2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)

Deep Reinforcement Learning of Robotic Precision Insertion Skill Accelerated by Demonstrations

Xiapeng Wu, Dapeng Zhang, Fangbo Qin, De Xu, *Senior Member, IEEE*

***Abstract*—Automatic high precision assembly of millimeter sized objects is a challenging task. Traditional methods for precision assembly rely on explicit programming with real robot system, and require complex parameter-tuning work. In this paper, we realize deep reinforcement learning of precision insertion skill learning, based on prioritized dueling deep *Q*-network (DQN). The *Q*-function is represented by the long short term memory (LSTM) neural network, whose input and output are the raw 6D force-torque feedback and the *Q*-value, respectively. According to the *Q* values conditioned on the current state, the skill model selects a 6 degree-of-freedom action from the predefined action set. To accelerate the learning process, the data from demonstrations is used to pre-train the model before the DQN starts. In order to improve the insertion efficiency and safety, insertion step length is modulated based on the instant reward. Our proposed method is validated with the peg-in-hole insertion experiments on a precision assembly robot.**

I. Introduction

Precision assembly robots are widely applied in industrial automation fields, such as the assembly of microelectromechanical systems (MEMS), biotechnology and micro-optical device [1-3]. For high-precision assembly tasks with micrometers leveled precision, traditional industrial robots based on position teaching are difficult to accomplish the tasks well [4]. In the case of complex tasks, especially when the robot is in contact with the environment or the working object, the traditional robot teaching method cannot meet the assembly task requirements. Under traditional position control, small deviations between industrial robots and the environment or work objects may cause assembly failure, and even cause hazardous contact forces to cause damage to the equipment.

Reinforcement learning (RL) can endow robots with the ability to learn new tasks by actively learning without explicit teaching. The robot learns from trial-and-error, by exploring the environment and own body. RL has already been applied to a wide range of robotics problems, the work [5] has already shown the potential to learn compliance control on the peg-in-hole problem and in [6] control policies for robots can be learned directly from camera inputs in the real world. In fact, these algorithms typically require a huge amount of data by trial and error before the task had been successfully learned on real robots. An alternative approach to enabling RL on robots is learning in simulation and then transferring to the real system [7], but it required considerable engineering effort to tune the simulator to match the physical system. In many cases, it is still difficult to obtain a precise model of the physical interaction between two objects [8], the simulator of some contact-rich tasks is difficult to establish. For assembly tasks, the robot frequently interacts directly with environment without efficiency and will cause wear and tear on the robot machinery. In many real-world settings of RL, we have access to data of the system being previously operated by human controller, but we do not have access to an accurate simulator of the system. Therefore, we want the robot to learn an initial policy from the demonstration data before running on the real system. Recently, demonstration data has been shown to help in difficult exploration problems in RL [9] and allow neural network controllers to be trained efficiently [10]. Under these circumstances, RL offers some additional advantages to make the robot learn from good initial demonstrations and gradually refine it.

In this paper, we propose an insertion skill learning approach for precision assembly, based on RL and neural network. 1) We train a LSTM neural network with prioritized dueling DQN algorithm, whose input is the raw 6-axis force/torque feedback. The output *Q*-values are used to determine which action is optimal among all the predefined actions. Besides, the skill model does not rely on accurate system calibration. RL improves the robotic skill's adaptive ability, and requires few prior knowledge and little human labor. 2) To accelerate the skill learning process, the neural network model is firstly pre-trained on a few demonstrations. The pre-training reduces the risky random trials of robot. 3) To improve the safety and efficiency of insertion process, the insertion step length is modulated by the instant reward. 4) In the experiments, the effectiveness of insertion skill learning is verified on a precision assembly system.

II. Precision Assembly Robot Configuration

The robot system shown in Fig. 1 is designed to realize precision assembly. It mainly consists of a 3-DOF (degree-of-freedom) manipulator, a 4-DOF adjusting platform, three microscopic cameras, a high precision force sensor, a lighting system, and a host computer. The light system and other mechanical parts are not shown in Fig. 1.

