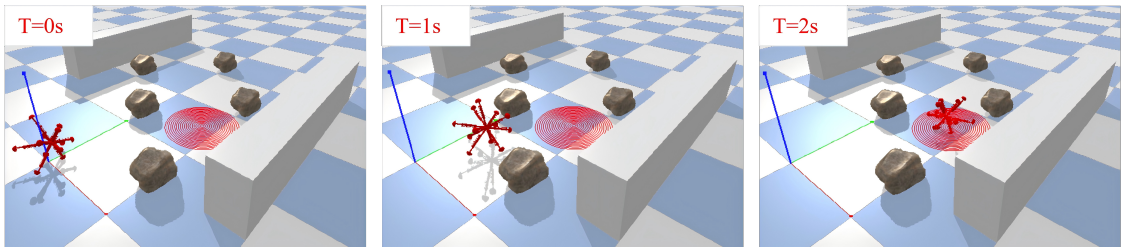



**Figure 3.** Robot motion in the first test experimental environment (target 1).

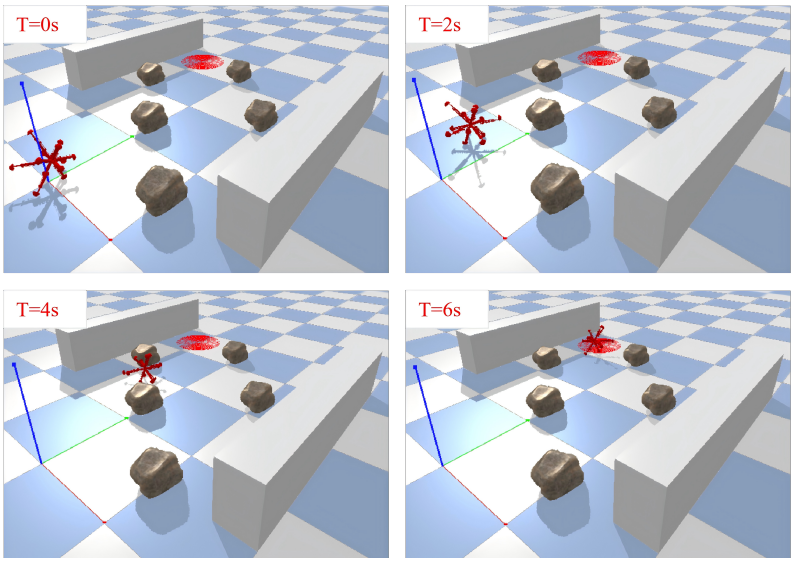
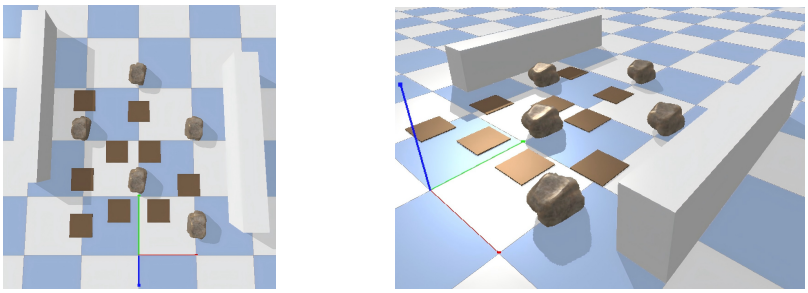



**Figure 4.** Robot motion in the first test experimental environment (target 2).

To evaluate the model's generalization ability across different environments, we constructed a second testing scenario, as illustrated in Figure 5. In contrast to the training simulation environment, this test setting incorporates nine wooden planks, each 5 cm in height, resulting in a more rugged terrain with higher obstacle density. This configuration is designed to examine whether the trained robot can still accomplish the task when confronted with substantial environmental variation. For the sake of comparative analysis, we selected target positions identical to those in Figures 3 and 4 and observed the corresponding robot motion in the new environment. The results indicate that, despite the environmental changes, the model can still successfully guide the robot to the targets. Figures 6 and 7 illustrate the robot's motion process in the new environment. Although the time required slightly increases, it remains within an acceptable range. Through this experiment, we validate the cross-environment generalization capability of the learned policy model.

