Deep Reinforcement Learning With a Stage Incentive Mechanism of Dense Reward for Robotic Trajectory Planning

Gang Peng, *Member, IEEE*, Jin Yang, *Student Member, IEEE*, Xinde Li, *Senior Member, IEEE*, and Mohammad Omar Khyam, *Member, IEEE*

***Abstract*—To improve the efficiency of deep reinforcement learning (DRL)-based methods for robot manipulator trajectory planning in random working environments, we present three dense reward functions. These rewards differ from the traditional sparse reward. First, a posture reward function is proposed to speed up the learning process with a more reasonable trajectory by modeling the distance and direction constraints, which can reduce the blindness of exploration. Second, a stride reward function is proposed to improve the stability of the learning process by modeling the distance and movement distance of joint constraints. Finally, in order to further improve learning efficiency, we are inspired by the cognitive process of human behavior and propose a stage incentive mechanism, including a hard-stage incentive reward function and a soft-stage incentive reward function. Extensive experiments show that the soft-stage incentive reward function is able to improve the convergence rate, get higher mean reward and lower standard deviation after convergence.**

***Index Terms*—Deep reinforcement learning (DRL), dense reward function, stage incentive mechanism, trajectory planning.**

I. Introduction

IN ORDER to use a robot manipulator to complete a task, it is essential to realize trajectory planning of the robot manipulator. The results of trajectory planning directly determine the quality index of the task carried out by the robot manipulator [1], [2]. Traditional trajectory planning for robot manipulators mainly includes artificial potential field methods [3], [4], segmented geometry method [5], and polynomial interpolation [6], [7]. These methods have low intelligence, poor dynamic planning, and no self-learning ability. In recent years, deep reinforcement learning (DRL) has been applied to trajectory planning of robot manipulators [8], [9], [10], [11]. This method can allow a robot manipulator to learn autonomously and plan an optimal path in a complex and random environment.

He et al. [8] developed a reinforcement learning control strategy that is based on actor–critic structure to enable vibration suppression while retaining trajectory tracking for the manipulator. Katyal et al. [9] used only the original image of the working environment as the state space, and demonstrated robust and direct mapping from the image to the action domain by exploiting simulation for learning based on DRL. The method is robust to environmental changes. It can learn a manipulation policy, which authors show takes the first steps toward generalizing to changes in the environment and can scale to new manipulators. But because the state space is an image, dimensionality disaster is likely to occur. Kamali et al. [10] proposed a Dynamic-goal DRL method to address the problem of robot arm motion planning in telemanipulation applications. This method intuitively maps human hand motions to a robot arm in real time while avoiding collisions, joint limits, and singularities. Wen et al. [11] proposed to use a DRL method to plan the trajectory of a robot arm to realize obstacle avoidance. In their approach, the rewards are designed to overcome the difficulty in the convergence of multiple rewards, especially when rewards are antagonistic to each other. Sangiovanni et al. [12] proposed a hybrid control methodology to achieve full body collision avoidance in robot manipulators, which improves classical motion planning algorithms by using a DRL approach for performing obstacle avoidance, while achieving a reaching task in the operative space. James and Davison [13] innovatively presented an attention-driven robotic manipulation reinforcement learning algorithm, which can be applied to a range of sparse-rewarded tasks, given only a small number of demonstrations. Kumra et al. [14] defined a task progress-based Gaussian (TPG) reward function that computes the reward based on actions that lead to successful motion primitives and progress toward the overall

Manuscript received 27 January 2022; revised 21 July 2022 and 13 November 2022; accepted 6 December 2022. Date of publication 30 December 2022; date of current version 18 May 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 91748106, and in part by the Hubei Province Natural Science Foundation of China under Grant 2019CFB526. This article was recommended by Associate Editor W. He. *(Jin Yang and Gang Peng are co-first authors.) (Corresponding author: Jin Yang.)*

Gang Peng is with the School of Artificial Intelligence and Automation and the Key Laboratory of Image Processing and Intelligent Control, Huazhong University of Science and Technology, Wuhan 430070, China (e-mail: penggang@hust.edu.cn).

Jin Yang was with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430070, China. He is now with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xian 710300, China (e-mail: m201972630@hust.edu.cn).

Xinde Li is with the School of Automation, Southeast University, Nanjing 210000, China (e-mail: xindeli@seu.edu.cn).

Mohammad Omar Khyam is with the School of Engineering and Technology, Central Queensland University, Melbourne, VIC 3000, Australia (e-mail: m.khyam@cqu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2022.3228901.

Digital Object Identifier 10.1109/TSMC.2022.3228901

Fig. 1. DRL framework.

task goal, which can speed up the learning process of the agent.

As shown in Fig. 1, the three components of DRL are environment, agent, and reward function. The agent in DRL performs exploration to identify the possible actions. The robot manipulator executes the action in the environment and feeds back the reward value to the agent according to the defined reward function. Through the iterative update method, the agent learns better strategies of trajectory planning.