This work is supported by Science Challenge project (No.TZ2018006-0204-02), and in part by the National Natural Science Foundation of China under Grant 61673383. Authors are with the Research Center of Precision Sensing and Control, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and School of Artificial Intelligence, University of Chinese Academy of Science, Beijing 100049, China. (wuxiapeng2017@ia.ac.cn; dapeng.zhang@ia.ac.cn)

978-1-7281-0356-3/19/$31.00

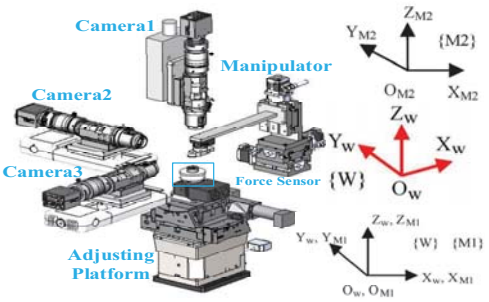


Fig. 1. Structure diagram of the precision assembly robot.

The 3-DOF manipulator with three translation DOFs can move along the X_w, Y_w and Z_w axis to align peg to hole in position. The 4-DOF adjusting platform has three rotation DOFs around X_w, Y_w and Z_w axis, respectively, and a translation DOF along the Z_w axis. The manipulator is employed for position alignment. The adjusting platform is used for attitude alignment. The sensing system includes three microscopes and a force sensor. The three microscopes are mounted approximately orthogonal to observe objects from different directions. Real-time forces and torques information from a force/torque sensor is essential to guides the contact-rich insertion. The host computer is used to control the whole assembly procedure.

The coordinates are established, as shown in Fig. 1. The world coordinates {W} and the manipulation coordinates {M1} are established on the adjusting platform. Manipulation coordinates {M2} is established on the manipulator.

III. Preliminary

Markov decision process (MDP) [11] is the foundation of reinforcement learning. MDP is defined as a tuple $\{S, A, R, P, \gamma\}$, where S represents a finite set of states, A represents a finite set of actions, $R(s,a)$ is a reward function with the state s and action a as input, P is a transition probability function, and γ is a discount factor used to calculate the cumulative reward.

For each state s, the robot selects an action a to interact with the environment. After action a is executed, the robot receives the environment's state information and reaches a new state determined by transition probability P, and the environment will give a scalar reward signal to the robot. A major aspect of RL is the ability to learn optimal decisions under various states for a given task. A common goal is to learn a strategy π that maximizes the expected sum of discounted rewards R_k:

$$R_k = r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + \ldots + \gamma^{n-k} r_n = r_k + \gamma R_{k+1} \quad (1)$$

where r_k is the instant reward at step k. The policy π maps state to action. The value $Q^{\pi}(s,a)$ of a given state-action pair $\{s,a\}$ is an estimate of the expected future reward following policy π. The function $Q^*(s,a)$ provides optimal maximum values in all states, and is determined by solving the Bellman equation:

$$Q^*(s,a) = E\left[R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q^*(s,a) \right] \quad (2)$$

The optimal policy is $\pi(s) = \arg\max_{a \in A} Q^*(s,a)$.

Deep Q-Network (DQN) [12] approximates the value function $Q(s, a)$ with a deep neural network that involves two techniques: experience replay [13] and target network. First, experience replay memory stores transitions in a replay buffer, enabling the RL agent to sample from buffer online, and store demonstration data. Second, it uses a separate target network that is updated to match the regular network after a fixed number of steps, which can stabilize the training of action values. The DQN loss is

$$J(\theta,\theta') = \left[R(s,a) + \gamma \max_{a'} Q(s',a';\theta') - Q(s,a;\theta) \right]^2 \quad (3)$$

where θ' labels the parameters of the target network. θ labels the parameters of the regular network that outputs a set of action values $Q(s, a; \theta)$ for a given state inputs. s' and a' are the next state and action, respectively.

