Received 2 November 2023, accepted 3 December 2023, date of publication 7 December 2023,
date of current version 18 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3340684

RESEARCH ARTICLE

Robotic Arm Trajectory Planning Method Using Deep Deterministic Policy Gradient With Hierarchical Memory Structure

DI ZHAO[1,2,3,4], (Member, IEEE), ZHENYU DING[5], WENJIE LI[1], SEN ZHAO[1], AND YUHONG DU[6]

[1]School of Mechanical Engineering, Tiangong University, Tianjin 300387, China
[2]Engineering Teaching Practice Training Center, Tiangong University, Tianjin 300387, China
[3]State Key Laboratory of Turbulence and Complex Systems, College of Engineering, Peking University, Beijing 100871, China
[4]Intelligent Bionic Design Laboratory, Peking University, Beijing 100871, China
[5]School of Electronics and Information Engineering, Tiangong University, Tianjin 300387, China
[6]Innovation College, Tiangong University, Tianjin 300387, China

Corresponding author: Yuhong Du (duyuhong@tiangong.edu.cn)

This work was supported in part by the Tianjin Science and Technology Bureau under Grant 20YDTPJC00740, and in part by the Ministry of Education of the People's Republic of China under Grant 220603032241503.

ABSTRACT Traditional robotic arm path planning methods are mainly carried out in the tool center point operation space, and frequently solve inverse kinematics problems, thus consuming a large number of computational resources. In contrast, the use of positive kinematics for planning in joint space not only enhances efficiency, but also provides analytic solutions with higher accuracy. In order to better cover the environmental state space, this paper adopts the full-preserving experience preservation approach. In order to realize fast and efficient sampling of high reward value experience, this study constructs an innovative hierarchical memory structure and eliminates the overfitting phenomenon that may be caused by biased sampling through the Bias-Free strategy. Experimentally validated in the continuous path planning task of a textile robot arm, the proposed hierarchical memory deep deterministic gradient strategy method (HM-DDPG) demonstrates excellent performance and practicality in the textile robot arm path planning problem. This method offers an efficient and robust solution for handling complex tasks with temporal dependencies, paving the way for future innovations and applications in industrial fields such as textiles.

INDEX TERMS Continuous control task, deep deterministic policy gradient, path planning, positive kinematics, hierarchical memory.

I. INTRODUCTION

The use of robotic arm in textile industry environments has become a key technology, especially in the area of path planning. Path planning involves the generation of a trajectory for the movement of a robotic arm from a starting point to a target point, which is directly related to the efficiency and safety of the operation. Effective path planning methods have become critical as the textile industry continues to increase the demand for high precision, high speed and complexity of operations in complex environments.

Conventional path planning methods mainly focus on Tool Center Point (TCP) operation space and frequently solve inverse kinematics (IK) problems, thus leading to a large consumption of computational resources. This approach, although effective in some cases, may encounter difficulties in the complex and dynamic environment of textile robotic arm. As deep reinforcement learning (DRL) has achieved significant results in several fields, its introduction into textile robotic arm path planning is expected to address some of the challenges in traditional approaches, thereby improving the overall performance of path planning. The ability of DRL to generate more optimized paths by continuously learning and adapting to the environment opens up new possibilities for textile robotic arm path planning.

The associate editor coordinating the review of this manuscript and approving it for publication was Guilin Yang.

Experience replay is an important component of DRL, including experience retention and experience sampling. Experience replay uses batch offline updates, which has three advantages over online updates: 1) Experience retention allows experience to be learned repeatedly, avoiding catastrophic forgetting; 2) The experience in the memory has strong correlation in the time series, and sampling can eliminate the correlation in the experience series and reduce the variance of the gradient update; 3) The distribution of the states in the memory changes with the optimization of the strategy, and sampling can effectively smooth the data distribution in the experience, avoiding the parameters from falling into the local optimum, or even oscillating or diverging.

In path planning applications of DRL, the Experience replay mechanism mainly involves two aspects: experience retention and experience sampling. There are two modes of experience retention, first-in-first-out (FIFO) and all-retention: the former deposits the interaction experiences into a fixed-capacity experience pool and overwrites the old data in a chronological order in order to mainly learn the frequently occurring states under the current policy; the latter preserves all the interaction experiences with the environment and ensures that the distribution of the pooled states extensively covers the environment state space, thus increasing the diversity of the experiences. However, both approaches face the problems of low sample utilization and high computational overhead, which limit their applications in high computational complexity tasks. Meanwhile, experience sampling methods include random sampling and prioritized sampling: random sampling does not take into account the priority of experiences, while prioritized sampling selects each experience according to its priority, but this tends to destroy the original distribution of the experience pool and increase the variance and instability of the algorithm. Combining these factors, it becomes crucial to design an Experience replay mechanism that can effectively balance sample diversity, computational overhead, and algorithmic stability.

And the reward setting is crucial for Agent's exploration in the environment. A well-set reward function can avoid the situation where the Agent obtains sparse rewards during exploration and fails to achieve the desired goal. For the task of trajectory planning in the robotic arm scenario, the agent must discover a long list of "correct" actions in order to generate positive rewards, which causes very large reward fluctuations, resulting in the learning algorithm taking a long time to converge or even failing to converge.

The aim of this study is to provide a data-driven approach to robotic arm joint space path planning. In order to achieve fast and efficient sampling of high-reward-value experiences, we construct an innovative hierarchical memory structure by dividing the original memory into high-reward and low-reward values through our hierarchical strategy, adjusting the ratio to allow it to prioritize sampling of the high-reward values during the initial learning, and then adjusting the sampling ratio to achieve unbiased sampling through the Bias-Free strategy. Further, since the state-action distribution in the FIFO memory approximates the state distribution in the current strategy, which does not cover the environment state space well and the overall explorability is weak; while the state distribution in the full-preservation memory can cover the environment state space better, this paper designs a stratified full-preservation memory, which ensures the diversity of experience and improves the sampling efficiency.

The main contributions of this paper are as follows:

(1) Path planning for robotic arm in conjunction with positive kinematics;

(2) A fully-preserved hierarchical memory is used to store experience samples and classify experiences based on the reward values in the samples, which solves the learning inefficiency caused by sparse rewards;

(3) Sampling the hierarchical memory by Bias-Free strategy for network parameter update of the deep strategic gradient algorithm to avoid overfitting;

(4) Experiments are conducted in a physical simulation scenario of a textile robotic arm, and the results show that compared with the baseline (randomly sampled DDPG) algorithm, the hierarchical empirical pooled deep deterministic gradient policy method (HM-DDPG) proposed in this paper can achieve better experimental results, and is also compared with the Soft Actor-Critic (SAC) and the proximal policy optimization (PPO) algorithms are compared to further prove the effectiveness of the proposed algorithm in this paper.

The structure of this paper is as follows: Section II presents a review of related studies on robotic arm trajectory planning methods. Section III introduces the foundational knowledge of the DDPG algorithm. Section IV defines the robotic arm trajectory planning task. Section V provides a detailed description of the algorithm. Section VI involves experimental validation and analysis. Finally, Section VII concludes the paper.

II. RELATED RESEARCH

Since the last century, the research on robotic arm trajectory planning has made remarkable progress. As early as in the 1980s, scholars proposed a method of trajectory planning using the cubic polynomial method in joint space [1]. Subsequently, Lin et al. experimentally proved the effectiveness of the cubic spline method for trajectory planning in joint space [2]. Since then, higher-order trajectory planning methods have emerged. For example, in 1987, Thompson and Patel proposed a method for local modification of trajectories using the B-spline method, which is highly efficient and occupies less computational resources [3]. Based on this, the development of the five-times B-spline function realizes the control of the trajectory starting and ending velocities [4].

Up to this point, more methods for trajectory planning of more complex robotic arm tasks have emerged. For example, in 2017, the A* algorithm was used by Italian scholars to realize the shortest path trajectory planning for a multi-degree-of-freedom robotic arm [5]. In 2019, the improved

A* algorithm was applied to the trajectory planning of a redundant robotic arm [6]. In terms of robotic arm obstacle avoidance, the Rapid-exploration Random Tree (RRT) algorithm has been proved to be very effective [7], [8]. RRT and its improved algorithms can be applied to the task of robotic arm obstacle avoidance planning for a variety of complex scenarios by exploring them in unstructured environments. For example, in 2006, RRT combined with an improved inverse kinematics solution method contributed to the obstacle avoidance capability of a robotic arm in the action space while performing the task of grasping an object [9], and in 2020, an improved RRT method was used for obstacle avoidance path planning of a snake robot [10].