In the history of the development of DRL, a typical method is the deep Q-learning network (DQN) [15], [16]. However, its spaces of output action are discrete, so it is difficult to apply to continuous action spaces such as the trajectory planning of robot manipulators. Subsequently, deep deterministic policy gradient (DDPG) [17] based on the actor–critic (AC) architecture, asynchronous advantage AC (A3C) [18], proximal policy optimization (PPO) [19], and soft AC (SAC) [20] were proposed one by one. Otherwise, multiagent DRL has attracted a lot of attentions of scholars in recent years [21], [22], which makes the model in line with the actual situation. But there are challenges to be addressed. For example, goal consistency between agents, model extensibility, and environmental instability.

However, there are still problems in DRL methods, such as randomness and blindness. The key to these problems is the reward function, which is an important part of DRL, but it can lead to a large amount of useless exploration and, thus, decrease the efficiency of the algorithm [23], [24]. To solve the problem, we present a stage incentive mechanism based on human behavior cognition for robot trajectory planning in DRL. The primary contributions of this article are summarized as follows.

1) Combining the characteristics of trajectory planning and work environment, three brand-new dense reward functions are proposed. Dense reward functions provide nonzero rewards, and they differ from the sparse reward function in that they provide more information after each action, which can reduce invalid and blind exploration of DRL during trajectory planning for the robot manipulator.
2) A posture reward function and a stride reward function are proposed. The posture reward function includes a position reward function and a direction reward function: the position reward function is composed of the task status item (whether or not the task is completed) and the distance guide item (the Euclidean distance between the end of the robot manipulator and the random target), and the direction reward function is modeled by the angle between the expected direction vector and the actual direction vector. The stride reward function includes a position reward and a movement distance reward. The position reward is the same as that mentioned above, and the movement distance reward is composed of the average movement distance of each joint of the robot manipulator. Together, the posture reward function and the stride reward function can make the robot manipulator explore more efficiently under reasonable constraints in position, direction, and movement distance, and reduce invalid and blind exploration.
3) In order to further improve learning efficiency, we are inspired by the cognitive process of human behavior and propose a stage incentive mechanism. The hard-stage incentive mechanism is established by combining the posture reward function and stride reward function. To improve its potential stability hazards, a soft-stage incentive mechanism is further proposed. With this innovative structure, we have increased the expected return obtained by the algorithm while ensuring the stability of the algorithm, which has improved the overall efficiency of the algorithm.

The remainder of this article is organized as follows. The structures of the posture reward function and stride reward function are presented in Sections II and III. In Section IV, the stage incentive mechanisms are introduced, including the hard-stage incentive reward function and the soft-stage incentive reward function. The implementation of the reward functions is illustrated in Section V, mainly how to implement the proposed reward functions in the current mainstream DRL methods. Then, experimental results are demonstrated and discussed in Section VI. Finally, the conclusions are drawn in Section VII.

II. Posture Reward Function

For DRL-based methods, the robot manipulator performs a great deal of ineffective exploration in a complex random environment, which is the main reason for reducing the efficiency of the algorithm. A posture reward function restricts the relative position and relative direction of the endpoint of the robot manipulator and target reasonably by using a position reward function and a direction reward function, respectively. Therefore, a posture reward function can make the algorithm generate more reasonable actions to be executed by the robot manipulator and thereby improve the efficiency of the algorithm.

A. Position Reward Function

In a random environment, the Euclidean distance between the end of the robot manipulator and target can be used to reflect the current state of the robot. The position reward function is designed in this article consists of two items: 1) the task status and 2) the distance guide. The task status item reflects the result of the trajectory planning, that is, whether the robot manipulator reaches the position of the target that appears in space randomly. The purpose of the distance guide item is to motivate the robot manipulator to approach the target point quickly.

Distance Guide Item: In order to motivate the robot manipulator to approach the target point T quickly, the distance guide item is represented by the Euclidean distance D_{PT}, between the end of the robotic arm P and the target T.

Task Status Item: The task status item is modeled by D_{PT}. The smaller the D_{PT}, the more likely it is that the robot manipulator will reach the target. The task status item is represented by parameters J_{reach}: as shown in

$$J_{\text{reach}} = \begin{cases} 0, & D_{PT} > \beta \\ 1, & D_{PT} < \beta \end{cases} \tag{1}$$

where β is adjustable according to the actual requirements of the environment, the value of β is set to 0.01 in this article.

By combining the task status item and distance guide item, the position reward function is designed as shown in

$$R_{\text{position}}(D_{PT}) = J_{\text{reach}} - D_{PT}. \tag{2}$$

B. Direction Reward Function

On the basis of the guidance of the position reward function, by adding a direction guide, the robot manipulator can obtain more information and reach the target faster.

