IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 23, 2020, accepted June 4, 2020, date of publication June 9, 2020, date of current version June 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3001130

A Reinforcement Learning-Based Framework for Robot Manipulation Skill Acquisition

DONG LIU[1,2], (Member, IEEE), ZITU WANG[1], BINPENG LU[1], MING CONG[1], HONGHUA YU[1], AND QIANG ZOU[1,2]

[1]School of Mechanical Engineering, Dalian University of Technology, Dalian 116024, China
[2]Jiangsu Research Institute Company, Ltd., Dalian University of Technology, Changzhou 213161, China

Corresponding author: Ming Cong (congm@dlut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61873045, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20180190, in part by the Dalian Sci & Tech Innovation Foundation Program under Grant 2019J12GX043, and in part by the Fundamental Research Funds for the Central Universities under Grant DUT19JC56.

ABSTRACT This paper studies robot manipulation skill acquisition based on a proposed reinforcement learning framework. Robot can learn policy autonomously by interacting with environment with a better learning efficiency. Aiming at the manipulator operation task, a reward function design method based on objects configuration matching (OCM) is proposed. It is simple and suitable for most Pick and Place skills learning. Integrating robot and object state, high-level action set and the designed reward function, the Markov model of robot manipulator is built. An improved Proximal Policy Optimize algorithm with manipulation set as the output of Actor (MAPPO) is proposed as the main structure to construct the robot reinforcement learning framework. The framework combines with the Markov model to learn and optimize the skill policy. A same simulation environment as the real robot is set up, and three robot manipulation tasks are designed to verify the effectiveness and feasibility of the reinforcement learning framework for skill acquisition.

INDEX TERMS Robot skill acquisition, reinforcement learning, reword function, MAPPO.

I. INTRODUCTION

Due to its unique operational flexibility, robot arm has a wide range of applications in industrial production and life service. Currently calibrating robots can perform repetitive tasks well like object picking and placing, but in many cases, it is still not flexible to adjust itself for new tasks [1]. With the emergence of advanced robots, the demand for teaching robots complex skills increases. When a robot is brought into a new practical general environment to perform complex tasks, and it has only limited knowledge. One way to further acquire knowledge is to explore and learn by interacting with the environment [2]. The current robotic skill acquisition usually adopts the method of teaching programming, which belongs to low-level motion planning. While higher-level planning such as behavior sequence planning and task planning is still in its infancy, which is one reason for robot difficult to obtain autonomy in complex dynamic environments.

Making robots have human-like intelligence, and acquire new skills in dynamic and uncertain environment is a goal that the intelligent robot field has been pursuing. Skills acquisition encompasses various forms of learning. Robots not only learn by observing the behaviors of other agents, but also learn new skills through exploring and trying [3]. Reinforcement learning in the field of robotics becomes a hot research topic because of its successful applications in recent years [4]. Reinforcement learning method takes the environmental feedback as input, and achieves strategy optimization using evaluative feedback function through trial-and-error interaction with a dynamic environment [5], [6]. This is also an instinctive way of learning when human faces an unknown state in nature. Therefore, applying reinforcement learning for robotic skill acquisition, the robot has the capacity of human-like learning, which can avoid lengthy robot programming, while no technical expertise is required for the operator.

When robot performs reinforcement learning, it has to be able to generalize to new actions to become autonomous. Most approaches concern that robots derive efficient representations from high-dimensional sensory inputs, and use these to generalize past experience to new situations [7], [8]. Generally, this is a hard problem, since the search space

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

the robot has to explore is potentially huge [9]. The high dimensional robot joint information and input image will take a long training time. Thus, the sensory-action space design is an important issues. Another key part is the design of reward function since the optimization of strategy is based on the cumulative reward of each state. How to design the reward function will affect the efficiency and effectiveness of skill learning in the framework of robot reinforcement learning [10]. It is still a difficult task to design a suitable reward function, since different tasks are often difficult to have a unified reward function. A suitable reward function to enable robots to learn efficiently and robustly in different similar tasks is necessary.

Integrated the sensory-action space and reward function design, this paper proposes a new reinforcement learning-based framework for robot Pick and Place skill acquisition. The primary contributions are summarized as follows:

1) Considering the features of robot manipulation tasks, low dimensional state space and higher-level action set are proposed to overcome the problems arising from high-dimensional and continuous sensory-action space. The object pose combines the pose of robot end effector and the gripper status to form the state of reinforcement learning model.

2) A reward function which is suitable for the Pick and Place tasks is designed based on objects configuration matching (OCM). Euclidean distance is used to calculate the similarity of the configuration vectors, which focuses on length property. Pearson coefficient is used to calculate correlation of the vectors, which focuses on direction property. The instant reward is a function of similarity. This can improve the learning efficiency with a more reasonable action.

3) Based on robot Markov model, the reinforcement learning framework is constructed through an improved proximal policy optimize algorithm as the main structure, which is called MAPPO. It can autonomously learn and optimize the manipulation skill policy to accumulate task experience in new environment.

II. RELATED WORK

The emergence of various reinforcement learning-based models and algorithms provide theoretical support for more and more applications. End-to-end training of visuomotor policies [11] are proposed. They directly input robot joint information and original image as the state to a deep convolutional neural network (DCNN) with 92,000 parameters. The policies are trained to output robot motor torque as an action. The Google team conducted similar researches in the following three directions: 1) Model-free reinforcement learning from original experience. Depth Determination Strategy Gradient Algorithm (DDPG) is proposed to train robot stacking blocks in virtual environment [12]. The input is the joint information of robot arm and gripper, while the output is the action performed by robot arm; 2) learning internal physical model of the object. Finn and Levine [13] used end-to-end learning model to train the robot randomly grabbing objects in a container; 3) Learning with human assistance. Similar to the above, Gu *et al.* [14] added artificial guidance as an initial strategy to improve learning efficiency.