In recent years, DRL as a method with exploration and learning ability, has been widely used in fields such as robotic arm control. However, DRL algorithms have many parameters and are difficult to tune [11], and there are problems such as low efficiency, high loss, not easy to model, and difficult to initialize the environment in physical training. There are large differences between simulation and the real world, and it is necessary to migrate the strategies in simulation to the real world, which brings some challenges to the application of reinforcement learning in the real world [12], [13]. In addition, today's DRL algorithms generally perform path planning for robotic arm in the operation space, which still requires the use of corresponding inverse kinematics computational methods. Today, scholars are still looking for effective methods to solve the above problems.

Mahmood et al. in 2018, while studying the problem of continuous control of robotic arm, proposed the promising application of model-free reinforcement learning [14] and used various algorithms including Trust Region Policy Optimization (TRPO) [15], Proximal Policy Optimization (PPO) [16], Soft Q-learning [17], and Deep Deterministic Policy Gradient (DDPG) [18]. They successfully realized the motion control of a 6-DOF robotic arm and conducted two types of experiments for the above four methods: 1) controlling some of the joints of the robotic arm to make the end of the robotic arm reach the target point, and 2) controlling all six joints to make the end of the robotic arm reach the target point. The results show that the fewer the controllable degrees of freedom, the better the learning results of the DRL algorithm. In other words, the more complex and structured the robotic arm's degrees of freedom are, the less likely the DRL algorithm is to learn an appropriate strategy.

DRL has also been studied in flexible robot control problems. For example, in 2021, American scholars [19] used five state-of-the-art DRL algorithms, including TRPO, PPO, DDPG, Twin Delayed DDPG (TD3) [20], and Soft Actor-Critic (SAC) [21], on the Elastica robot to successfully realize the soft robot control of the soft robot. The research team designed four scenarios to discuss the application of DRL to the distributed dynamic control of soft robots. The algorithms were able to reach convergence after 10 million training sessions.

Overall, DRL with its mechanism of exploration and learning, offers a wide range of applications for solving various problems such as robotic arm control. However, current research still faces some challenges, such as the large number of parameters, the difficulty of tuning parameters, and the problem of attrition in physical training. Future research needs to continue to explore effective methods to address these issues so that DRL can realize greater potential in practical applications. In addition, combining DRL with other advanced algorithms to improve the control performance of robotic arm in complex scenarios will be an important direction for future research.

III. BASIC GENERAL KNOWLEDGE

A. MARKOV DECISION PROCESS

Markov decision process (MDP) is the cornerstone of reinforcement learning, in which the interaction between reinforcement learning and optimal policy solving can be expressed in the form of probabilistic and mathematical formulas. MDP is a mathematical model of a sequential decision-making process, which can be represented by a quintuple (S, A, P, r, γ) The policy of the MDP model π is a state-to-action mapping that gives a probability distribution for each action in the set of actions based on the current state.

The Agent interacts with the environment according to its policy, generating a series of interaction sequences. The Agent then gathers experience from the interaction sequences and uses this experience to optimize its strategy with the goal of maximizing the expectation of cumulative returns, i.e.,

$$R = \sum_{t=0}^{T} \gamma^t r^t \tag{1}$$

where r is the reward value at round t, and γ represents the impact of future present decisions, i.e., the attenuation factor of the reward at each step that determines the long term of the gaze. After many iterations, the intelligent body can learn the optimal strategy to accomplish the corresponding tasks.

B. EXPERIENCE REPLAY

It is well known that DRL updates the policy network and the value network based on experiences collected from the environment. However, this process faces two main problems: first, the experiences are discarded as soon as they are used, which leads to their inefficient utilization; second, there are non-smooth distributions and strong correlations between the experiences, which make it difficult for the algorithms to converge stably during the training process.

To solve these problems, Mnih et al. introduced an experience replay mechanism in the DRL algorithm [22]. The experience transitions collected by the Agent are not directly used for training, but are stored in an experience buffer. Then, a certain number of experience transitions are extracted from the experience buffer for training by random sampling. The advantage of this approach is that the correlation between the experience transitions is eliminated, resulting in a smoother distribution of the data, which improves the efficiency of the utilization of the experience transitions.

C. DDPG ALGORITHM

The DDPG algorithm [18] is based on an Actor-Critic architecture that parameterizes a Critic function, respectively $Q(s, a)$ and an Actor function $\mu(s|\theta)$. The structure of the DDPG network is shown in Fig. 1, where first, actions are selected and rewarded through the policy network and transferred to the next state through state transfer. Second, the experience tuple is saved in the Experience replay buffer. Third, the experience is extracted from the Experience replay buffer for training. Fourth, the Q network is updated. Finally, the strategy network is updated.

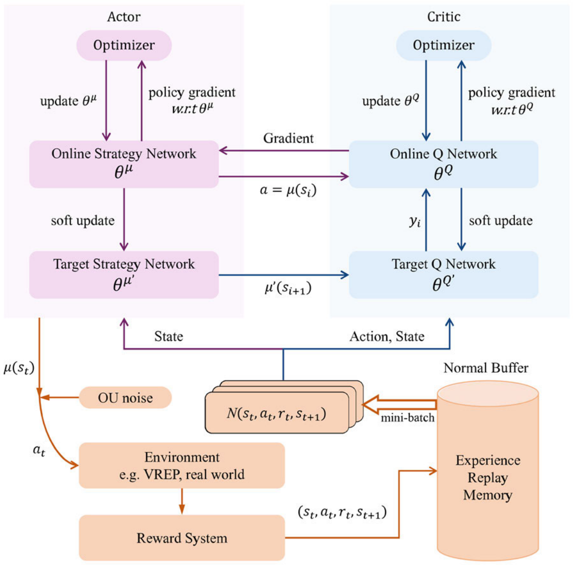

Actor
Optimizer
update θ^μ
policy gradient w.r.t θ^μ
Critic
Optimizer
update θ^Q
policy gradient w.r.t θ^Q
Gradient
Online Strategy Network θ^μ
$a = \mu(s_i)$
Online Q Network θ^Q
soft update
y_i
soft update
Target Strategy Network $\theta^{\mu'}$
$\mu'(s_{t+1})$
Target Q Network $\theta^{Q'}$
State
Action, State
Normal Buffer
$\mu(s_t)$
OU noise
$N(s_t, a_t, r_t, s_{t+1})$
mini-batch
a_t
Environment e.g. VREP, real world
Experience Replay Memory
(s_t, a_t, r_t, s_{t+1})
Reward System

FIGURE 1. DDPG algorithmic framework [18].

The Critic network is used to represent the distribution of values of the action space in the state space and its parameters are updated using the Q-learning algorithm. The loss function and parameter updating process for the actor network is as follows:

$$J(w) = minE_\pi\left[\frac{1}{2}(r + \gamma Q(s', a') - Q(s, a))^2\right] \tag{2}$$

$$\nabla_w J(w) = E_\pi\left[(r + \gamma Q(s', a') - Q(s, a))\nabla_w Q(s, a)\right] \tag{3}$$

$$w = w + \delta_w \nabla_w J(w) \tag{4}$$

The Actor network deterministically maps the current state to a specific operation that updates its parameters using a deterministic policy gradient algorithm. The process of updating the parameters of the Actor function is as follows.

$$J(\theta) = maxE_\pi[Q(s, \mu_\theta(s))] \tag{5}$$

$$\nabla_\theta J(\theta) = E_\pi\left[[\nabla_a Q(s, a)]_{a=\mu_\theta(s)} \nabla_\theta \mu_\theta(s)\right] \tag{6}$$

$$\theta = \theta + \delta_\theta \nabla_\theta J(\theta) \tag{7}$$

The target network is updated by replicating the original network, which contains the delay factor φ:

$$w' = \varphi w' + (1 - \varphi) w' \tag{8}$$

$$\theta' = \varphi\theta' + (1 - \varphi)\theta' \tag{9}$$

Since the actor function represents a deterministic strategy, the Agent will only take certain actions based on the current strategy when interacting with the environment, which will inevitably lead to insufficient exploration of the environment.