The direction reward function is modeled by the relationship between two vectors in three-dimensional space: the expected direction and the actual direction of the end of the robot manipulator. As shown in Fig. 2, PT is the expected motion direction, which is represented by $\overrightarrow{V_{PT}}$, and PP' is the actual motion direction, which is represented by $\overrightarrow{V_{PP'}}$. The arithmetic expressions of $\overrightarrow{V_{PT}}$ and $\overrightarrow{V_{PP'}}$ are formulated in (3) and (4) as follows:

$$\overrightarrow{V_{PT}} = \langle (T_x - P_x), (T_y - P_y), (T_z - P_z) \rangle \tag{3}$$

$$\begin{cases} \overrightarrow{V_{PP'}} = \langle (P'_x/\text{temp}), (P'_y/\text{temp}), (P'_z/\text{temp}) \rangle \\ \text{temp} = \sin((P'_w)) \end{cases} \tag{4}$$

where T_x, T_y, and T_z are the coordinates of the target, P_x, P_y, and P_z are the coordinates of the end of the robot manipulator in the current state, P'_x, P'_y, P'_z, and P'_w is the quaternion of the end of the robot manipulator in the current state, and φ represents the angle between $\overrightarrow{V_{PT}}$ and $\overrightarrow{V_{PP'}}$, which is applied to measure the deviation between the motion vector planned by the algorithm and the expected motion vector. The smaller the φ, the lower the deviation. The arithmetic expressions of

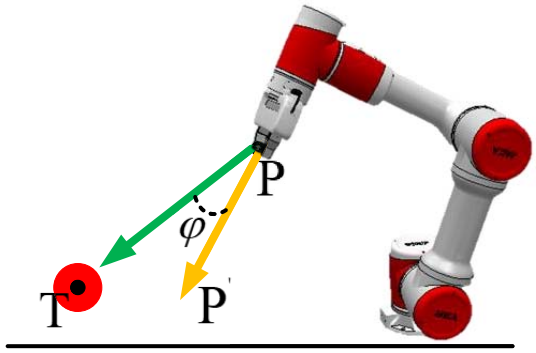

Fig. 2. Scheme of the direction reward function.

φ are formulated in

$$\begin{cases} \varphi = \left| \cos^{-1} \dfrac{\overrightarrow{V_{PT}} \cdot \overrightarrow{V_{PP'}}}{\sqrt{\left(\overrightarrow{V_{PT}} \cdot \overrightarrow{V_{PT}}\right) \times \left(\overrightarrow{V_{PP'}} \cdot \overrightarrow{V_{PP'}}\right)}} \right| \\ \varphi \in [0, \pi]. \end{cases} \tag{5}$$

The direction reward function designed in this article is shown in

$$R_{\text{direction}}(\varphi) = \lfloor \varphi \rfloor_* / 2\pi \tag{6}$$

where $\lfloor \varphi \rfloor_*$ represents an operation in which the value of the function is output normally when the calculation result in $\lfloor \cdot \rfloor_*$ is less than $\pi/2$; otherwise, the result is $\pi - \varphi$.

C. Modeling of the Posture Reward Function

In the process of trajectory planning of the robot manipulator, if only the position reward function or only the direction reward function is used, the performance of the algorithm will be poor. Instead, the position reward function and direction reward function can be combined to form a posture reward function R_{posture}, as shown in

$$R_{\text{posture}}(D_{PT}, \varphi) = R_{\text{position}}(D_{PT}) - R_{\text{direction}}(\varphi). \tag{7}$$

III. Stride Reward Function

The purpose of the stride reward function is to not only enable the robot manipulator to reach the target accurately but also make the movement distance of the robot manipulator is as small as possible for the optimal trajectory we expect. This allows the manipulator to reach the target quickly and reduces the energy consumption during the operation of the robot manipulator. The stride reward function is modeled by the position reward function and the movement distance reward function. The position reward function remains the same as in Section II-A.

A. Movement Distance Reward Function

In this article, we take the average movement distance of each joint as a constraint condition while the robot manipulator is running and model the movement distance reward function. It is difficult to obtain the movement distance of

each joint directly during the operation of the robot manipulator. Therefore, we start from the speed of each joint of the robot manipulator to calculate the distance of each joint. We define the joint velocity vector of the robot manipulator as

$$\overrightarrow{V} = [v_1,\ v_2,\ v_3, \ldots, v_N],\ N = \text{number of joints}. \quad (8)$$

The movement distance reward function R_{move} is shown in

$$R_{\text{move}}\left(\overrightarrow{V}\right) = \Delta t * \left(\overrightarrow{V}\ \cdot\ \overrightarrow{V}\right)/N \quad (9)$$

where Δt represents the working frequency of the robot manipulator, that is, the robot manipulator runs according to the speed command every time at Δt. N is the number of joints of the robot manipulator. In this article, we set Δt as 0.05, and N as 6.

B. Modeling of the Stride Reward Function

The stride reward function proposed in this article is a combination of the position reward function and the movement distance reward function. We use the position and the movement distance of each joint of the robot manipulator as constraints to promote the policy of the trajectory planning learned by the algorithm, which can ensure the target is reached and the movement distance of each joint of the robot manipulator is reduced.

The stride reward function designed in this article is shown in

$$R_{\text{stride}}\left(D_{PT}, \overrightarrow{V}\right) = R_{\text{position}}(D_{PT}) - R_{\text{move}}\left(\overrightarrow{V}\right). \quad (10)$$

IV. Stage Incentive Mechanism

The stride reward function will restrict the motion of the robot manipulator. Actually, at the beginning of the task, we do not hope the robot was restricted, we hope the robot manipulator can move boldly to approach the target at this time. Therefore, we proposed a stage incentive mechanism.