Dueling DQN is a variant of DQN, whose architecture decomposes the Q-function into the state-value function $V^{\pi}(s)$ and the advantage function $A^{\pi}(s,a)$. $A^{\pi}(s,a)$ represents the relative advantage of action by $A^{\pi}(s,a)=Q^{\pi}(s,a)-V^{\pi}(s)$. Dueling DQN benefits from a uniform baseline for the state in the form of $V^{\pi}(s)$, and it is easier to learn relative values in the form of $A^{\pi}(s,a)$[14].The combination of the dueling DQN with prioritized experience replay is the state-of-the-art technique in discrete action settings [15]. In traditional DQN, experience transitions are uniformly sampled from the replay memory, regardless of the significance of experiences. Prioritized experience replay can sample the important transitions more frequently from its replay buffer, which improves the sampling efficiency and makes learning more efficient.

IV. Approach

Our approach is based on dueling DQN. We pre-train the neural network from the demonstration data before interacting with the assembly environment. After the pre-training, the robot starts interacting with the environment with its initial policy.

Before RL starts, the demonstration data is stored into the replay memory pool. By sampling from a mix of demonstration and trial-generated data, the robot updates its network and continues improving its performance. The following interaction takes place between the network and the robot at each time step during training. State vector containing force-torque and pose information is input to the network, whose output was used to select the optimal control action for the given input. Then, the action is executed by the robot, resulting in the motion of the objects and different contact situation. The network's output is then evaluated based on the new peg position and the contact forces. The reward is computed to evaluate the selected action by the robot. Using this evaluation, the robot adjusts its actions iteratively and the cycle is repeated.

A. Pre-training with Demonstration Data

The robot learns an initial skill model from the demonstration data with the form of RL transitions (s, a, s', r)

before running automatically. During this pre-training, the robot samples mini-batches from the demonstration data and optimizes the network. Algorithm1 shows the pseudo code of the pre-training. The pre-training loss function is a combination of the three losses: the DQN loss J, a supervised large margin classification loss J_E, and an L2 regularization loss on weights,

$$J_P = J + \lambda_1 J_E + \lambda_2 \|w\|^2 \tag{4}$$

where λ_1 and λ_2 are weighting parameters. The DQN loss ensures that the network satisfies the Bellman equation. The supervised loss forces the values of the other actions and those of the demonstrator's action have a large margin [10], namely,

$$J_E = \max_{a \in A}[Q(s,a) + l(a_D, a)] - Q(s, a_D) \tag{5}$$

where a_D is the action the demonstrator takes under state s and $l(a_D, a)$ is a margin function that is 0 when a=a_D and positive otherwise. L2 regularization loss is applied to the weights of the network to prevent over-fitting on the relatively small demonstration dataset.

Algorithm1: Pre-training with demonstration data

1: **Inputs** M_0: replay buffer with demonstration data
θ: random weights for regular network
θ': random weights for target network
τ: frequency of target network update
k: number of pre-training steps
2: **for** step $t \in \{1,2,...k\}$ **do**
3: Sample a mini-batch of n transitions from the demonstration data
4: Calculate loss J_P using target network
5: Perform a gradient descent step to update θ
6: **if** $t \,\%\, \tau$=0 then $\theta' \leftarrow \theta$ **end if**
7: **end for**

B. RL Based Insertion Skill Learning

The RL based insertion skill learning framework is shown in Fig. 2. The robot learning module is responsible for training the LSTM neural network by using RL to select the right action based on a given system state. In addition, the robot learning module also sends action command to the robot controller in order to align and insert the peg into a hole.

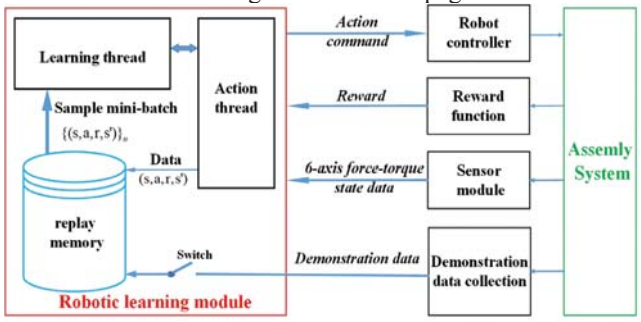


Fig. 2. The framework of insertion skill learning. The switch only turns on during pre-training stage.