Often, one improves the exploratory power of an actor model by adding Ornstein-Uhlenbeck (OU) noise to the original action. OU noise can be used to simulate time-dependent noise data as follows:

$$dx_t = \theta^{ou}\left(\mu^{ou} - x_t\right)dt + \sigma_t^{ou} dW_t \tag{10}$$

where x_t is the noisy data to be generated, and μ^{ou} is the expected value of the random variable, and W_t is the random variable generated by the Wiener process, which can be replaced by a simple random function. θ^{ou} and μ^{ou} are the parameters in the stochastic process, given before the experiment. The pseudo-code of DDPG is shown in TABLE 1:

IV. ROBOTIC ARM TRAJECTORY PLANNING TASK DEFINITION

The goal of the trajectory planning task of the robotic arm is to obtain an optimal policy, denoted as π^*, by which this strategy can generate more appropriate path solutions in various tasks involving path planning (e.g., grasping an object up and placing it at a certain place). Such tasks have a Markovian nature and can be solved by DRL, and at the same time, this Markovian nature simplifies the modeling and solving of the problem, which enables DRL algorithms to efficiently learn and generate optimal trajectory planning policies. In practice, in order to use DRL algorithms for robot trajectory planning, it is first necessary to determine the input information required by the algorithms, i.e., state values and action values.

To ensure that the DRL algorithm is applicable to different robots and end-effector, the intelligences need to acquire the following key information when observing the environment: 1) the state of the robot's joint corners; 2) the state of the end-effector; and 3) the relative relationship between the end-effector TCPs and the target object, including information on the position and attitude. In this model, the state s_t is defined as:

$$s_t = [\boldsymbol{TCP}, \boldsymbol{A}, \boldsymbol{S}, Done] \tag{11}$$

where $TCP = [T_x, T_y, T_z]$ is the absolute position of the end-effector (global coordinate system). $\boldsymbol{A}$ is a vector consisting of each joint angle A_i of the robotic arm at the current moment t. $S = [\Delta x, \Delta y, \Delta z, \Delta\alpha, \Delta\beta, \Delta\gamma]$ represents the position and attitude offset between the end-effector and the target location. $Done$ is a Boolean-type value that takes the value of 1 if the end-effector has arrived at the target object, and 0 if it still has not.

TABLE 1. Implementation of DDPG algorithm.

```
Algorithm 1 Implementation of DDPG algorithm
1: Initialize Actor network parameters $\theta$, Critic network parameters $\phi$
2: Initialize target Actor network parameters $\theta'$, target Critic network parameters $\phi'$
3: Initialize experience replay buffer ReplayBuffer
4: for each episode do
5:     Reset environment state, get initial state $s_1$
6:     for each time step $t$ do
7:         Select action $a_t$ using the current policy network (Actor): $a_t = \mu(s_t|\theta) + \epsilon$
8:         Execute action $a_t$, observe environment feedback, get next state $s_{t+1}$ and reward $r_t$
9:         Store experience tuple $(s_t, a_t, r_t, s_{t+1})$ in experience replay buffer ReplayBuffer
10:        if number of experiences in ReplayBuffer reaches a predefined threshold then
11:            Sample a batch of experience samples $(s_i, a_i, r_i, s_{i+1})$ from ReplayBuffer
12:            Calculate target Q value: $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta')|\phi')$
13:            Update the current Critic network ($\phi$) by minimizing the mean squared loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\phi))^2$
14:            Update the Actor network ($\theta$) using policy gradient: $\nabla_\theta J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\phi)|_{s=s_i,a=\mu(s_i)} \nabla_\theta \mu(s|\theta)|_{s=s_i}$
15:            Update target network parameters: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ and $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
16:        end if
17:        Update current state to next state: $s_t \leftarrow s_{t+1}$
18:    end for
19: end for
```

In order to adapt to different robots and end-effector, the actions in the DRL algorithm a_t is defined as the amount of translation and rotation of the end-effector that is used to adjust the position and attitude of the robot to achieve a specific task or reach a goal. It is defined as:

$$a_t = [\delta x, \delta y, \delta z, \delta\alpha, \delta\beta, \delta\gamma] \tag{12}$$

where δx, δy, δz represent the distance moved from the center point of the tool in the Cartesian 3D coordinate system, and $\delta\alpha$, $\delta\beta$, $\delta\gamma$ denote the amount of change in the Tait-Bryan angle, respectively. The process of trajectory generation, although it is possible to derive actions from trajectory changes at the end of the robot, this approach is still accompanied by two main challenges. First, by means of Eq. (12) the guided actions induce a continuous evolution of the position and attitude of the TCP at the robot's end to form a smooth motion path. This strategy allows the intelligence to plan the task path independently, but relies on a complex inverse kinematics solution when transforming the Operation Space into the Joint Space. For robots with six degrees of freedom or less, mapping the planned trajectory to continuous joint angle variations is difficult; and when dealing with redundant robots, the inverse kinematics solution will become extremely cumbersome. Second, the approach of using direct rotation of the end TCP to plan the pose may suffer from Gimbal Lock phenomenon, which results in the loss of rotational flexibility in a certain direction.

In order to construct the planning of the robot's task trajectory based on positive kinematics, in this paper, the control is performed directly in the joint space, and the action is a_t defined as:

$$a_t = [\Delta a_1, \Delta a_2, \ldots, \Delta a_i] \tag{13}$$

where Δa_i represents the relative amount of change in the angle of each joint controller (e.g., servo motor). In the framework of this study, we control the amount of rotation of each joint, i.e., the increment of the joint angle $\Delta a_i\,(i = 1, 2, \ldots, n)$, to realize the robot's path planning. This approach allows the intelligence to fine-tune the robot's joint angles based on the current state and strategy, thus inducing small translations and rotations of the end TCP. Since the exact values of the individual joint angles are known, we can uniquely determine the position and attitude of the TCP through the positive kinematics of the robot.

V. DEEP DETERMINISTIC POLICY GRADIENT METHOD BASED ON HIERARCHICAL MEMORY STRUCTURE

A. FULL RESERVATION EXPERIENCE

In DRL, Experience Replay is a key mechanism, and there are two mainstream strategies, FIFO and Full Retention. The FIFO strategy stores experiences in a fixed capacity buffer and gradually replaces old experiences in the order of generation time. This approach focuses on frequently accessed states under the current policy. Comparatively, the Full Retention strategy stores all experiences generated from interactions with the environment, thus achieving comprehensive coverage of the entire state space and a high degree of experience diversity.

T De Bruin et al. proved that algorithms incorporating different ways of experience retention have significantly different returns in different tasks [23], i.e., experience retention has an important impact on algorithm performance. It also proved experimentally that the size of the memory is an important parameter of experience retention [23], and the state distribution of the memory tends to have different distributions depending on the size of the memory. Full retention can be regarded as a special case of FIFO, and the two are equivalent when the memory is large enough.

To further explore the performance differences between these two strategies, this study conducted an experimental comparison on the MountainCar-v0 and Acrobot-v1 tasks in the Gym environment. It is worth noting that MountainCar-v0 is a discrete state space task while Acrobot-v1 is a continuous state space task. The experimental results are shown in Fig. 2, where DQN+FIFO denotes deep Q-network combined with first-in-first-out experience replay, and DQN+FULL denotes deep Q-network combined with full retention experience replay. In the MountainCar-v0 environment, the FIFO strategy outperforms the full retention strategy. However, in the Acrobot-v1 environment with continuous state space, the full retention strategy performs even better. This further emphasizes the superiority of the all-retention strategy in continuous state space, especially in scenarios that require a high degree of experience diversity and comprehensive state space coverage.

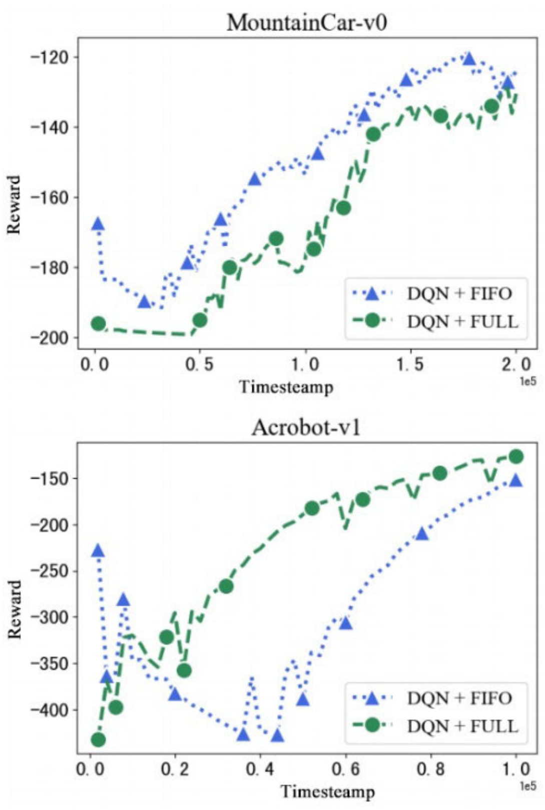


FIGURE 2. FIFO and full retention of experience put back on the classic control task returns.