A. Hard-Stage Incentive Reward Function

We use an adjustable coefficient γ to achieve the different reward functions at different stages of the task during the operation of the robot manipulator. The mechanism of the hard-stage incentive divides the task of trajectory planning into two stages, including the fast approach area and the slow adjustable area, as shown in Fig. 3. In the fast approach area, the posture reward function is used to prompt the robot manipulator to approach the target quickly. In the slow adjustable area, the stride reward function is used as an incentive mechanism.

In this article, we use $D_{PT} = 0.5$ as the boundary to divide the fast approach area and the slow adjustable area. (If $D_{PT} > 0.5$, the posture reward function has short time to work, this will reduce the exploration of algorithm, and meanwhile the stride reward function has long time to work, this will result in slow convergence of the algorithm; If $D_{PT} < 0.5$, it means that when the robotic manipulator is closed to the target, the algorithm switches the reward function. At this time, there is not enough space for agent to adapt the stride reward function, which will keep the algorithm from converging. Therefore, in order to avoid the above situation, the mean value of D_{PT} is usually taken.) The relationship between the adjustable coefficient γ and the motion area of the robot manipulator is shown in Fig. 4.

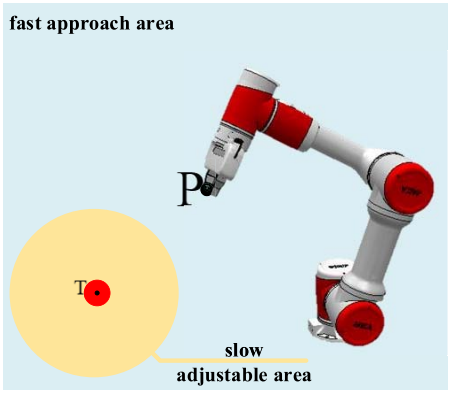

Fig. 3. Scheme of the hard-stage incentive reward function.

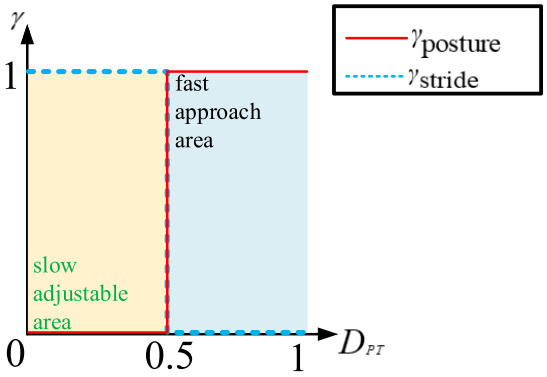

Fig. 4. Diagram of the adjustable coefficient γ.

The value of γ can be calculated by

$$\gamma = \begin{cases} \begin{bmatrix} \gamma_{\text{posture}} = 1 \\ \gamma_{\text{stride}} = 0 \end{bmatrix}^T, & P \in \text{fast approach area} \\ \begin{bmatrix} \gamma_{\text{posture}} = 0 \\ \gamma_{\text{stride}} = 1 \end{bmatrix}^T, & P \in \text{slow adjustable area}. \end{cases} \quad (11)$$

The mechanism of the hard-stage incentive reward function R_{HAR} we proposed is shown in

$$\begin{aligned} R_{\text{HAR}} &= \gamma \left[R_{\text{posture}}(D_{PT}, \varphi)\ R_{\text{stride}}\left(D_{PT}\ \overrightarrow{V}\right)\right]^T \\ &= \left[\gamma_{\text{posture}}, \gamma_{\text{stride}}\right]\left[R_{\text{posture}}(D_{PT}, \varphi)\, R_{\text{stride}}\left(D_{PT}\ \overrightarrow{V}\right)\right]^T. \end{aligned} \quad (12)$$

B. Soft-Stage Incentive Reward Function

Although the hard-stage incentive reward function achieved good results in experiments, we found that it has potential stability problems. That is, the adjustment process is rough, as it is easy to cause fluctuation of the reward curve when changing the reward function, which results in unstable factors for the algorithm. The switching process of the hard-stage incentive reward function is similar to the bang-bang control

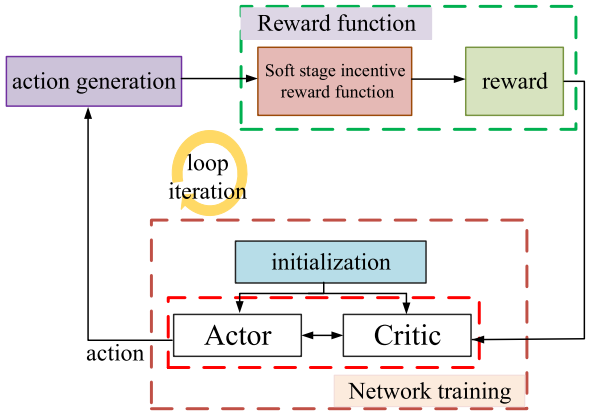

Fig. 5. Diagram of the training process for DRL with an AC frame.

in the classic control, and the method is bound to affect the stability of the algorithm. To solve this problem, we proposed the soft-stage incentive reward.