To systematically analyze the contribution of the proposed algorithm, a series of ablation experiments were conducted. First, SAC combined with the HER mechanism was adopted as the baseline, and its training efficiency and performance were evaluated. Subsequently, the proposed LALB-HER method was incorporated into the SAC framework for comparative analysis. To further validate the superiority of the proposed method, the reinforcement learning algorithm Twin Delayed Deep Deterministic Policy Gradient (TD3), which follows a different policy paradigm, was selected as an additional baseline and also combined with HER. Its performance, as well as that of TD3 integrated with LALB-HER, was evaluated. The comparative results are summarized in Table 5. It should be noted that the success rates reported in Table 3 were obtained under the same parameter settings, based on 2000 randomly generated target points.



(a) (b)

**Figure 5.** The second test experimental environment. (**a**) Top view of the test environment. (**b**) 45-degree forward-tilted view of the test environment.

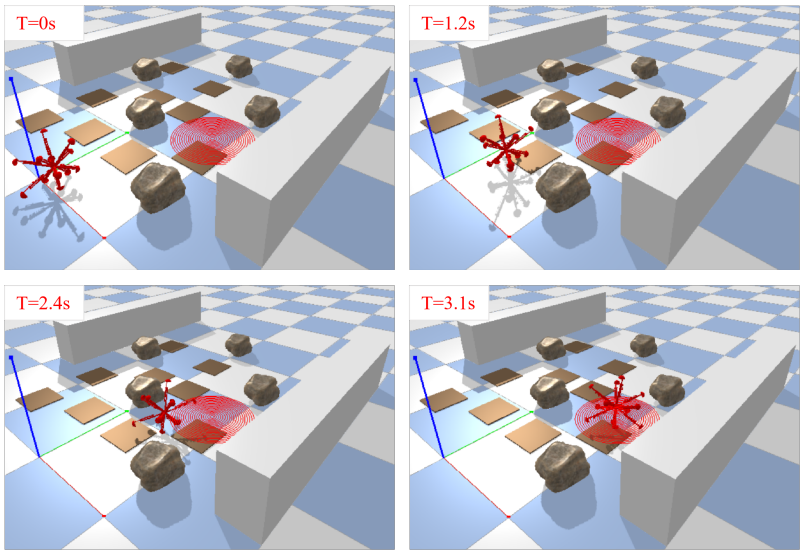



**Figure 6.** Robot motion in the second test experimental environment (target 1).

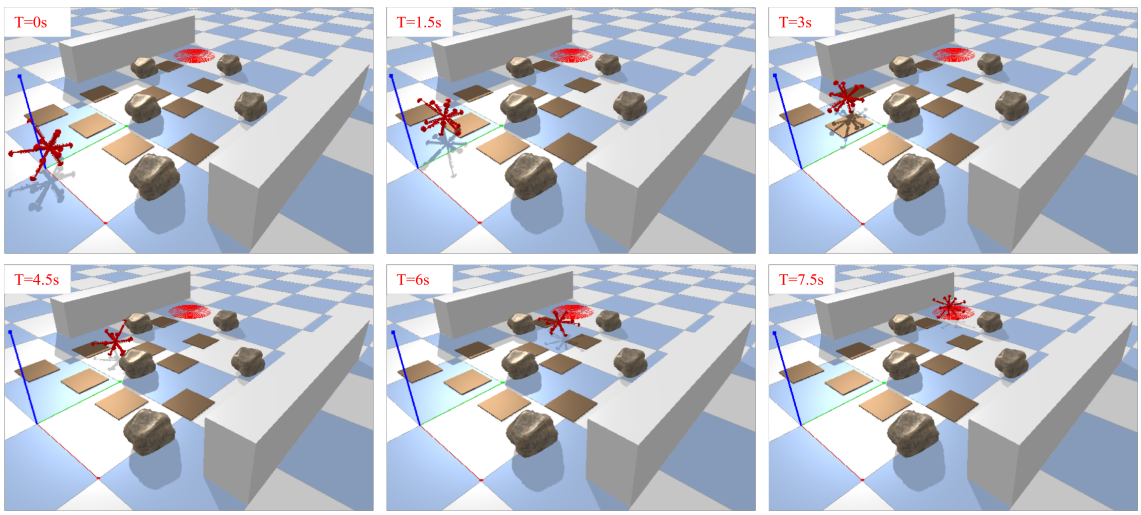



**Figure 7.** Robot motion in the second test experimental environment (target 2).

To analyze the variation of reward values during training and to more clearly illustrate the convergence performance and stability of the proposed method, the average reward over the current episode and the preceding 100 episodes is computed at the end of each training episode. The resulting curves of average reward for different algorithms during the training process are shown in Figure 8.

The simulation results indicate that the proposed SAC+LALB-HER method significantly outperforms the SAC algorithm combined with the HER mechanism in terms of training efficiency, while also surpassing both TD3 integrated with HER and TD3 combined with LALB-HER. Overall, the proposed algorithm demonstrates clear