This study employs a full retention strategy, a decision based on several key advantages. First, the full retention strategy provides a richer training dataset by providing more comprehensive coverage of the state space. Second, by retaining all experiences, the strategy is able to learn from diverse states and actions, thus enhancing the generalization ability of the model. Finally, by including all possible states and actions, the full-preservation strategy reduces the instability caused by ignoring certain rare but important states.

B. HIERARCHICAL MEMORY STRUCTURE

The offline learning goal of DRL algorithms is to eliminate the temporal correlation between sample data before and after network parameter updates [24]. To achieve this goal, many algorithms such as DQN, DDPG, TD3, SAC, etc. introduce an memory to store limited historical empirical data (s_t, a_t, r_t, s_{t+1}). These memories operate according to the first-in-first-out (FIFO) principle, where the earliest saved data are replaced by new empirical data when the data reaches the capacity limit. However, these algorithms usually use random uniform sampling, which ignores the importance of sample data and thus reduces the learning efficiency.

Prioritized Experience Replay (PER) attempts to address this problem by preferentially picking data with large TD differences for sampling, especially in the case of sparse rewards [25]. However, when the rewards are thicker, the effect of PER is limited and does not even significantly improve the performance [26]. In addition, the computation of TD difference involves the current reward and the target Q value, which increases the computational complexity.

The core goal of DRL is to maximize the cumulative reward. Thus, in the early stages of training, when Actor and Critic's network parameters are not perfect, highly rewarding sample data is indeed more helpful for intelligences to find higher cumulative rewards. This observation highlights the importance of sample data, especially when the memory contains a large amount of undesirable data. Highly rewarding sample data has the potential to drive intelligences to more successful experiences, thus providing a more efficient learning path.

In order to utilize the experience at different stages of the training process more rationally and to improve the convergence speed and performance of the driving strategy, this paper proposes a novel hierarchical memory structure, as shown in Fig. 3 shown. We design an experience separation mechanism that distinguishes between high reward-value experiences and low reward-value experiences and stores them in the Priority Experience Buffer (B_{Pri}) and the Secondary Experience Buffer (B_{Inf}), respectively, and both experience pools follow the all-retention principle.

This is crucial for convergence because a well-trained model is always updated by previous successes, which can lead to greater robustness and flexibility in most states encountered by the intelligence. In short, what starts out as an attempt to speed up the training process fails when the model tends to converge. Therefore, in this paper, we design a mini-batch sampling strategy that tends to be unbiased sampling, where each small batch of experience used for training is selected in different proportions from the pool of prioritized experience and the pool of secondary experience.

C. BIAS-FREE STRATEGY

The hierarchical memory structure addresses a key issue: how to effectively utilize high-reward data to facilitate the learning

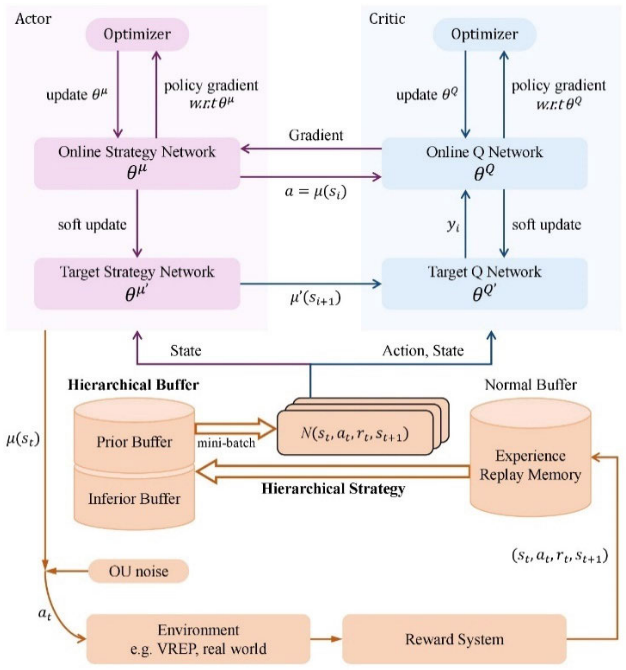


FIGURE 3. Network structure of hierarchical memory depth deterministic policy gradient algorithm.

of intelligences without falling into overfitting. Although the sampling of high-reward data is beneficial to enhance the learning speed, sampling high-reward data throughout will lead to model overfitting [27]. Therefore, this paper also designs a Bias-Free strategy to balance the sampling ratio of high and low reward data, so that the algorithm gradually shifts from biased sampling at the beginning of training to unbiased sampling.

At the beginning of training, the Bias-Free strategy tends to sample high-reward data from the prioritized memory to quickly improve the smart body strategy. As training advances, the strategy gradually increases the proportion of sampling low-reward data from the secondary memory to avoid overfitting and ensure global optimization. Assume that the total capacity of the full memory is N_{total}, where the number of high-reward data is N_{Pri} and the number of low reward data is N_{Inf} which satisfies $N_{total} = N_{Pri} + N_{Inf}$.

In general, when training DRL models, a small batch of data is usually randomly drawn from the memory with a total capacity of N. While the Bias-Free strategy draws from the prioritized memory and the secondary memory with capacity of $N - N_1$ and N_1 of the experience memories as the sampled data for extraction. Where N_1 the value of the Bias-Free policy grows logarithmically from 0 until it reaches the upper limit N_{1max} this approach leads the algorithm to perform biased sampling while learning, which in turn leads to the overfitting problem, so we restrict Eq. (14) to the condition that the algorithm gradually transitions from biased sampling to unbiased sampling:

$$\frac{N_{1max}}{N} = \frac{N_{Inf}}{N_{total}} \tag{14}$$

This Bias-Free strategy works through biased sampling in the early and middle stages of algorithm training to rapidly improve the performance of the intelligences with the help of highly rewarded data and increase the experience of success. As training progresses, the strategy gradually shifts to unbiased sampling, ensuring that the algorithm stabilizes to reach the global optimal solution.

The hierarchical experience pool library structure with the pseudocode for the Bias-Free sampling strategy is shown in TABLE 2. In this paper, we incorporate this approach into the DDPG algorithm to form an innovative hierarchical empirical pooling Deep Deterministic Policy Gradient (HM-DDPG) algorithm and demonstrate its potential and efficiency in practical applications.

TABLE 2. Implementation of DDPG algorithm.

```
Algorithm 2 Implementation of the Hierarchical Memory with Bias-Free strategy
 1: Initialize: $N_{Pri} = 0$, $N_{Inf} = 0$, $N = 128$, $N_1 = 0$
 2: for each episode do
 3:     $N_{total} = N_{Pri} + N_{Inf}$
 4:     $N_{1max} = \frac{N_{Inf}}{N_{total}} \times N$
 5:     if $N_1 < N_{1max}$ then
 6:         $N_1 \leftarrow \min(N_{1max}, N_{1max} \log_{\beta} N_1)$
 7:     end if
 8:     Sample $N_1$ and $N - N_1$ data from $B_{Inf}$ and $B_{Pri}$
 9:     TrainModel(samples)
10:     Update $N_{Pri}$ and $N_{Inf}$ based on new experiences
11: end for
```

VI. EXPERIMENTATION

A. DESCRIPTION OF THE EXPERIMENTAL ENVIRONMENT

In this study, a set of highly accurate virtual simulation platform is constructed on VREP platform, such as Fig. 4 shown, the platform comprehensively demonstrates the configuration of the simulation environment from multiple perspectives. The right side of the robotic arm is equipped with a tabletop on which two obstacles and a manipulable yarn cylinder model are arranged. Correspondingly, on the right side, there is an open working platform dedicated to precisely position the yarn bobbin to the intended target. A beam hangs at the rear of the environment, acting as an aerial support structure whose presence clearly defines the no-go area for collisions. Additionally, there is a humanoid model scattered throughout the environment as a potential obstacle during mission execution.