In this article, the weight coefficient $\alpha = [\alpha_1\ \alpha_2]$ is introduced to model a soft-stage incentive reward function, as shown in

$$\alpha_1 = f(D_{PT}) = 1 - \lfloor D_{PT} \rfloor_-^{\sigma_1} \tag{13}$$

$$\alpha_2 = f(D_{PT}) = \lfloor D_{PT} \rfloor_-^{\sigma_2} \tag{14}$$

where $\lfloor \cdot \rfloor_-$ represents an operation constraining the value of D_{PT} in the range [0, 1], and σ_1 and σ_2 can be adjusted according to the actual situation of the task. In this article, we set $\sigma_1 = \sigma_2 = 1$ according to experimental experience.

The soft-stage incentive reward function, which adjusts the proportions of different reward functions through α, is defined as

$$R_{\text{SAR}} = \alpha_1 R_{\text{stride}}\left(D_{PT}, \vec{V}\right) + \alpha_2 R_{\text{posture}}(D_{PT}, \varphi). \tag{15}$$

The soft-stage incentive reward function does not need to divide the working space of the robot manipulator. According to the real-time change of the weight coefficient α, the reward function is adjusted dynamically and continuously.

V. Implementation of the Reward Function

As shown in Fig. 5, the learning process of the robot manipulator mainly consists of four stages: initialization, action generation, reward calculation and network training. The overall process is summarized in Algorithm 1.

VI. Experimental Results and Discussions

In the experiment, we set 10 000 episodes, and each episode has 50 steps. We used four indicators to evaluate the performance of our method: 1) convergence rate (expressed by the episode E_{start} ($0 < E_{\text{start}} < 10\,000$) when the algorithm starts to converge); 2) the reward of each episode R_{episode} is shown in (16) (during the experiment, we set any time step N in an episode to complete the trajectory planning and to immediately stop the current episode and enter the next episode; therefore, the total reward was calculated within the time step before completing the trajectory planning in each episode.);

Algorithm 1 Trajectory Planning Algorithm With the Soft-Stage Incentive Reward Function

Input: Environment state space S.
Output: Action a

```
Initialize Actor Network μ(S|θ_μ) and Critic Network Q(S|θ_Q)
for episode = 1 to M do
  for t = 1 to T do
    a_t ← μ(S|θ_μ)
    R_SAR ← F(s, a)
    reward = R_SAR
    Update weight of Actor Network θ_μ
    Update weight of Critic Network θ_Q
  end for
end for
```

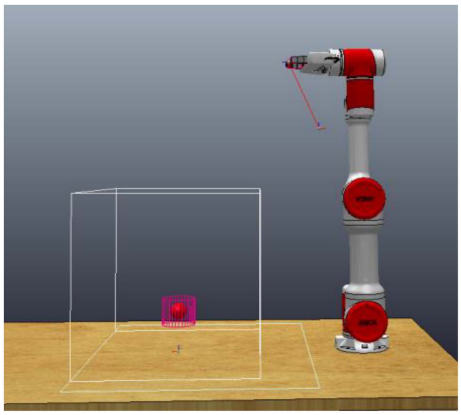

Fig. 6. Simulation environments for the robot manipulator.

3) the average number of steps to complete the task (it is impossible for the robot manipulator to complete the task of trajectory planning in one step); and 4) standard deviation V_{STDEV} is shown in (17), is used to judge the stability and robustness of the algorithm

$$R_{\text{episode}} = \sum_{s=1}^{N} \text{Reward}\ (1 \le N \le 50) \tag{16}$$

$$\begin{cases} \overline{R} = \frac{1}{(10000 - E_{\text{start}} + 1)} \sum_{i=E_{\text{start}}}^{i=10000} R_{\text{episode}} \\ V_{\text{STDEV}} = \sqrt{\frac{\sum_{i=E_{\text{start}}}^{n} \left(R_{\text{episode}}^{i} - \overline{R}\right)^2}{n-1}}\ (n = 10000). \end{cases} \tag{17}$$

Simulation experiments were conducted in V-REP [25]. A random environment was initialized as shown in Fig. 6. The red ball is the target that randomly appears in the workspace. An additional reward of +20 was given after each task is successful.

A. Posture and Stride Reward Function

In this section, three types of reward functions, basic [a sparse reward function as shown in (18)], posture, and stride, were applied to two DRL methods. (According to experimental verification, DDPG and SAC could not converge even after a long training period based on a sparse reward function, so we