Some learning algorithms such as Proximal Policy Optimization (PPO) [15] and Critics of Asynchronous Advantage Actors (A3C) [16] are proposed to make robot learn efficiently. Additionally, the OpenAI team proposed a hierarchical reinforcement learning algorithm [17] to decompose complex tasks into multiple high-level action sequences, which is an effective way to solve dimensional disaster. Similarly, the layered guiding framework [18] divides into several subtasks to learn. Other methods for improving learning ability include speeding up strategy search through human teaching actions [19], using the knowledge acquired early in agent live [20], and parameterizing initial teaching as a dynamic motion primitive [9], etc.

As reinforcement learning becomes autonomous, the design of reward function that elicit desired behaviors becomes important and difficult [21]. Ratner *et al* [22] derived an approach to infer a common reward by using inference to generate a posterior distribution over the true reward from independently designed rewards. Guo *et al.* [23] presented an adaptation of policy-gradient for reward design for learning a reward-bonus function to improve Monte Carlo Tree Search algorithm. MacGlasha *et, al* [24] presented a system to ground natural language commands to reward functions that captured a desired task, and used natural language as an interface for specifying rewards. Palan *et al.* [25] used demonstrations to learn a coarse prior over the space of reward functions, to reduce the effective size of the space from which queries are generated.

The various reinforcement learning models and algorithms make the theory of reinforcement learning more mature [14], [26], [27]. While reinforcement learning is still in its infancy for robot manipulation skill acquisition. The methods have struggled due to various practical challenges such as skill representation, reward definition and exploration training [28]. In this work, we focus on sensory-action space modelling and reward function design to build the reinforcement learning framework through MAPPO for robot manipulation skill acquisition.

III. ROBOT MARKOV MODEL FOR REINFORCEMENT LEARNING

A. MARKOV MODELING OF ROBOT MANIPULATOR

The robot manipulator continuously samples in the process of interacting with the environment, and acquires task skills through learner training. The Markov model is described by the tuples $< s, a, P, R, \gamma >$:

State (s): The state information includes two parts, which are the object state information and robot arm and gripper state information in the environment. According to the learning task, the poses of all the objects related to the task are extracted by the RGBD sensor. The manipulated object pose

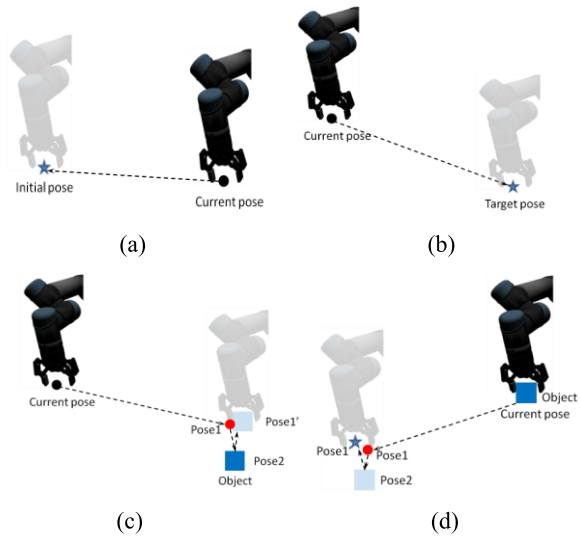


FIGURE 1. Action path (a)arm_init, (b)move, (c)pick, (d)place.

combine the post of robot end effector and the gripper status to form the state of reinforcement learning model. Suppose the number of objects in the environment is N, which is represented by the set $obj_1, obj_2, obj_3, \cdots, obj_n$. At time step t we select an object obj_t and then choose an action which can be described as:

$$d(t) = ((s_{objt}, s_r), a_t) \tag{1}$$

where s_{objt} is the state of obj_t which is expressed as the object pose $(x_i, y_i, z_i, orien_i)$, s_r is the state of robot, which is the end effector pose and the opening and closing status of the gripper.

Action (a): The action set consists of high-order, discrete actions designed by MoveIt!, and interacts with the environment by selecting actions from the set {*arm_init*, *move*, *pick*, *place*$\cdots$}, as shown in Figure 1. *arm_init* indicates the initialization of robot pose, moving from current post to initial post. At this time, the whole robot manipulator is out of the camera's field of view, leaving enough space to obtain the current status information. *move* represents robot movement action that the end effector moves from current post to target post. *pick* means robot grasping action. The end effector moves from current post to pose1 above the object, and moves from pose1 to the grab point pose2, then the gripper closes, and the robot grabs object for a certain distance back to pose1. pose1 has a certain safety distance from the grab point pose2. *place* represents robot placement action. The end effector moves from current post to pose1 above the object, and moves down to pose2, then the gripper opens to place object, and the end effector rises to pose1 again for keeping a certain safety distance.

Strategy (P): The strategy is represented by a neural network. It outputs a probability distribution of discrete actions according to current state, and selects an action from the action set based on the probability distribution.

Reward (R): The reward is computed according to the reward function after the robot manipulator performs an action. It will be discussed next.

γ: Discount factor for calculating cumulative reward.

In the initial state s_0, the robot selects an action a_0 to interact with the environment, transfers to the next state s_1 according to state transition probability $P[s_1|s_0, a]$, and obtains an immediate reward r_0, and then circulates the above process. In this process, the state-action-reward transfer sequence $(s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2 \cdots s_t, a_t, r_t)$ is collected.