The simulation platform enables us to conduct large-scale policy training, thus obtaining a rich and high-quality training dataset that can be directly used for trajectory planning and generation of real robots. More importantly, the simulation system not only facilitates policy migration and safety constraints, but also takes into account a wide range of production

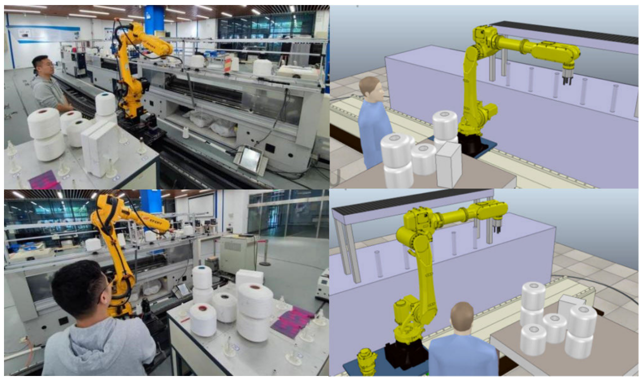


FIGURE 4. Composition of virtual simulation platform.

scenarios and environmental variables in the simulation. This feature allows us to provide more comprehensive and accurate training data for the actual operating robots, and to verify the robustness and reliability of the robots under a variety of environmental conditions. The system also allows us to pre-test and optimize safety and stability in a safe and controlled virtual environment.

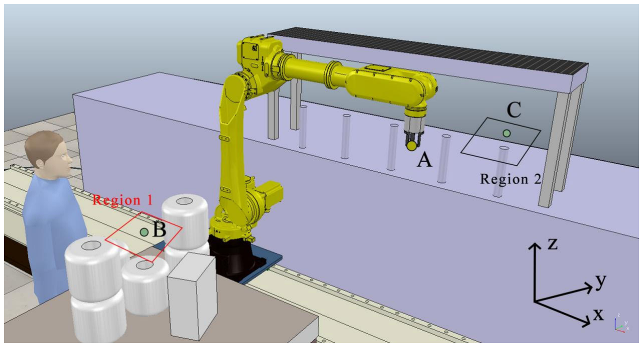


FIGURE 5. Continuous trajectory planning task for a textile robot arm.

B. MISSION STATEMENT

Fig. 5. depicts the continuous trajectory planning task of a textile robotic arm in a virtual simulation platform. At the start of the task, the robotic arm is located at and near the initial position point A. A number of obstacles of different sizes and dimensions are placed on the tabletop to the right of the robotic arm, details of which will be further elaborated in subsequent sections. In addition, there is a yarn bobbin on the tabletop as a workpiece to be gripped and placed, which is located over point B.

The process of gripping and placing the bobbin involves a series of precise movements. First, the robotic arm actuator needs to face down vertically, insert the claw into the cavity in the center of the yarn bobbin and hold it open, then lift it with the help of friction, and finally place it in a suitable posture in the region of point C on the left platform. In order to increase the generalization of the problem, only one target yarn cylinder (directly below area 1) is placed in the virtual simulation platform, but it can be placed in a different location before each simulation to accommodate more possibilities.

In the process of moving from point A to point B, the end-effector of the robotic arm needs to move a short distance in the direction of the -x-axis, then move along the -y-axis, then continue to move in the direction of the -x-axis and approach the bobbin with a suitable attitude, and finally lift the bobbin along the z-axis. During this process, if the robotic arm moves directly from point A to point B, it will inevitably collide with the beams in the environment.

Thus, the entire grasp-place task is carefully divided into complete trajectories consisting of the following consecutive trajectory segments:

(1) The yarn bobbin appears at an arbitrary position on the table, and the end of the robotic arm TCP moves from the initial position A along the trajectory segment 1 with a suitable attitude to the preparatory position B. The point B is located above the center point of the yarn bobbin along the direction of the z-axis, and it may appear at an arbitrary position in Region1.

(2) The robotic arm moves along the trajectory segment 2, from the preparation position B to the placement position C (which can be randomly specified in Region2) in a suitable attitude, contracts the hand claw and places the yarn bobbin on the platform.

(3) Reset and move from the placement position C to the vicinity of the initial position A.

There is a temporal sequence of these three steps, which together constitute the entire process of grasping-placing the yarn cylinder, demonstrating the complexity and variety of the task.

C. EXPERIMENTAL RESULTS AND ANALYSIS

All experiments were performed on a computer equipped with AMD Ryzen7 3700X 8-Core Processor @ 3.60GHz, NVIDIA GeForce RTX3080 Ti. The code compilation environment was PyTorch 2.0.0 + cuda11.8 framework with Python 3.8. the virtual simulation platform used was V-REP PRO EDU Version 3.6.2. the hyperparameters of the HM-DDPG training process were as follows Table 3 shown.

TABLE 3. Experimental parameter settings.

Parameters	Value
Motion space actions	6
Training rounds episodes	1000
Maximum steps	1000
Learning rate	0.001
Discount factor γ	0.99
Exploring factor ε	0.9
Soft update factor τ	0.005
Batch size	64
Explore Noise	OU noise (μ^{ou} is 0, andθ^{ou} 0.2)

In order to verify the effectiveness of the proposed algorithm in this paper, a robot grasping and placing task is

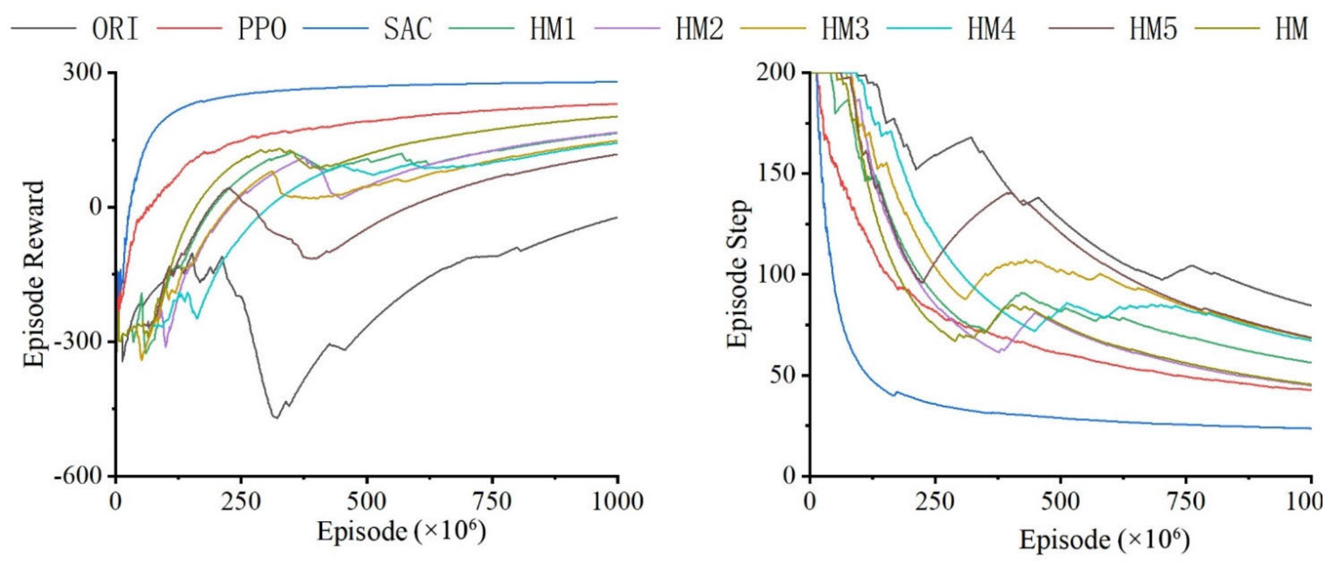


FIGURE 6. Comparison of training effect of each algorithm.

TABLE 4. Experimental parameter settings.