As shown in Fig. 3, we use dueling DQN formed by LSTM layers to approximate the Q-function. In dueling architecture used in our implementation, LSTM outputs are converted to Q values via a fully-connected layer [16], and the LSTM layer is followed by two streams of FC layers, to estimate value function and advantage function separately and then combine the two streams to estimate action value. LSTM has a memory of the historical sensed data and is suitable to deal with the time series data having intervals and delays.

The current state s of robot is defined as:

$$s = [F_x, F_y, F_z, M_x, M_y, M_z, P_z] \tag{6}$$

where F and M are the force and moment obtained from the force-torque sensor, and their subscript x, y, z denotes the components aling X_w, Y_w and Z_w axes. The position P_z is the relative position indicating the insertion depth.

The network selects an action a with the form,

$$a = [d_x, d_y, d_z, \theta_x, \theta_y]. \tag{7}$$

where d_x, d_y are the translation of the manipulator along the axis X_w, Y_w to align peg to hole in position and peg is moved along the axis Z_w with the step length d_z in order to insert it into hole. θ_x, θ_y are the rotation angles around X_w, Y_w axis, respectively.

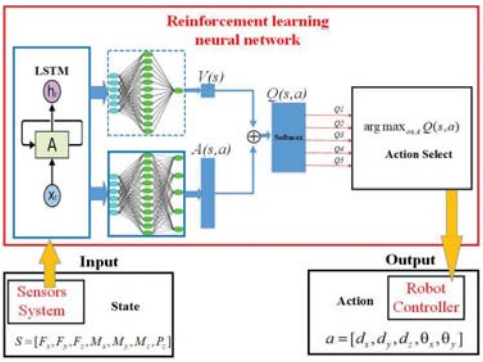


Fig. 3. Network structure using dueling DQN

In addition, we realize the modulation of insertion step length for smooth and efficient insertion. Forces along the X_w and Y_w axis are required to be less than a threshold during insertion. Furthermore, under the premise of safety, the task is also required to be completed with fewer steps for time efficiency. To satisfy this requirement, we design the K value used to modulate the reasonable step length for the current state. In practice, K value is computed by the exponential relationship K=e^r, which is related to the reward r. In sum, the modulation of insertion step length can improve the safety and efficiency of insertion. When the force along the X_w or Y_w axis exceeds the threshold, the robot gets a negative reward, so that robot should be more careful to adjust the insertion step length for safety. If the radial force is smaller, the robot gets a positive reward value means the K value is relatively large, the step length of each insertion can be relatively increased.

We design the following reward function for our experiments. In order to protect the objects and improve the assembly quality, the robot adjusts the pose of objects to

reduce the force. Therefore, in the insertion process, the radial force is used to compute the reward r,

$$r = \begin{cases} \min(1, \log(1+\beta\left|\frac{F_T}{F_R}\right|)), & F_R - F_T \le \delta \\ \max(-1, -\log(1+\beta\left|\frac{F_R}{F_T}\right|)), & F_R - F_T > \delta \end{cases} \quad (8)$$

Where F_T is the threshold value of the radial force. F_R denotes the actual radial force and is obtained by $F_R = \mathrm{sqrt}(F_x^2 + F_y^2)$, F_x and F_y are the forces along the axis X_w and Y_w. When F_R is larger than F_T, a negative reward was given. Otherwise, a positive reward was given. The logarithmic operation keeps the rewards over a reasonable scale for the neural network to learn. δ is a proximity threshold. β is a scaling parameter.

In order to make action and learning simultaneous, the proposed method uses two threads: an action thread and a learning thread.