TABLE I
RESULTS WITH THE STAGE INCENTIVE REWARD FUNCTION

Method	Reward Function	Train				Evaluation		
		Episode	Reward	Step	Standard deviation	Reward	Step	Success rate
SAC	Basic	——	——	——	——	——	——	——
	Posture	5244	9.956±1.352	15	18.161±1.430	13.902	10	89.2%
	Stride	6773	16.125±0.833	12	9.728±1.049	16.584	12	90.8%
	HAR	5369	15.366±0.403	**12**	10.358±0.348	**17.349**	**10**	**93.2%**
	SAR	**3593**	**16.830±0.255**	**10**	**7.594±0.222**	**17.969**	**7**	**97.0%**
DDPG	Basic	——	——	——	——	——	——	——
	Posture	5879	15.644±0.518	9	11.625±3.073	14.871	11	90.4%
	Stride	7988	16.142±0.827	11	8.872±2.505	16.163	11	88.6%
	HAR	6385	**17.266±0.276**	**9**	**6.688±0.549**	**17.361**	**8**	**93.8%**
	SAR	**4781**	**18.076±0.166**	**8**	**4.283±0.305**	**18.96**	**5**	**99.6%**

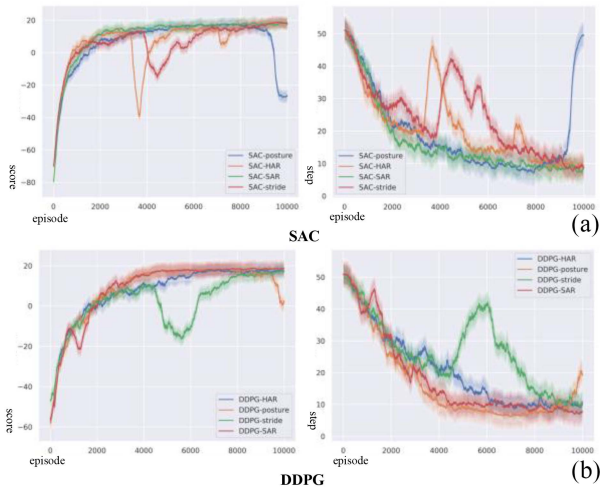

Fig. 7. Diagram of the convergence process with the different reward functions in the training.

do not discuss these scenarios.) During the experiments, we initialized the same working environment 20 times

$$R_{\text{basic}} = \begin{cases} 1, \text{ task is done} \\ 0, \text{ task not done.} \end{cases} \tag{18}$$

After all the methods converged, we calculated the convergence rate, the mean reward of each episode, the average steps to complete the task, and the standard deviation of the latter, as summarized in Table I. The changing process of the reward and the average steps to complete the task in the training for each method are displayed in Fig. 7, and the changing process in the evaluation is shown in Fig. 8.

From Table I, we can see that SAC converged faster in general. The convergence rate of SAC with posture reward function was 10.8% faster than that of DDPG with the posture reward function, and the convergence rate of SAC with the stride reward function was 17.9% faster than that of DDPG with the stride reward function. Compared with the stride reward function, the convergence rate of the posture reward function increased by 22.6%–26.4%. However, the standard deviation of the stride reward function was 23.7%–46.4% lower than that if the posture reward function, and the mean reward of the stride reward function was 3.2% higher than that of the posture reward function in DDPG.

As shown in Fig. 7, during the training, the posture reward function fluctuated to a large extent after convergence; in contrast, the stride reward function converged slowly, fluctuated

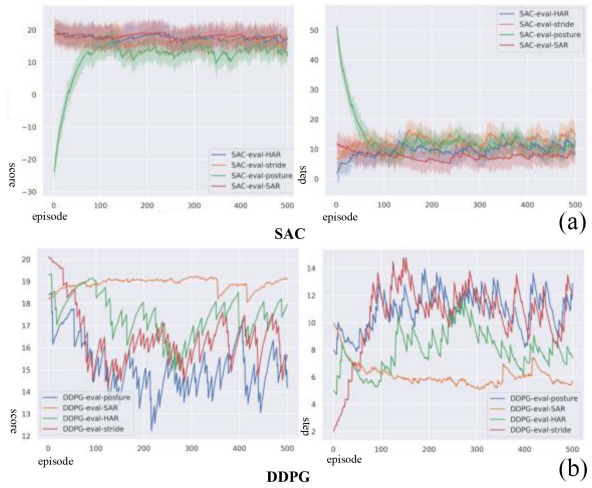

Fig. 8. Diagram of the process with the different reward functions in the evaluation.

greatly before convergence, and was stable after convergence. The reason is simple, the posture reward function guides the robot manipulator closer to the target with distance and direction constraints, which is more oriented to complete the task. However, its stability is poor, and slight interference can make the manipulator move away from the target quickly, resulting in mission failure. The stride reward function takes the distance and the movement distance of each joint of the robot manipulator as constraints to guide the robot manipulator to approach the target. Hence, the manipulator will not suddenly move significantly. Compared to the posture reward function, the stride reward function is more cautious. The fluctuations in training are caused by the agent's exploration. Under the dual effects of exploration and the stride reward function's own characteristics, the robot manipulator cannot reach the target quickly.

During the evaluation, we conducted 500 random trials, used the trained model to realize the trajectory planning with the robot manipulator, and calculated its success rate. As shown in Table I, the success rates of DDPG and SAC based on the posture reward function were 90.4% and 89.2%, respectively, and the success rates of DDPG and SAC based on the stride reward function were 88.6% and 90.8%, respectively.