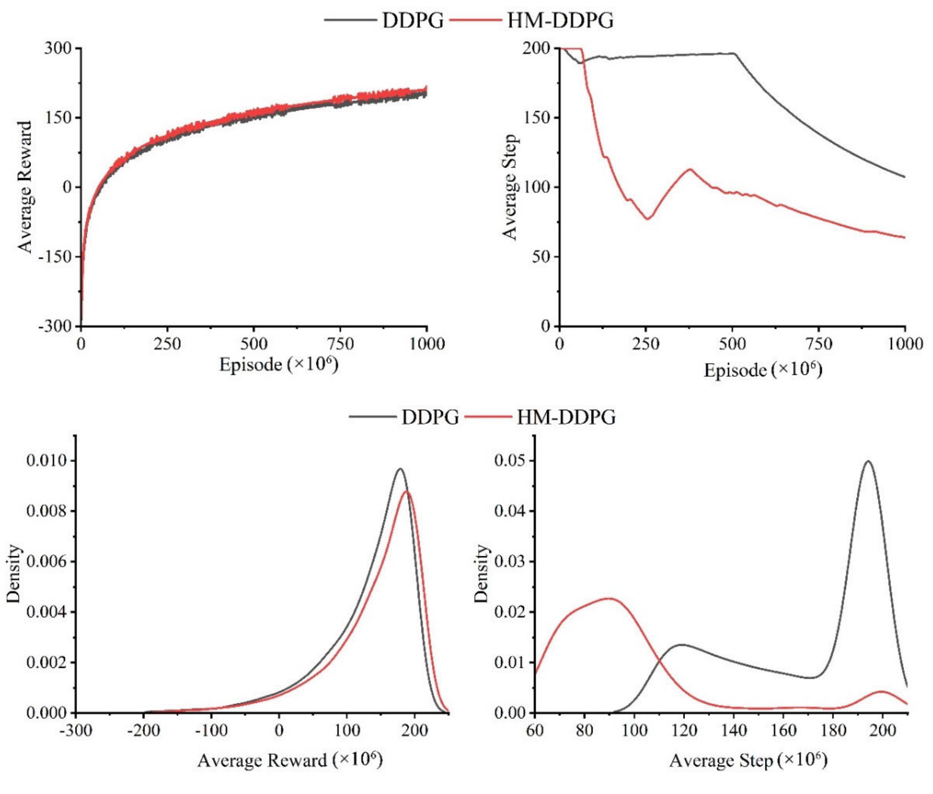
Method	Training Set				Test Set
	1-250	251-500	501-750	751-1000	1-500
ORI	22.80%	57.60%	80.40%	92.80%	86.60%
PPO	70.80%	88.40%	94.00%	95.60%	94.80%
SAC	91.60%	100.00%	100.00%	100.00%	100.00%
HM1	52.00%	70.40%	85.60%	96.20%	92.40%
HM2	51.20%	73.20%	98.80%	97.20%	92.00%
HM3	52.40%	75.60%	82.80%	97.60%	90.20%
HM4	43.60%	82.00%	83.60%	95.60%	89.60%
HM5	40.00%	64.80%	69.60%	76.80%	68.20%
HM	66.80%	77.20%	98.00%	99.30%	99.00%

designed in this paper, which consists of the following steps: (1) moving from the initial position A to the preparatory position B; (2) moving from the preparatory position B to the placing position C; and (3) moving from the placing position C back to the initial position A. The experimental environment consists of the elements of obstacles, workpieces, and robots. At the beginning of each round, the robotic arm stays at the initial position and its joint angle is set as [0,0,0,-90°,0,90°,0]. In order to make the continuous movements generated by the Actor stable enough, its output movement value is limited to [-1.5°,1.5°] and the size of the OU noise is limited to [-0.5,0.5] within the size of the OU noise.

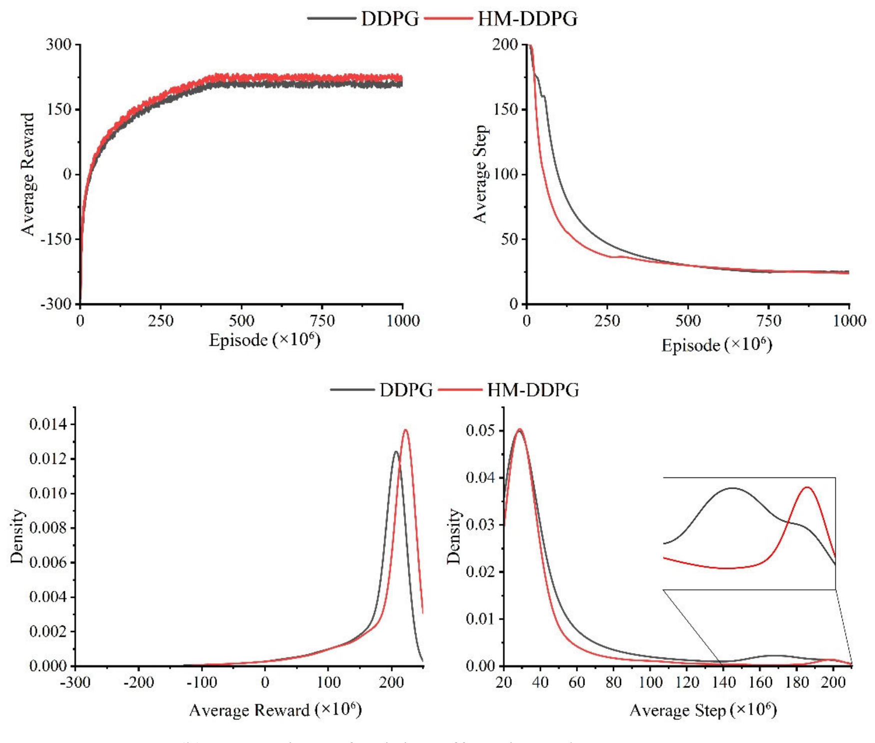
In this paper, average round steps, round rewards and success rate are used as evaluation metrics to measure the performance of different algorithms in accomplishing the task [28].

In the experimental part, we used various algorithms as comparison benchmarks. Among them, ORI stands for the benchmark DDPG algorithm [18]; PPO refers to the Proximal Policy Optimization (PPO) algorithm [16]; and SAC is the Soft Actor-Critic (SAC) algorithm [21]. In addition, HM denotes the DDPG algorithm with a hierarchical Memory structure. To further explore the effect of the proportion of sampling from high reward intervals on the performance of the algorithm, we also set up five variants, HM1 to HM5. These variants are all based on the HM-DDPG algorithm, but they fix the proportion of data sampled from the high reward intervals during the training process. Specifically, these proportions were set to 94%, 80%, 60%, 50% and 20%, respectively. Fig. 6. shows the Average Episode Reward and Average Episode Step for each algorithm during training. Together, these two metrics provide a comprehensive view to evaluate the performance of different algorithms in terms of task execution efficiency and learning stability.

Table 4 demonstrates the success rate of each algorithm at different training stages, which is defined as the ratio of the number of successful rounds to the total number of rounds



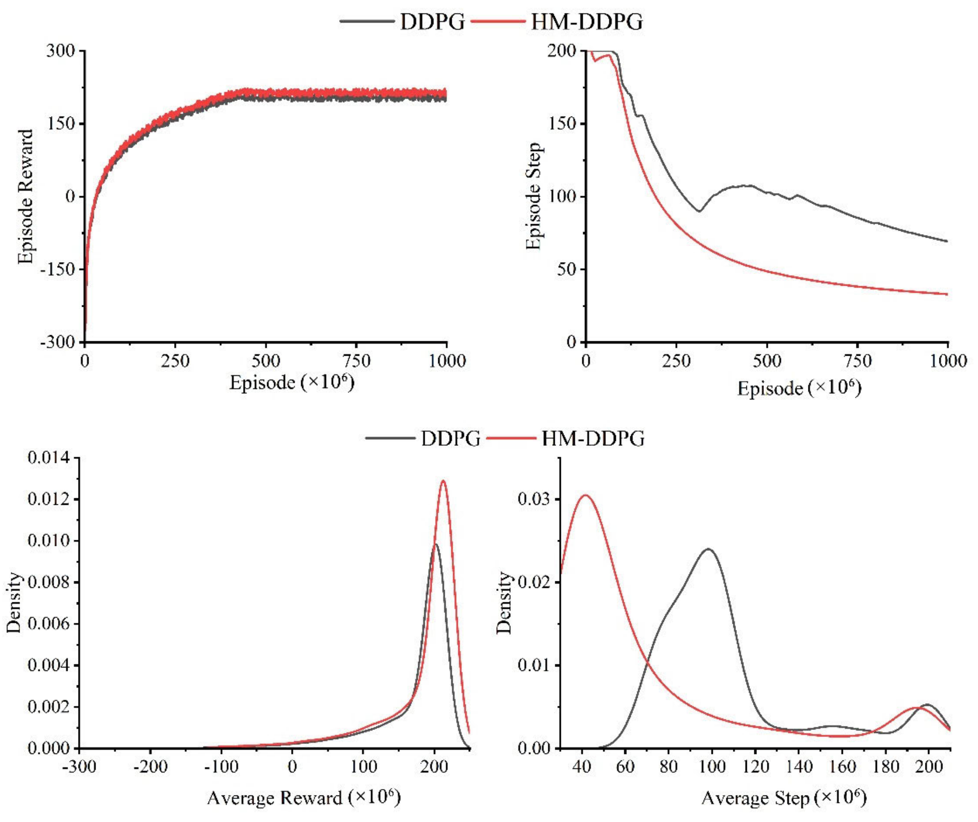
(a) Comparison of training effects in track segment 1