B. REWARD FUNCTION DESIGN BASED ON OCM

According to the tasks of most robot arms interacting with objects in environment, we propose a reward function design method based on OCM. Robot operation tasks are often necessary to configure objects in environment to a target configuration. Here Objects Target Configuration (OTC) indicates that the objects are distributed in space according to a certain desired pose relationship. For stacking blocks task, the blocks may need to stack randomly from bottom to top in a configuration of {*blue_cube*, *red_cube*, *yellow_cube*$\cdots$}. Therefore, this paper considers giving the OTC for different tasks, and constructs reward function according to the similarity between OTC and Objects Current Configuration (OCC).

Given the OTC, the mutual pose relationship between the objects can be known. An object can be regarded as a point, and each point has its pose information, and then multiple objects in the environment constitute the point set data. Similarly, the OCC also corresponds to a point set data. The point cloud matching algorithm is used to calculate the relationship between two or more frames of point set data, so as to splice adjacent point set data or calculate the pose transformation matrix of two point set data. Inspired by this, this paper calculates the similarity between OTC and OCC based on the point set matching algorithm. The instant reward is a function of similarity.

In reinforcement learning model, because the agent needs to interact with environment at each step, the objects configuration will change at different times. In the target configuration and the current configuration, the ID of each object can be obtained through image recognition. Then two point sets are determined. It is no longer necessary to find the matching relationship between points. Therefore, we simplify the point cloud matching, and obtain the similarity between OTC and OCC based on vector similarity measurement. Iterating all points in the point set, every two points constitute a vector. Calculating the similarity of the corresponding vector between target configuration and current configuration, the similarity combination of all vectors is the similarity between OTC and OCC. We use Euclidean distance to calculate the similarity of the vectors, which focuses on length property of the vectors. We use Pearson coefficient to calculate the correlation of the vectors, which focuses on direction property of the vectors.

Assuming that there are objects 1, 2, and 3 in environment, and the robot arm needs to interact with the objects to achieve

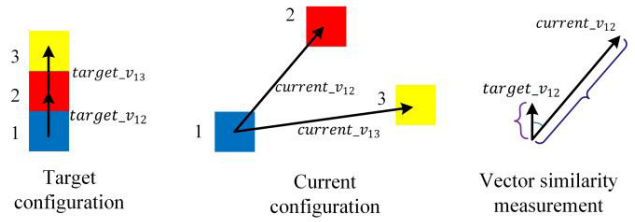


FIGURE 2. Similarity measurement of objects configuration.

a target configuration, as shown in Figure 2. Here the target configuration is given as the dictionary:

$$target_{conf} = \{1 : (x_1, y_1, z_1, orien_1), \\ 2 : (x_2, y_2, z_2, orien_2), 3 : (x_3, y_3, z_3, orien_3) \quad (2)$$

where *orien* represents the object pose, which is quaternion (w, x, y, z).

The OCC is obtained through the sensor:

$$current_{conf} = \{1 : (x'_1, y'_1, z'_1, orien'_1), \\ 2 : (x'_2, y'_2, z'_2, orien'_2), 3 : (x'_3, y'_3, z'_3, orien'_3) \quad (3)$$

Tacking object 1 as a reference, the configuration vectors for object 1 and object 2, object 1 and object 3 are constructed as:

$$\begin{aligned} target\_v12 &= (x_2 - x_1, y_2 - y_1, z_2 - z_1, orien_2 - orien_1) \\ target\_v13 &= (x_3 - x_1, y_3 - y_1, z_3 - z_1, orien_3 - orien_1) \\ current\_v12 &= (x_2 - x_1, y_2 - y_1, z_2 - z_1, orien_2 - orien_1) \\ current\_v13 &= (x_3 - x_1, y_3 - y_1, z_3 - z_1, orien_3 - orien_1) \end{aligned} \quad (4)$$

The configuration vector reflects the relative pose relationship of two objects. Here we only give the configuration vector of object 1 to other objects. The Euclidean distance and Pearson coefficient of the corresponding vectors are computed as:

$$d_{12} = \sqrt{\sum_{k=1}^{n} (target\_v12_k - current\_v12_k)^2} \quad (5)$$

$$d_{13} = \sqrt{\sum_{k=1}^{n} (target\_v13_k - current\_v13_k)^2} \quad (6)$$

Among them, the rewards in intermediate process appear in the form of negative rewards, avoiding the robot arm making unnecessary actions in the interaction process in order to retrieve reward [14]. The reward range associated with the Euclidean distance is $(-m, 0)$. The smaller the distance, the greater the similarity, and the bigger the reward. The reward range associated with the Pearson coefficient is $(-1, 0)$. The greater the relevance, the bigger the reward. R_{final} is the final reward received by the robot after completing task. When the task is not completed in previous period, the reward value of each step is small. When the task is finally completed, the robot manipulator will get a larger reward. The specific values given in the formula need to be adjusted according to the task (7) and (8), as shown at the bottom of the next page. Then the reword function is computed as (9), as shown at the bottom of the next page:

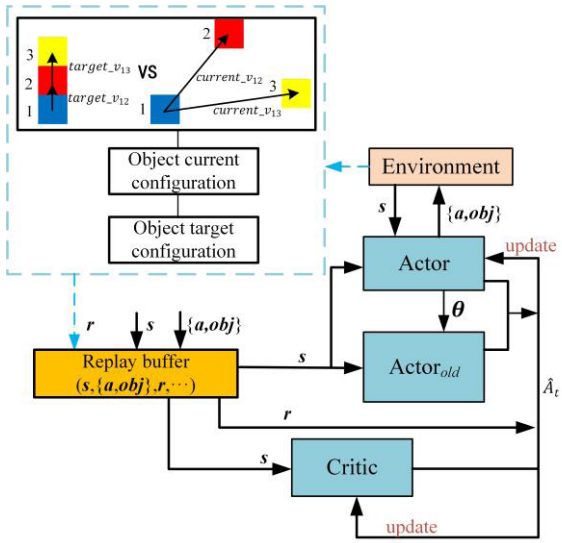


FIGURE 3. The MAPPO model for robot reinforcement learning.