Algorithm 2: Action thread

1: Allocate a replay memory pool M_{replay} with capacity N
2: Load the demonstration data into M_{replay}
3: **For** episode $e \in \{1, \ldots, E_m\}$ **do**
4: Copy latest network weights θ from learning thread
5: Reset the initial state s_0
6: **While** the current state s is not a termination state
7: Select an action randomly with probability ε, otherwise select $a_k = \mathrm{argmax}_a Q_\theta(s, a)$
8: Execute a_k and calculate the reward r_k, observe the next state s_{k+1}, $s \leftarrow s_{k+1}$
9: Store transition (s_k, a_k, r_k, s_{k+1}) in M_{replay}, k=k+1
10: **end while**
11: **end for**
12: Send a termination signal to the learning thread

Algorithm 2 shows the pseudo code of the action thread. The episode ends when we successfully finish the insertion phase, or a safety violation occurs, i.e., excessive force. Each step, the robot stores experience transitions in a replay memory and selects an action according to the neural network output.

Algorithm 3: Learning thread

1: Initialize the learning network with the weights of the pre-training result
2: **Repeat**
3: **if** current episode e is greater than $E_{threshold}$ **then**
4: Sample mini-batch with size B_{batch} randomly from M_{replay} with prioritization
5: Calculate the loss $J = \left[r + \gamma Q(s_{k+1}, a'; \theta') - Q(s_k, a; \theta)\right]^2$
6: Perform a gradient descent step to update θ
7: $\theta' \leftarrow \theta$
8: **end if**
9: **until** Receive a termination signal from action thread

Algorithm 3 shows the learning thread. It updates the network with a mix of demonstration and trial-generated data. In practice, we also use prioritized replay mechanism to automatically tune the ratio between demonstration and trial-generated data while sampling from replay memory,

V. Experiments and Results

The robotic assembly skill had two phases: pose alignment phase and peg-in-hole insertion phase. In the pose alignment phase, the image Jacobian based visual controller was used to align the two objects in position and attitude. In the insertion adjustment phase, the robot was required to insert the peg into the hole smoothly and effectively. The proposed method was evaluated with objects that had narrow clearance. The experiment system was set up as shown in Fig. 4.

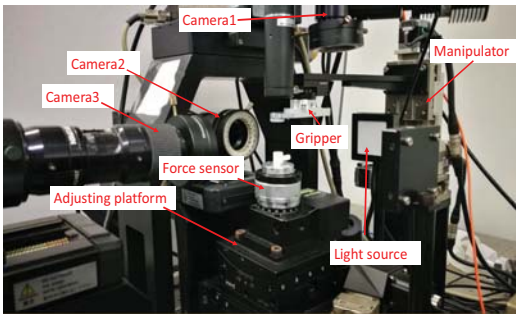


Fig. 4. Experimental system

The rotating resolution of the adjusting platform was 0.02°, and the translation resolution was 1μm. The force sensor was fixed at the end of the adjusting platform, which had a measuring range of 18N and the resolution of force was 1/128N. The forces were filtered by a Butterworth low-pass filter. The manipulator had the translation resolution of 1μm. The clearance between the peg and hole is shown in Table I.

TABLE I. Peg-Hole Dimensions

Type	Shape	Material	Height (mm)	Diameter (mm)	Clearance
Peg1	round	plastic	4.00	5.990	15μm (to hole 1)
Peg2	round	plastic	4.00	5.990	10μm (to hole 2)
Peg3	round	plastic	4.00	5.992	8μm (to hole2)
Hole1	round	plastic	3.00	6.005	-
Hole2	round	plastic	3.00	6.000	-

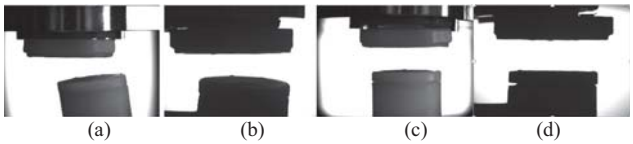
The experimental platform was controlled by a host computer with the 4.0GHz Intel Core i7 CPU and the Nvidia GTX1080 GPU.