improvements over deterministic policy reinforcement learning methods such as TD3, in terms of both training efficiency and final model performance.

**Table 5.** Algorithm Comparison and Ablation Study Results.

| Algorithm | Success Rate (%) |
|---|---|
| SAC+HER (Baseline) | 86.1 |
| SAC+LALB-HER | 94.2 |
| TD3+HER | 84.9 |
| TD3+LALB-HER | 92.7 |

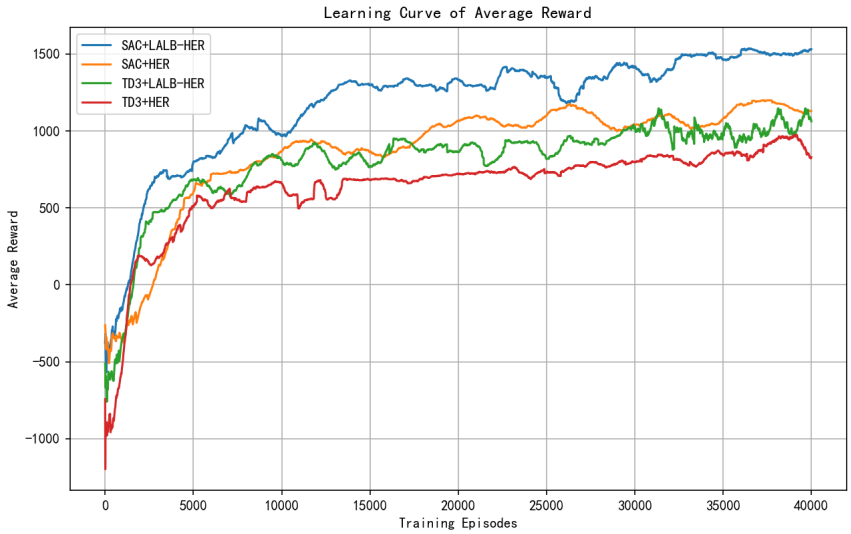



**Figure 8.** Learning curve of average reward.

## 7. Conclusions

This study proposes a novel algorithm aimed at effectively mitigating the common issue of low sample efficiency in reinforcement learning. The algorithm significantly improves training efficiency while ensuring the model's generalization capability in both unseen targets and novel environments. After training, the model can autonomously generate appropriate action sequences for different target locations, accurately guiding the robot to reach the desired goal. Through systematic parameter tuning experiments, we further identified the optimal configuration that achieves a balance between training efficiency and final performance.

In addition, we conducted comprehensive ablation studies, comparing the proposed method with mainstream deterministic policy reinforcement learning algorithms, including TD3+HER and TD3 combined with the proposed LALB-HER, as well as SAC+HER. Experimental results demonstrate that the model based on the SAC framework integrated with the proposed approach exhibits clear advantages in sample efficiency, convergence speed, and final success rate, thereby validating the effectiveness of the proposed algorithmic framework.

Looking forward, we plan to conduct further investigations with increased training resources. Beyond the current generalization experiments involving variations in target positions and the addition of perturbation obstacles, future work will extend to more challenging scenarios involving overall structural changes in the environment. This will further enhance the model's adaptability in dynamic and diverse environments, providing a solid foundation for deployment in complex and rapidly changing real-world scenarios.