IV. MAPPO FOR ROBOT REINFORCEMENT LEARNING

Here we propose an improved PPO algorithm with manipulation set as the output of *Actor* to construct the reinforcement learning framework for robot manipulation skill learning, called MAPPO. PPO algorithm [15], [29] has lower complexity with using first-order optimization, and can attain the data efficiency and reliable performance. As shown in Figure 3, the *Actor* acts on the environment according to the state and output a manipulation set: $\{a, obj\}$ with selected action a and the specific object *obj* that to be operated. The instant reward r is obtained by comparing the similarity between OTC and OCC. A batch of $[s, \{a, obj\}, r]$ is stored in the replay buffer. The *Critic* calculates an estimator of the advantage function according to the r and s sampled from buffer, which is used to update itself and *Actor*.

The *Critic* in this paper is a state value function $v(s; w)$. Here a neural network is used to construct the *Critic*, and w is the network parameter. The input of the neural network is the state s_t sampled from replay buffer and the output is state value $V(s_t)$. The updating of *Critic* takes the form of a policy gradient method with s_t and r_t which sampled from buffer:

$$minimize(-\sum_{t=1}^{T} \left(\hat{A}_t\right)^2) \quad (10)$$

$$\hat{A}_t = \sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t) \quad (11)$$

where $\hat{A}_t$ is an estimator of the advantage function at timestep t, and T is the length of samples. $-\sum_{t=1}^{T} \left(\hat{A}_t\right)^2$ is used as the loss function of the neural network, and the BP algorithm is used to update the network parameters.

Actor and $Actor_{old}$ are strategy learning networks $\boldsymbol{\pi}(\{a, obj\}|s; \theta)$ and $\boldsymbol{\pi}(\{a, obj\}|s; \theta_{old})$, respectively. θ and θ_{old} are the network parameters. The input is state s_t, and

the output is action a_t and the object obj_t to be operated in the next step. The *Actor* selects actions by the outputting probability of each action, as Gibbs softmax:

$$\pi_t(s,a) = Pr\{a_t = a | s_t = s\} = \frac{e^{p(s,a)}}{\sum_b e^{p(s,b)}} \tag{12}$$

where $p(s,a)$ is the strategy parameter of *Actor* at time t, indicating the probability of selecting each action under state s.

When constructing the *Actor* network, the softmax layer is added after the output layer to obtain the probability of each action and the operated object, which is expressed by $p(s,a)$ and $p(s,obj)$. In order to balance the exploration and utilization in reinforcement learning, the action is selected based on its probability distribution. Actions with high probability are more likely to be selected. It can not only make the best decision of current state, but also try different actions to collect more information.

Actor and $Actor_{old}$ are two neural networks which have the same structure but different parameters. The updating of *Critic* takes the form of a policy gradient method:

$$maxmize(-\sum_{t=1}^{T} \min(\mathrm{r_t}(\theta)\hat{A}_t,{}^{2} \quad clip(\mathrm{r_t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)) \tag{13}$$

$$r_t(\theta) = \frac{\pi_\theta\left((a,obj)_t \mid s_t\right)}{\pi_{old}\left((a,obj)_t \mid s_t\right)} \tag{14}$$

where π and π_{old} are policies of *Actor* and $Actor_{old}$, respectively. The parameter θ is passed to $Actor_{old}$ after *Actor* updates for several times. The second term, $clip(\mathrm{r_t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$, is added as a constrain to avoid an excessively large policy update. Algorithm 1 shows the process of acquiring skills through MAPPO for robot reinforcement learning.

Algorithm 1 Skills Acquiring Through Robot Reinforcement Learning

for $i \in \{1,\ldots,N\}$ **do**
  Run policy π_θ for T timesteps, collecting $\{s_t, (a,obj)_t, r_t\}$
  Estimate advantages $\hat{A}_t = \sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t)$
  $\pi_{\text{old}} \leftarrow \pi_\theta$
  for $j \in \{1,\ldots,M\}$ **do**
    $J(\theta) = \sum_{t=1}^{T} \min[(r_t(\theta)\hat{A}_t, \quad clip(r_t(\theta), 1-\epsilon, 1+\varepsilon,)\hat{A}_t]$
    Update θ by a gradient method w.r.t. $J(\theta)$
  end for
  for $j \in \{1,\ldots,B\}$ **do**
    $L_{BL}(\phi) = -\sum_{t=1}^{T}(\hat{A}_t)^2$
    Update ϕ by a gradient method w.r.t. $L_{BL}(\phi)$
  end for
end for

V. EXPERIMENTAL RESULTS

A. EXPERIMENTAL ENVIRONMENT ESTABLISHMENT

In order to reduce the training cost and hardware damage of physical robots in training process, we built the same simulation environment of the real robot to test algorithm feasibility. The real robot system includes a UR5 robot, a Kinect2 camera and a pneumatic gripper. The simulation environment is built in Gazebo physical engine. By combining all the hardware, the same simulation model as the real robot system is obtained, as shown in Figure 4. The way to control robot in the simulation environment is the same as the way to control robot in real environment. Therefore, tasks such as low-level action planning, high-level teaching and strategy learning can be used in the simulation environment. Since we only consider the poses of objects and end-effector, the differences of camera and gripper does not affect experiment performance. When we unify the robot basic coordinate system of the simulation environment and real environment, both data can be used universally, including object state information and robot state information after coordinate transformation.