(b) Comparison of training effects in track segment 2

FIGURE 7. Performance comparison between DDPG and HM-DDPG for planning different trajectories.

in a given time interval. It can be observed that HM-DDPG and its variants (HM1 to HM4) significantly outperform the benchmark algorithm ORI in terms of success rate in the early stages of training (rounds 1-500). This phenomenon



(c) Comparison of training effects in track segment 3

FIGURE 7. *(Continued.)* **Performance comparison between DDPG and HM-DDPG for planning different trajectories.**

TABLE 5. **Comparison of the performance of DDPG and HM-DDPG under different noise disturbances.**

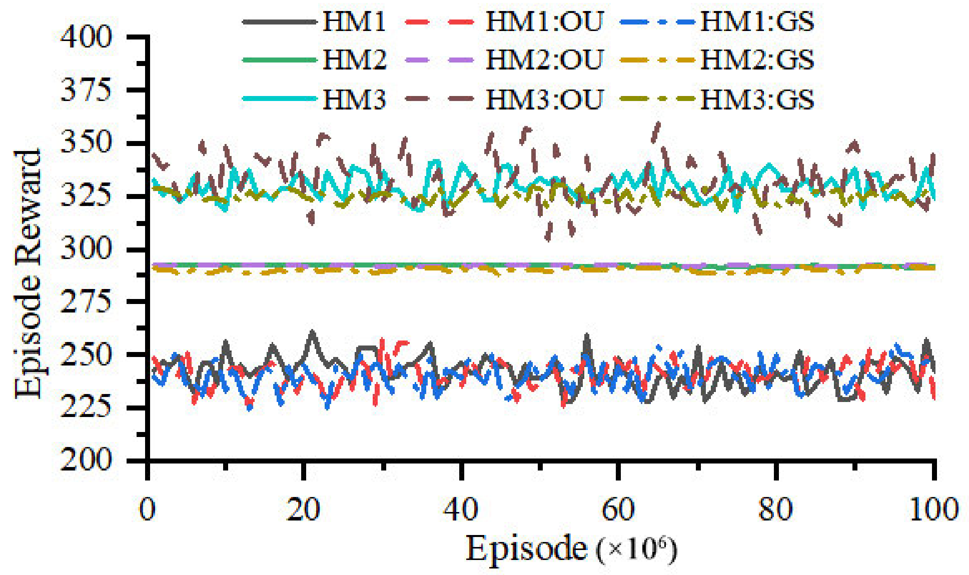
	Di/HMi		*Di*:*OU*/*HMi*:OU		*Di*:*GS*/*HMi*:GS	
	Average episode steps ($\times 10^6$)	SD	Average episode steps ($\times 10^6$)	SD	Average episode steps ($\times 10^6$)	SD
$i = 1$	13.4/12.5	17.4/12.3	13.8/12.9	7.4/6.5	14.6/12.9	43.2/12.0
$i = 2$	14.1/13.4	8.2/7.5	14.2/13.6	1.8/1.7	15.0/13.9	13.5/12.2
$i = 3$	29.4/24.8	31.1/30.2	29.5/25.8	27.6/24.3	30.6/27.6	50.4/47.3

corroborates the role of highly rewarding experiences (stored in the B_{Pri} in) facilitates in the early learning of the algorithm.

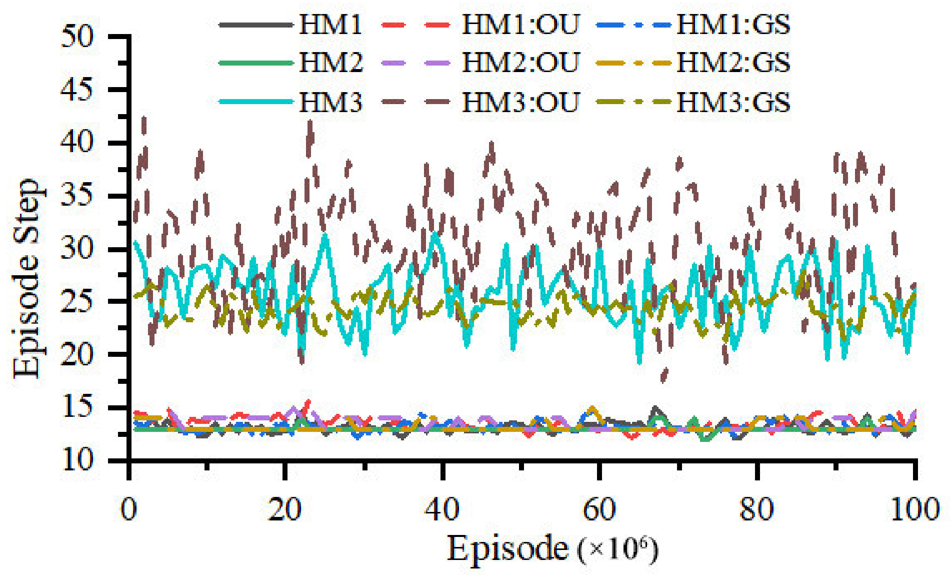
However, continued reliance on data sampling with high reward intervals, i.e., local data-driven training, may lead to model overfitting or convergence of locally optimal solutions [29]. As a result, although HM1 to HM4 have significantly higher success rates on the training set than ORI, they fail to outperform ORI on the test set.

In contrast, the HM algorithm maintains a higher success rate and achieves a faster convergence rate by implementing a stratified sampling strategy in both the training and testing phases. More importantly, after convergence, the HM algorithm demonstrated a more stable and smoother trend of round reward changes, while the other algorithms still exhibited significant performance fluctuations. This observation further confirms the effectiveness of the stratified sampling strategy in promoting the global optimization and stability of the algorithm.

In the comparison experiments of three mainstream DRL algorithms, PPO, SAC, and DDPG, the experimental results show that PPO and SAC outperform DDPG in terms of convergence speed and final performance. However, it is important to note that the model parameters of PPO and SAC are relatively large, which leads to an increase in



(a) Variation of HM-DDPG reward function fluctuation under different noise disturbances



(b) Variation of fluctuation of HM-DDPG round steps under different noise disturbances

FIGURE 8. Anti-noise interference performance of HM-DDPG.

computational and storage overheads [30]. Moreover, due to its deterministic policy structure, DDPG has lower overhead in terms of model storage and deployment. Therefore, DDPG is a more worthwhile choice in resource-constrained scenarios.

As Fig. 7 demonstrates, HM-DDPG performs well in the six-degree-of-freedom robotic arm continuous three-segment trajectory planning task described in Section VI-B. Clearly, HM-DDPG achieves faster convergence, lower training steps, and higher reward gains in each subtask. The algorithm demonstrates superior efficiency and stability in the training phase, as well as a more refined action strategy. The changes in its round rewards are also relatively smooth with limited fluctuations. Notably, among the three trajectory segments, Trajectory 1 and Trajectory 3 showed significant fluctuations in their round steps due to higher task complexity. Further observation reveals that compared to the baseline DDPG algorithm, HM-DDPG reaches convergence faster in the training of each trajectory segment, thus realizing significant savings in the total training time of the three consecutive trajectory segments.

Finally, in order to demonstrate the robustness of the HM-DDPG proposed in this paper under multiple noise disturbances, it is compared with the DDPG and each algorithm is tested for 100 rounds under three different trajectories of the task: (1) all the algorithms are tested under the original OU noise, where the OU noise is within [−0.5,0.5]; (2) the original OU noise is replaced with another larger OU noise with amplitude within [−1,1] (algorithms named *Di*:OU and *HMi*:OU); (3) OU noise was replaced with Gaussian noise with variance 0.5 (algorithms named *Di*:GS and *HMi*:GS).

As Fig. 8 is the same as the Table 5 clearly demonstrate that the *Di*:GS and *HMi*:GS algorithms are consistent with the original model in terms of the average number of steps in each step of the noise experiment. Notably, *Di*:OU and *HMi*:OU require more time steps to complete the task compared to the other algorithms and exhibit higher standard deviation (SD), which reflects their larger fluctuations in the average number of round steps. Further observations show that the HM-DDPG algorithm is overall more resistant and robust to interference than DDPG. This result supports the ability of the HM-DDPG algorithm proposed in this study to maintain a high degree of stability in the face of a variety of noise conditions, thus adapting to the demands of multi-stage assembly tasks.