Generally speaking, the posture reward function completes trajectory planning with fewer average steps, and the stride reward function obtains more rewards. In terms of improving the convergence rate, the posture reward function is more advantageous. The stride reward function plays a more important role in improving algorithm stability.

B. Hard-Stage Incentive Reward Function

It can be seen from the above experiments that in a complex environment, the dense reward function will achieve better results, but there are still some defects. In this section, we applied our hard-stage incentive reward function (hereafter referred to as HAR) to SAC and DDPG, and the convergence results are shown in Table I. In the training, the changing process of the reward and the average steps to complete the task

in the training for each method are shown in Fig. 7, and in the evaluation, the changing process for each method is visualized in Fig. 8.

As shown in Table I, in the process of training, for robustness, the standard deviation decreased by about 42.6% with HAR compared to the posture reward function. The convergence rate of the HAR reward function was about 20.4% faster than that of the stride reward function, but it was slower than the posture reward function. When the mechanism of the hard-stage incentive adjusted the reward function used in different stages, the switch-type adjustment method was adopted without a smooth transition process. This is also one of the reasons why the SAC with HAR fluctuated greatly in the training, as shown in Fig. 7(a), although its performance was not obvious in the DDPG.

In the evaluation, SAC and DDPG were greatly improved in both convergence performance and robustness when HAR was used. The reward increased to 24.7%, the average steps decreased by 16.7%–27.2%, and the success rate for trajectory planning increased by 2.4%–5.2%.

C. Soft-Stage Incentive Reward Function

Although HAR is able to offer improvements in both the training and evaluation processes, its learning efficiency and robustness still need to be improved.

In the last set of experiments, the soft-stage incentive reward function (hereafter referred to as SAR) was used. As shown in Table I, the results of SAR were superior in all cases. In the training, the changing process of the reward and the average steps to complete the task in the training for each method are shown in Fig. 7, and the changing process in the evaluation is shown in Fig. 8.

In the training, compared with the above three reward functions, the convergence rate was accelerated by 18.7%–40.1% in DDPG and by 31.4%–46.9% in SAC. For the convergent mean reward, the promotion was between 4.6%–15.5% in DDPG and 4.4%–9.5% in SAC. The performance of robustness was also excellent; the standard deviation decreased by 35.9%–63.2% in DDPG and by 21.9%–26.7% in SAC. This shows SAR has a great convergence rate, stability, and robustness. Why does it work so well? One reason is that it combines the advantages of the posture reward function and the stride reward function to ensure fast and stable convergence in the early stages of exploration. Another reason is that it solves the switch adjustment mode of HAR by smoothing the transition between different reward functions in different stages.

The convergence rate of SAR in DDPG was slower than that of SAR in SAC, but other indicators were better in DDPG than in SAC. With the use of the model obtained by the DDPG with SAR for evaluation, the success rate of the trajectory planning reached 99.6%. The average number of steps to complete the trajectory planning was 5. From Fig. 8, compared with the other three reward functions, we can observe that SAR needs fewer steps to realize trajectory planning of the robot manipulator, obtains more rewards, and is more stable.

VII. Conclusion

To deal with the inefficiency, instability, and blindness of DRL-based methods in trajectory planning, this article proposed three dense reward functions: 1) the posture reward function; 2) stride reward function; and 3) the mechanism of stage incentive. The posture reward function can reduce the blindness of exploration to accelerate the learning process, and the stride reward function can make the learning process more stable. However, the soft-stage incentive reward function exhibits the advantages of both, offering faster convergence, higher stability, and greater robustness. The experimental results showed that state-of-the-art DRL methods using the proposed reward functions will have the best performance.

In the future, we will further explore the mechanism of reward shaping, and we plan to apply DRL methods to other complex tasks with robot manipulators.

References

[1] X. Lu and Y. Jia, "Trajectory planning of free-floating space manipulators with spacecraft attitude stabilization and manipulability optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 12, pp. 7346–7362, Dec. 2021.

[2] L. Wang, X. Lai, P. Zhang, and M. Wu, "A control strategy based on trajectory planning and optimization for two-link underactuated manipulators in vertical plane," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 6, pp. 3466–3475, Jun. 2022.

[3] H.-I. Lin and M.-F. Hsieh, "Robotic arm path planning based on three-dimensional artificial potential field," in *Proc. 18th Int. Conf. Control Autom. Syst. (ICCAS)*, Daegwallyeong, South Korea, 2018, pp. 740–745.

[4] S. N. Gai, R. Sun, S. J. Chen, and S. Ji, "6-DOF robotic obstacle avoidance path planning based on artificial potential field method," in *Proc. 16th Int. Conf. Ubiquitous Robots (UR)*, Jeju, South Korea, 2019, pp. 165–168.

[5] Z. Mu, H. Yuan, W. Xu, T. Liu, and B. Liang, "A segmented geometry method for kinematics and configuration planning of spatial hyper-redundant manipulators," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 5, pp. 1746–1756, May 2020.

[6] S. Fang, X. Ma, Y. Zhao, Q. Zhang, and Y. Li, "Trajectory planning for seven-DOF robotic arm based on quintic polynormial," in *Proc. 11th Int. Conf. Intell. Human-Mach. Syst. Cybern. (IHMSC)*, Hangzhou, China, 2019, pp. 198–201.