However, since the long training time that the moving of robot takes, we build a virtual environment for the learning process. The virtual environment and the simulation

$$\rho_{12} = \frac{\sum_{k=1}^{n}\left(target_{v12_k} - \overline{target_{v12}}\right)\cdot\left(current_{v12_k} - \overline{current_{v12}}\right)}{\sqrt{\sum_{k=1}^{n}\left(target_{v12_k} - \overline{target_{v12}}\right)^2 \cdot \sum_{k=1}^{n}\left(current_{v12_k} - \overline{current_{v12}}\right)^2}} \tag{7}$$

$$\rho_{12} = \frac{\sum_{k=1}^{n}\left(target_{v12_k} - \overline{target_{v12}}\right)\cdot\left(current_{v12_k} - \overline{current_{v12}}\right)}{\sqrt{\sum_{k=1}^{n}\left(target_{v12_k} - \overline{target_{v12}}\right)^2 \cdot \sum_{k=1}^{n}\left(current_{v12_k} - \overline{current_{v12}}\right)^2}} \tag{8}$$

$$R = \begin{cases} -\left(d_{12}^{0.5} + d_{13}^{0.5}\right) + \left(\begin{array}{l}\left(\left(\frac{\rho_{12}+1}{2}\right)^{0.5} - 1\right) \\ +\left(\left(\frac{\rho_{13}+1}{2}\right)^{0.5} - 1\right)\end{array}\right), & if\ OTC \neq OCC \\ R_{final}, & if\ OTC = OCC \end{cases} \tag{9}$$

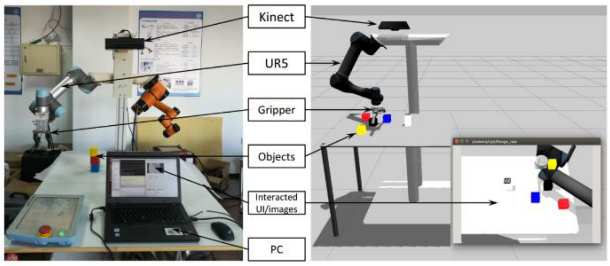


FIGURE 4. Experiment setup.

environment have the same state information of objects and robot. We only consider the results of operation and omit the action process of robot, which reduces the time spent in learning process. The parameters of *Actor* and *Critic* neural network will be saved after being trained in virtual environment, and they will be used in simulation environment for testing.

B. TASKS SETTING AND SKILLS ACQUISITION EXPERIMENTS

The goal of Block Stacking task is to stack a set of blocks into a target configuration. Because of its moderate difficulty and logic, it is widely used in learning experiments. We use a number of wooden blocks with different colors and sizes of 5 cm × 5 cm as the task objects. The blocks are randomly placed on desktop. Three tasks related to Block Stacking are designed to prove the proposed methods.

1) Task a: Robot moves to target location from initial location, and stays until this episode ends. Here blue cube is set as the target location, and other blocks are environmental interference.
2) Task b: A red cube and a yellow cube are placed on table. Robot picks yellow cube and places it on the red cube. Other blocks are environmental interference.
3) Task c: Simulate the sorting task. Three cylinders of different colors and three cubes with holes are placed on table. Robot picks the cylinders and then places them in the cubes of corresponding color.
4) Task d: Four cubes are placed on table. Robot picks red cube and places it on the blue cube, and then picks yellow cube and places it on the red cube, and finally picks green cube and places it on the yellow cube. Other blocks are environmental interference.

The proposed algorithm is feasible for all objects in the working space of the robot arm. The MAPPO algorithm only receives state information of the environment, and outputs actions and objects to be operated. For some special cases, the intermediate rewards are added. For example, when the gripper is closed, if the *Actor* outputs a *pick* action, it should be punished. When the gripper is open, if the *Actor* outputs a *place* action, it should also be punished. The experiment is terminated when task is completed, object leaves the workspace of robot arm, robot joint motion planning fails, or the episode ends.

Parts of the learning process for task c and task d are shown in Figure 5. The video is available online: https://youtu.be/psQLtW0wgZo. During the experiments, the reset function is firstly executed to initialize environment, and the environment state is s_0. The *Actor* takes state s_0 as input, and outputs action a_0 and object obj_0. The reset (a_0, obj_0) function is executed, the robot arm operates the object obj_0 with action a_0, and the function returns a new environment state s_1, instantaneous reward r_0 and the flag of whether the task is completed. The *Actor* performs this operation for several times and stores $[s_t, \{a_t, obj_t\}, r_t]$ in replay buffer. The *Actor* and *Critic* learning function update the network parameters with $[s_t, \{a_t, obj_t\}, r_t]$ stored in buffer, and iterate until the termination condition is reached.

C. PERFORMANCE DISCUSSION

Two experiments are utilized to demonstrate the feasibility and efficiency of the proposed approaches. Task b, c and d are carried out based on the proposed OCM reward function and linear reward function respectively to verify the performance of OCM method. Meanwhile, task a is performed through MAPPO, PPO [15] and A3C [16] algorithm respectively to verify the performance of MAPPO.

Figure 6 shows the cumulative reward curves of MAPPO algorithm using OCM reward function and linear reward function for task b, task c and task d. The abscissa and ordinate of task b are on the left and below, while task c and task d are on the above and right, which are represented by green curve and blue scales respectively. For linear reward function, when the task is not completed, the instant reward of task b, c and d are set to -1, -10 and -20 respectively, while set to 1, 10 and 20 after the task is completed. A total of 900 episodes are conducted in task b, while 5000 and 10000 episodes are conducted in task c and task d respectively. We can see that the solid curves have a faster ascent rate in the early period and less time consumption to reach convergence, which demonstrate the OCM reward function can reduce the exploration in the early period of reinforcement learning and accelerate the convergence speed.