## Author Contributions

X.L.: methodology, writing—original draft preparation; F.X.: methodology, data curation; X.Z.: writing—reviewing and editing. All authors have read and agreed to the published version of the manuscript.

## Funding

This research received no external funding.

## Data Availability Statement

The authors confirm that the data supporting the findings of this study are available within the article.

## Acknowledgments

The authors gratefully acknowledge anonymous editors and reviewers.

## Conflicts of Interest

The authors declare no conffict of interest.

## Use of AI and AI-Assisted Technologies

No AI tools were utilized for this paper.

## References

1. Ortigoza, R.S.; Marcelino-Aranda, M.; Ortigoza, G.S.; et al. Wheeled mobile robots: a review. *IEEE Lat. Am. Trans.* **2012**, *10*, 2209–2217.
2. Chan, R.P.M.; Stol, K.A.; Halkyard, C.R. Review of modelling and control of two-wheeled robots. *Annu. Rev. Control.* **2013**, *37*, 89–103.
3. Chung, W.; Iagnemma, K. Wheeled robots. In *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 575–594.
4. Zhao, D.; Revzen, S. Multi-legged steering and slipping with low DoF hexapod robots. *Bioinspir. Biomimetics* **2020**, *15*, 045001.
5. Wen, Q.; He, J.; Gao, F. Kinematic design of a novel Multi-legged robot with Rigid-flexible coupling grippers for asteroid exploration. *Robotica* **2022**, *40*, 3699–3725.
6. Peng, S.; Ding, X.; Yang, F.; et al. Motion planning and implementation for the self-recovery of an overturned multi-legged robot. *Robotica* **2017**, *35*, 1107–1120.
7. He, X.; Huo, J.; Lin, R. Multi Spherical Robot Control System for Nuclear Radiation Leak Detection. *Intell. Control.* **2025**, *1*, 2.
8. Liu, H.; Ying, F.; Jiang, R.; et al. Obstacle-Avoidable Robotic Motion Planning Framework Based on Deep Reinforcement Learning. *IEEE/ASME Trans. Mechatronics* **2024**, *29*, 4377–4388.
9. Wang, H.; Gao, W.; Wang, Z.; et al. Research on Obstacle Avoidance Planning for UUV Based on A3C Algorithm. *J. Mar. Sci. Eng.* **2023**, *12*, 63.
10. Tang, W.; Wu, F.; Lin, S.W.; et al. Causal deconfounding deep reinforcement learning for mobile robot motion planning. *Knowl.-Based Syst.* **2024**, *303*, 112406.
11. Mnih, V.; Kavukcuoglu, K.; Silver, D.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.
12. Xu, Q.; Zhang, T.; Zhou, K.; et al. Active Collision Avoidance for Robotic Arm Based on Artificial Potential Field and Deep Reinforcement Learning. *Appl. Sci.* **2024**, *14*, 4936.
13. Choi, J.; Lee, G.; Lee, C. Reinforcement learning-based dynamic obstacle avoidance and integration of path planning. *Intell. Serv. Robot.* **2021**, *14*, 663–677.
14. Hart, F.; Okhrin, O. Enhanced method for reinforcement learning based dynamic obstacle avoidance by assessment of collision risk. *Neurocomputing* **2024**, *568*, 127097.
15. Andrychowicz, M.; Wolski, F.; Ray, A.; et al. Hindsight experience replay. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
16. Dong, M.; Ying, F.; Li, X.; et al. Efficient policy learning for general robotic tasks with adaptive dual-memory hindsight experience replay based on deep reinforcement learning. In Proceedings of the 7th International Conference on Robotics, Control and Automation (ICRCA), Taizhou, China, 5–7 January 2023; pp. 62–66.
17. He, Q.; Zhuang, L.; Li, H. Soft hindsight experience replay. *arXiv* **2020**, arXiv:2002.02089.
18. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1587–1596.
19. Haarnoja, T.; Zhou, A.; Hartikainen, K.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.