[7] S. Fang, X. Ma, J. Qu, S. Zhang, N. Lu, and X. Zhao, "Trajectory planning for seven-DOF robotic arm based on seventh degree polynomial," in *Proc. Chin. Intell. Syst. Conf. (CISC)*, vol. 593, 2019, pp. 286–294.

[8] W. He, H. Gao, C. Zhou, C. Yang, and Z. Li, "Reinforcement learning control of a flexible two-link manipulator: An experimental investigation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 12, pp. 7326–7336, Dec. 2021.

[9] K. Katyal, I.-J. Wang, and P. Burlina, "Leveraging deep reinforcement learning for reaching robotic tasks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Honolulu, HI, USA, 2017, pp. 490–491.

[10] K. Kamali, I. A. Bonev, and C. Desrosiers, "Real-time motion planning for robotic teleoperation using dynamic-goal deep reinforcement learning," in *Proc. 17th Conf. Comput. Robot Vis. (CRV)*, Ottawa, ON, Canada, 2020, pp. 182–189.

[11] S. Wen, J. Chen, S. Wang, H. Zhang, and X. Hu, "Path planning of humanoid arm based on deep deterministic policy gradient," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Kuala Lumpur, Malaysia, 2018, pp. 1755–1760.

[12] B. Sangiovanni, G. P. Incremona, M. Piastra, and A. Ferrara, "Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning," *IEEE Contr. Syst. Lett.*, vol. 5, pp. 397–402, 2021.

[13] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1612–1619, Apr. 2022.

[14] S. Kumra, S. Joshi, and F. Sahin, "Learning robotic manipulation tasks via task progress based Gaussian reward and loss adjusted exploration," *IEEE Robot. Autom. Lett.*, vol. 7, no. 1, pp. 534–541, Jan. 2022.
[15] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[16] T. D. Le, A. T. Le, and D. T. Nguyen, "Model-based Q-learning for humanoid robots," in *Proc. 18th Int. Conf. Adv. Robot. (ICAR)*, Hong Kong, 2017, pp. 608–613.
[17] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, May 2016, pp. 1–14.
[18] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, New York, NY, USA, Jun. 2016, pp. 1928–1937.
[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 1861–1870.
[21] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative, multi-agent games," 2021, *arXiv:2103.01955*.
[22] Z. Zhang, D. Wang, and J. Gao, "Learning automata-based multiagent reinforcement learning for optimization of cooperative tasks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4639–4652, Oct. 2021.
[23] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep reinforcement learning with optimized reward functions for robotic trajectory planning," *IEEE Access*, vol. 7, pp. 105669–105679, 2019.
[24] S. Jang and M. Han, "Combining reward shaping and curriculum learning for training agents with high dimensional continuous action spaces," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, 2018, pp. 1391–1393.
[25] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IROS*, Tokyo, Japan, Nov. 2013, pp. 1321–1326.

Gang Peng (Member, IEEE) received the doctoral degree in control science and engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2002.

He is currently an Associate Professor with the Department of Automatic Control, School of Artificial Intelligence and Automation, HUST. His research interests include intelligent robots, machine vision, multisensor fusion, machine learning, and artificial intelligence.

Dr. Peng is a Senior Member of the China Embedded System Industry Alliance, China Software Industry Embedded System Association, and Chinese Electronics Association, and a member of the Intelligent Robot Professional Committee of the Chinese Association for Artificial Intelligence.

Jin Yang (Student Member, IEEE) received the B.S. degree in automation from Tianjin Polytechnic University, Tianjin, China, in 2019, and the M.S. degree in control science and engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2022. He is currently pursuing the doctoral degree with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China.

His research interests focus on the deep learning methods for video understanding, intelligent robots, and reinforcement learning.

Xinde Li (Senior Member, IEEE) received the doctoral degree in control science and engineering from the Huazhong University of Science and Technology, Wuhan, China, in June 2007.

He worked with the School of Automation, Southeast University, Nanjing, China, in December 2007, where he is a Professor and a Doctoral Tutor. From January 2012 to January 2013, he was a National Public Visiting Scholar with Georgia Tech Visit and exchange for one year. From January 2016 to the end of August 2016, he worked as a Research Fellow with the ECE Department, National University of Singapore, Singapore. His research interests include intelligent robots, machine vision perception, machine learning, human–computer interaction, intelligent information fusion, and artificial intelligence.

Prof. Li was selected as an IEEE Senior Member in 2016.

Mohammad Omar Khyam (Member, IEEE) received the doctoral degree in electrical and electronics engineering from the University of New South Wales, Sydney, NSW, Australia, in 2015.

He, respectively, engaged postdoctoral research with the National University of Singapore, Singapore; Nanyang Technological University, Singapore; and Virginia Polytechnic Institute and State University, Blacksburg, VA, USA. He has been with the School of Engineering and Technology, Central Queensland University, Norman Gardens, QLD, Australia, since July 2019. His main research interests include intelligent robots, machine vision perception, and artificial intelligence.