Figure 7 shows the cumulative reward curves for task a. The abscissa and ordinate of MAPPO is on the left and blew, while PPO and A3C algorithm are on the right and above, indicated by yellow and blue scales respectively. We have total 300 episodes performed by MAPPO, while 450 episodes performed by PPO and 650 episodes performed by A3C. The maximum step in each episode is set to 40. In the early exploration period, the more steps an agent attempts in the environment, the more negative cumulative rewards it gets. It takes far less time to reach convergence using OCM reward function than linear reward function, which we believe it is because the OCM reward function based learning method greatly reduces unnecessary and ineffective explorations in the early period.

According to the four tasks, the learning time and average reward are utilized as evaluation indicators for performance

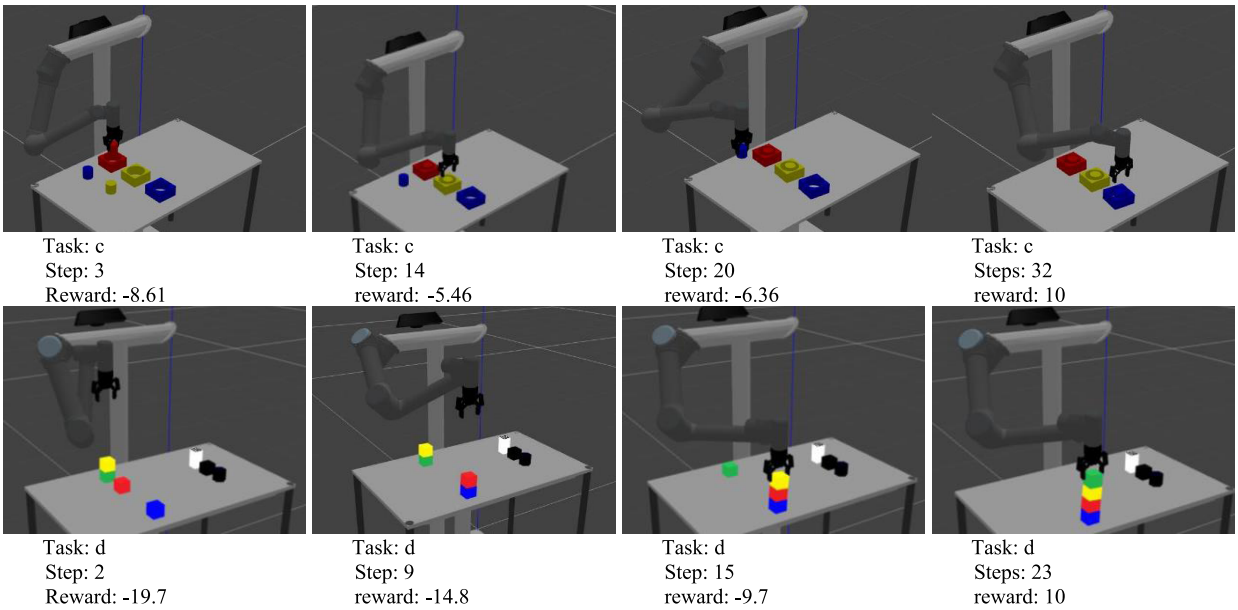


FIGURE 5. Training process of task *c* and task *d*.

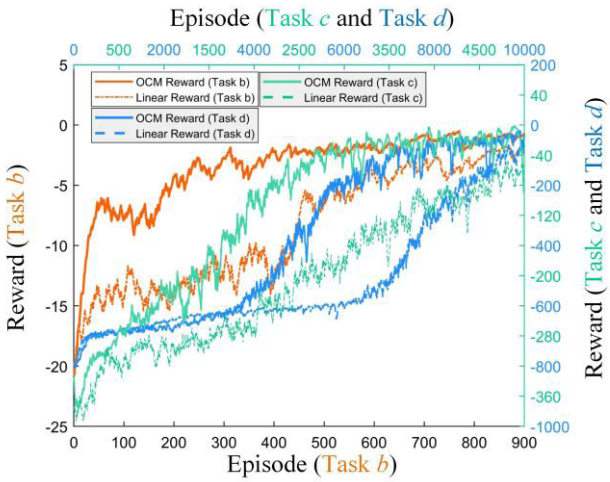


FIGURE 6. Learning convergence curves of MAPPO using OCM reward function and linear reward function for task *b*, *c* and *d*.

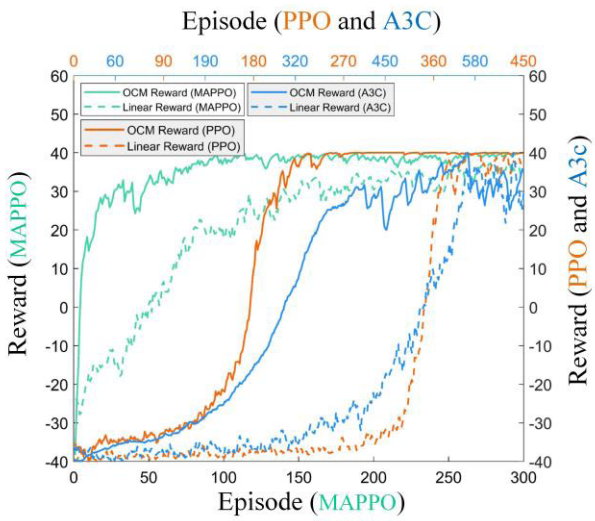


FIGURE 7. Learning convergence curves of MAPPO, PPO and A3C using OCM reward function and linear reward function for task *a*.

comparison. We use the episode steps taken when the reward first reached 95% of the maximum reward as the learning rate in each task. The performances evaluation using three different algorithms is shown in Table 1. All the tasks with different algorithm have been performed five times, and the average value is taken as the final result. Compared with the linear reward method, the OCM reward method improves the learning rate by 257.4, 1336, 1752 episodes for task *b*, *c* and *d* respectively based on MAPPO algorithms, and the average reward increases by 3.17, 55.9 and 139 respectively. For task *a,* the OCM reward method improves the learning rate by 174.6, 133, and 301.6 episodes for MAPPO, PPO and A3C algorithm respectively, and the average reward increases by 12.05, 23.68 and 27.15 respectively. Meanwhile, the MAPPO algorithm has 178.8 episodes (137.2 episodes for linear reward method) less consumption than PPO algorithm and 344.8 episodes (471.8 episodes for linear reward method) less than A3C algorithm for OCM reward method. It indicates that the proposed reinforcement learning framework for robot manipulation skill acquisition is feasible and has better learning efficiency.