VII. CONCLUSION

In this research, we developed a novel Hierarchical Memory-based Deep Deterministic Policy Gradient (HM-DDPG) algorithm for complex scenarios in the textile industry, incorporating positive kinematics for effective multi-segment robotic arm trajectory planning. This algorithm innovatively adopts a hierarchical memory structure to classify experience data based on reward values, enhancing learning efficiency and reducing overfitting. The Bias-Free strategy ensures balanced sampling of diverse reward data during training, promoting comprehensive strategy exploration. Applied to a challenging robotic arm grasp-place task, HM-DDPG notably enhances task execution accuracy and efficiency. Rigorous testing in a simulated environment demonstrated the algorithm's superiority in complex tasks, validating our approach. The use of positive kinematics bypasses the intricacies of inverse kinematics, ensuring mechanical feasibility of trajectories and addressing the gimbal locking issue. Future research may extend to multi-agent communication and collaboration, furthering the algorithm's applicability.

REFERENCES

[1] P. Morasso and F. A. M. Ivaldi, "Trajectory formation and handwriting: A computational model," *Biol. Cybern.*, vol. 45, no. 2, pp. 131–142, Sep. 1982.

[2] C.-S. Lin, P.-R. Chang, and J. S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for mechanical manipulators," in *Proc. 21st IEEE Conf. Decis. Control*, Dec. 1982, pp. 330–335.

[3] S. E. Thompson and R. V. Patel, "Formulation of joint trajectories for industrial robots using B-splines," *IEEE Trans. Ind. Electron.*, vol. IE-34, no. 2, pp. 192–199, May 1987.

[4] Y. Li, T. Huang, and D. G. Chetwynd, "An approach for smooth trajectory planning of high-speed pick-and-place parallel robots using quintic B-splines," *Mechanism Mach. Theory*, vol. 126, pp. 479–490, Aug. 2018.

[5] A. Gasparetto and V. Zanotto, "A new method for smooth trajectory planning of robot manipulators," *Mechanism Mach. Theory*, vol. 42, no. 4, pp. 455–471, Apr. 2007.

[6] G. Oriolo, M. Cefalo, and M. Vendittelli, "Repeatable motion planning for redundant robots over cyclic tasks," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1170–1183, Oct. 2017.

[7] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Proc. 7th IEEE-RAS Int. Conf. Humanoid Robots*, Nov. 2007, pp. 477–482.

[8] S. Feraco, S. Luciani, A. Bonfitto, N. Amati, and A. Tonoli, "A local trajectory planning and control method for autonomous vehicles based on the RRT algorithm," in *Proc. AEIT Int. Conf. Electr. Electron. Technol. Automot. (AEIT AUTOMOTIVE)*, Nov. 2020, pp. 1–6.

[9] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2006, pp. 1874–1879.

[10] Z. Wang, J. Chang, B. Li, C. Wang, and C. Liu, "Application of improved rapidly-exploring random trees (RRT) algorithm for obstacle avoidance of snake-like manipulator," in *Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA)*, Oct. 2020, pp. 490–495.

[11] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. Conf. AAAI Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.

[12] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," 2018, *arXiv:1803.07055*.

[13] O. Anschel, N. Baram, and N. Shimkin, "Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning," 2016, *arXiv:1611.01929*.

[14] A. R. Mahmood, D. Korenkevych, and G. Vasan, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proc. Conf. Robot Learn.*, 2018, pp. 561–591.

[15] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015, *arXiv:1502.05477*.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[17] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," 2017, *arXiv:1702.08165*.

[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[19] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, and M. Gazzola, "Elastica: A compliant mechanics environment for soft robotic control," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3389–3396, Apr. 2021.

[20] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv:1802.09477*.

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018, *arXiv:1801.01290*.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[23] T. De Bruin, J. Kober, K. Tuyls, and R. Babuska, "Experience selection in deep reinforcement learning for control," *J. Mach. Learn. Res.*, vol. 19, no. 9, pp. 1–56, 2018.

[24] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," 2017, *arXiv:1705.02755*.

[25] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2469–2478.

[26] J. Eschmann, "Reward function design in reinforcement learning," in *Reinforcement Learning Algorithms: Analysis and Applications*. Cham, Switzerland: Springer, 2021, pp. 25–33.

[27] S. Carta, A. Ferreira, A. S. Podda, D. R. Recupero, and A. Sanna, "Multi-DQN: An ensemble of deep Q-learning agents for stock market forecasting," *Exp. Syst. Appl.*, vol. 164, Feb. 2021, Art. no. 113820.

[28] Y. Ma, D. Xu, and F. Qin, "Efficient precision insertion strategy based on demonstration learning and reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4492–4502, 2021.

[29] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 819–825.

[30] Q. He, H. Su, C. Gong, and X. Hou, "MEPG: A minimalist ensemble policy gradient framework for deep reinforcement learning," 2021, *arXiv:2109.10552*.

[31] B. Zheng, G. Jiang, Z. Dong, and H. Liu, "Design of warp knitting electronic shogging system based on mixed-velocity planning curve," *Textile Res. J.*, vol. 91, nos. 13–14, pp. 1594–1608, Jul. 2021.

DI ZHAO (Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Tiangong University, Tianjin, China, in 2010 and 2015, respectively, where he is currently pursuing the Ph.D. degree in mechanical engineering.

As a Senior Visiting Scholar with Peking University, Beijing, he is studying mechanical engineering and mechanics and electromagnetism related topics. Since 2015, has been the Head of the Electromechanical Innovation Laboratory and an Assistant Professor, engaged in theoretical research and technical application in the fields of machinery, electronics, and electromagnetism. He has participated in three projects of the National Natural Science Foundation of China. He is the author of two books, more than 16 articles, and more than 21 inventions. His research fields include electromechanical control theory, wireless charging related mechanism and testing, target recognition, and artificial intelligence algorithms.

Mr. Zhao received the Scientific and Technological Progress Award (Tianjin Science and Technology Bureau), the Second Prize for Innovation and Invention (Ministry of Education of China), and the Title of Excellent Innovative Method Teacher (Ministry of Science and Technology of China).

ZHENYU DING was born in Ulanhot, Inner Mongolia, China, in 2002. He is currently pursuing the bachelor's degree in electronic information engineering with Tiangong University, Tianjin.

His research interest is focused on the deep reinforcement learning and engineering applications. His inaugural academic paper was showcased at the 12th IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems.

WENJIE LI was born in Luliang, Shanxi, China, in 1998. He received the B.E. degree in mechanical design, manufacturing, and automation from Tiangong University, in 2021, where he is currently pursuing the master's degree in mechanical engineering.

He is also the Director of the Tianjin Key Laboratory, from 2021 to 2024. The first academic article was published in 2022 and the second article on the application of genetic algorithms in the engineering department was submitted in 2023. His research areas include algorithm research in engineering applications, such as path planning and algorithm optimization for six degree of freedom robotic arm.

Mr. Li won the Bronze Award of China International Internet Plus, in 2023; the Gold Award of Undergraduate Entrepreneurship and Innovation Competition, in 2022; and the Second Prize of School Level Scholarship, from 2021 to 2022.

SEN ZHAO was born in Hanzhong, Shaanxi, China, in 1999. He received the B.E. degree in mechanical design, manufacturing, and automation from the Mingde College, Northwestern Polytechnical University, in 2021. He is currently pursuing the master's degree in mechanical engineering with Tiangong University. His research interests include predominantly in path planning algorithms for robotic arm and collision detection for obstacle avoidance planning.

YUHONG DU was born in Shanghai, China, in 1975. She received the B.S. and M.S. degrees in mechanical engineering from Zhejiang University, Hangzhou, China, in 1998 and 2001, respectively, and the Ph.D. degree in mechanical engineering from Tiangong University, Tianjin, China, in 2011.

She has published over 100 peer-reviewed academic papers and holds a patent in the field of mobile platform control. Her research primarily focuses on control systems for mobile platforms, pattern recognition algorithms, and applied image processing technologies.

Dr. Du has garnered numerous accolades over the course of her career, signaling her eminence in the field of mechanical engineering and artificial intelligence. She was a recipient of the American Society of Mechanical Engineers (ASME) Outstanding Early Career Award, in 2013, highlighting her groundbreaking work in mobile platform control systems. In 2018, she received the Institute of Electrical and Electronics Engineers (IEEE) Computational Intelligence Society's Best Journal Paper Award, for her seminal paper on reinforcement learning applications in robotic systems. Furthermore, her entrepreneurial acumen led her to be honored with the National Innovation and Entrepreneurship Award, in 2021, recognizing her successful transition of academic research into commercial products.