The statistical performance analysis of the six different methods for task *a* is shown in Figure 8. All the tasks with different algorithm have been performed five times. The statistical test includes the average learning rate and the average reward. As can be seen from the figure, the MAPPO algorithm with OCM reward method has the fastest learning

TABLE 1. The performances evaluation using three different algorithms.

Tasks	Methods and algorithm		Learning time (Episode)	Average reward
b	MAPPO	OCM	494.8	-6.96
		Linear	752.2	-10.13
c		OCM	3097	-139.8
		Linear	4433	-195.7
d		OCM	7374	-378.2
		Linear	9126	-517.2
a	MAPPO	OCM	88	34.99
		Linear	262.6	22.94
	PPO	OCM	266.8	8.25
		Linear	399.8	-15.43
	A3C	OCM	432.8	5.10
		Linear	734.4	-22.05

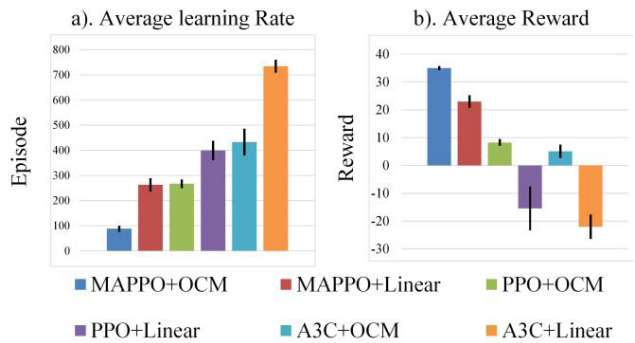


FIGURE 8. Statistical performance analysis of the six different methods for task a.

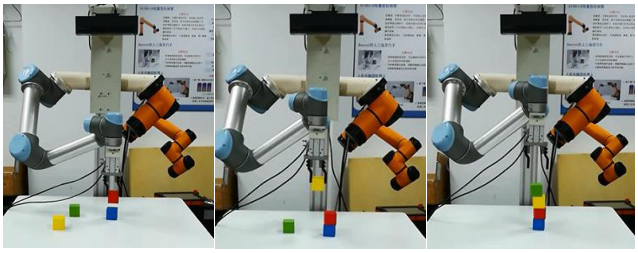


FIGURE 9. The implementation results of the test process on real robot.

rate, biggest average reward and the lowest standard deviation (12.14 for learning rate and 0.75 for average reward). For all the three RL algorithms, there are significant improvement for OCM reward method compared with linear reward method. The proposed framework is tested on the UR5 robot. The implementation results of the test process is shown in Figure 9. The results on the real robot system and the simulation platform are the same.

VI. CONCLUSION

In this paper, a robot reinforcement learning framework for acquiring manipulation skill is proposed. Aiming at the robot manipulation task, the reward function is designed based on OCM. Through integrating robot and object state with high-level action set, the MAPPO algorithm is presented for robot reinforcement learning. The *Actor* in MAPPO algorithm outputs an action and an object from the manipulation set. The learning framework can largely reduce the early exploration process of robot skill acquisition, and has a better learning efficiency. We think that the performance improvement of the framework comes from two parts. First, the OCM reward function points out the learning direction for the reinforcement learning algorithm rather than the aimless attempt. Second, the proposed *Actor* of MAPPO algorithm greatly reduces the useless attempt in the learning process. Thus, the MAPPO algorithm is suitable for other actor-critic based reinforcement learning algorithms. We verified the effectiveness and feasibility of the proposed methods according to four different manipulation tasks. The learning rate of OCM reward based learning increases 34.22% (for task *a*), 30.14% (for task *c*) and 19.20% (for task *d*) compared to linear reward based learning using MAPPO algorithm. Compared with PPO and A3C algorithms, MAPPO algorithm improves the learning rate by 33.27% and 41.07%.

In the future work, we plan to extend the framework to continuous multi-task scenarios for robot manipulation, which will further increase the universality of the method. Since all the tasks used in this paper are all independent. At the same time, robot experience will be considered in the learning framework to further increase the learning efficiency and robotic intelligence.

REFERENCES

[1] X. Broquere, D. Sidobre, and K. Nguyen, "From motion planning to trajectory control with bounded jerk for service manipulator robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, May 2010, pp. 4505–4510.

[2] M. Richter, Y. Sandamirskaya, and G. Schöner, "A robotic architecture for action selection and behavioral organization inspired by human cognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura, Portugal, Oct. 2012, pp. 2457–2464.

[3] H. Ngo, M. Luciw, A. Förster, and J. Schmidhuber, "Confidence-based progress-driven self-generated goals for skill acquisition in developmental robots," *Frontiers Psychol.*, vol. 4, p. 833, Nov. 2013.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.

[5] Y. Gao, F. X. Chen, and X. Lu, "Research on reinforcement learning technology: A review," *Acta Autom. Sinica*, vol. 30, no. 1, pp. 86–100, Jan. 2004.

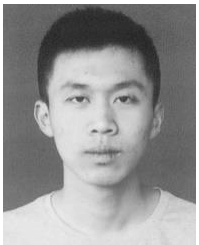
[6] X. S. Chen and Y. M. Yang, "Reinforcement learning: Survey of recent work," *Appl. Res. Comput.*, vol. 27, no. 8, pp. 2834–2844, Aug. 2010.

[7] W. Jing, C. F. Goh, M. Rajaraman, F. Gao, S. Park, Y. Liu, and K. Shimada, "A computational framework for automatic online path generation of robotic inspection tasks via coverage planning and reinforcement learning," *IEEE Access*, vol. 6, pp. 54854–54864, 2018.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[9] B. Nemec and A. Ude, "Robot skill acquisition by demonstration and explorative learning," in *New Trends in Medical and Service Robots* (Mechanisms and Machine Science), vol. 20. Cham, Switzerland: Springer, 2014, pp. 163–175.
[10] 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.
[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, Apr. 2015.
[12] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," Apr. 2017, *arXiv:1704.03073*. [Online]. Available: https://arxiv.org/abs/1704.03073
[13] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, May 2017, pp. 2786–2793.
[14] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, May 2017, pp. 3389–3396.
[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Aug. 2017, *arXiv:1707.06347*. [Online]. Available: https://arxiv.org/abs/1707.06347
[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Anaheim, CA, USA, 2016, pp. 1928–1937.
[17] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta learning shared hierarchies," presented at the 6th Int. Conf. Learn. Represent., Vancouver, BC, Canada, 2018.
[18] H. M. Le, N. Jiang, A. Agarwal, M. Dudik, Y. Yue, and H. Daume, "Hierarchical imitation and reinforcement learning," presented at the 35th Int. Conf. Mach. Learn., Stockholm, Sweden, Jul. 2018.
[19] B. Balaguer and S. Carpin, "Combining imitation and reinforcement learning to fold deformable planar objects," presented at the IEEE/RSJ Int. Conf. Intell. Robots Syst., San Francisco, CA, USA, Sep. 2011.
[20] F. M. Garcia and, P. S. Thomas, "A Meta-MDP approach to exploration for lifelong reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Feb. 2019, pp. 5692–5701.
[21] D. Dewey, "Reinforcement learning and the reward engineering principle," presented at the AAAI Spr. Symp. Ser., San Francisco, USA, Mar. 2014.
[22] E. Ratner, D. Hadfield-Menell, and A. D. Dragan, "Simplifying reward design through divide-and-conquer," Jun. 2018, *arXiv:1806.02501*. [Online]. Available: https://arxiv.org/abs/1806.02501
[23] X. Guo, S. Singh, R. Lewis, and H. Lee, "Deep learning for reward design to improve Monte Carlo tree search in ATARI games," Apr. 2016, *arXiv:1604.07095*. [Online]. Available: https://arxiv.org/abs/1604.07095
[24] J. MacGlashan, M. Babes-Vroman, M. desJardins, M. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang, "Grounding English commands to reward functions," in *Proc. Robot., Sci. Syst. XI*. Rome, Italy: Sapienza Univ. of Rome, Jul. 2015.
[25] M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions by integrating human demonstrations and preferences," Jun. 2019, *arXiv:1906.08928*. [Online]. Available: https://arxiv.org/abs/1906.08928
[26] X. Chen, A. Ghadirzadeh, J. Folkesson, M. Björkman, and P. Jensfelt, "Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 3110–3116.
[27] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proc. Conf. Robot learn.*, Zurich, Switzerland, Oct. 2018, pp. 1–23.
[28] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.
[29] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," Jul. 2017, *arXiv:1707.02286*. [Online]. Available: https://arxiv.org/abs/1707.02286

DONG LIU (Member, IEEE) received the Ph.D. degree in mechatronics engineering from the Dalian University of Technology, Dalian, China, in 2014. He was a Research Fellow in electrical and computer engineering with the National University of Singapore, Singapore, from 2015 to 1016, a Visiting Scholar in mechanical engineering with the University of British Columbia, Canada, from 2011 to 2012. He is currently an Associate Professor with the School of Mechanical Engineering, Dalian University of Technology. His research interests include intelligent robotics and system, machine learning, and cognitive control.



ZITU WANG received the B.S. degree from the College of Engineering, Nanjing Agricultural University, Nanjing, China, in 2019. He is currently pursuing the M.S. degree with the School of Mechanical Engineering, Dalian University of Technology, Dalian, China. His research interests include robot learning, cognitive control, and intelligent robotics.

BINPENG LU received the B.S. degree from the School of Mechanical and Electronic Information, China University of Geosciences, Wuhan, China, in 2016, and the M.S. degree from the School of Mechanical Engineering, Dalian University of Technology, Dalian, China. His research interests include robot learning, robot behavior planning, and intelligent robotics.

MING CONG received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 1995. Since 2003, he has been a Professor with the School of Mechanical Engineering, Dalian University of Technology, Dalian, China. He was an Outstanding Expert enjoying special government allowances approved by the State Council, an advanced worker of intelligent robot theme in the field of automation by National High Technology Research and Development Program (863), and a member of the industrial robot expert group of the fifth intelligent robot theme for the 863 program. His research interests include robotics and automation, intelligent control, and biomimetic robots.

HONGHUA YU received the B.S. degree from the School of Electromechanical and Automotive Engineering, Yantai University, Yantai, China, in 2018. He is currently pursuing the M.S. degree with the School of Mechanical Engineering, Dalian University of Technology, Dalian, China. His research interests include humanoid robot, robot learning, and artificial intelligence.

QIANG ZOU received the B.S. degree from the School of Mechanical Engineering, Liaoning Technical University, Fushun, China, in 2010. He is currently pursuing the Ph.D. degree in mechanical engineering with the Dalian University of Technology, Dalian, China. His research interests include intelligent robotics, cognitive control, and computer vision.