A. Pose Alignment Phase

The pose alignment was implemented by the methods in [17]. Firstly, the attitude alignment were executed with the adjusting platform, based on the angle errors observed in images. After the attitude alignment was finished, the position of hole was adjusted by the manipulator according to the position errors observed in images. Fig. 5 showed the images of the two objects before and after pose alignment.

To demonstrate that the insertion skill model can adapt to small pose misalignment, we used a template matching

method to extract the pose features from raw image, which was fast and easy to configure, but had lower precision than the specified image feature extraction algorithm in [17]. Thus a small pose error ranging within 50μm and 1 degree still existed after the pose alignment phase.



(a) (b) (c) (d)

Fig. 5. Images before and after pose alignment in the views of camera 2 and 3. (a, b) before alignment, (c, d) after alignment.

B. Pre-training Skill Model with Demonstration Data

We collected the demonstration data via manually operating the insertion by a human demonstrator. In each demonstration step, the demonstrator selected an action from the predefined action set, according to the current state. Thus, the demonstration data format would be compatible with the following RL frame work. We firstly used this demonstration data to pre-train the neural network for insertion tasks. After demonstration, we collected 16 episodes of data in total. For pre-training, we chose Adam optimizer [18] with the epsilon $1e{-}6$ to perform gradient descent optimization.

C. RL Based Insertion Skill Learning

The RL based insertion skill learning was executed after pre-training. Peg was moved up by the adjustment platform. An episode was terminated if F_z was larger than the threshold 800mN, which meant that the vertical contact was solid and the insertion was finished. For safety, the episode was also immediately terminated in the case of excessive force or jammed in narrow clearance.

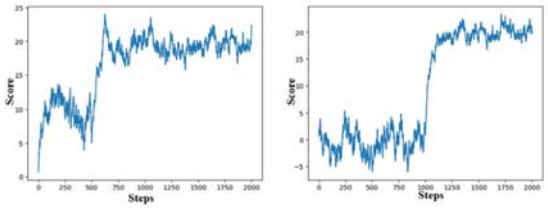
The neural networks were trained through trial and error of RL after pre-training. Note that the input $s = [F_x, F_y, F_z, M_x, M_y, M_z, P_z]$ were linearly rescaled before fed to the neural network. To execute insertion, the DQN output was chosen from the following five actions:

1)[0, 0, -d_z, 0, 0]
2)[+d_x, 0, -d_z, -θ_x, 0]
3)[-d_x, 0, -d_z, +θ_x, 0]
4)[0, +d_y, -d_z, 0, -θ_y]
5)[0, -d_y, -d_z, 0, +θ_y]

The action parameters were set as: d_x=3μm, d_y=3μm, d_z =70μm, θ_x=0.2°, θ_y=0.2°.The neural network in Fig. 3 was formed by LSTM layers with h=20 hidden units. For computing the reward in (8), we set the threshold value of radial force as F_T =100mN and δ=20mN. The parameters shown in Algorithm 2 and Algorithm 3 were set as: N=10000, E_m=200, B_{batch}=64, $E_{threshold}$=20. The initial exploration rate ε_0 for the network was set to 1.0 and progressively decay until it reached 0.1. The Adam optimizer with the epsilon $1e{-}6$ is also used in RL. The discount factor γ was set as 0.5.

1) Influence of Pre-training

As is shown in Fig.6, the y-axis indicates the *score*, i.e., the cumulative sum of reward at each step. It clear at the end of each episode. The score curve over RL training steps shows the skill learning efficiency. By comparing Fig. 6(a) and Fig. 6(b), it is found that the RL score curve with pre-training was positive and increased significantly after 500 steps. However, RL score curve without pre-training had negative values initially and increased significantly after 1000 steps. Therefore, the RL with demonstration based pre-training showed accelerated learning process. The parameters used for pre-training were set as: λ_1= 1.0, λ_2 = 10^{-4}, k=8000, τ =500, $l(a_D, a)$ = 0.8.



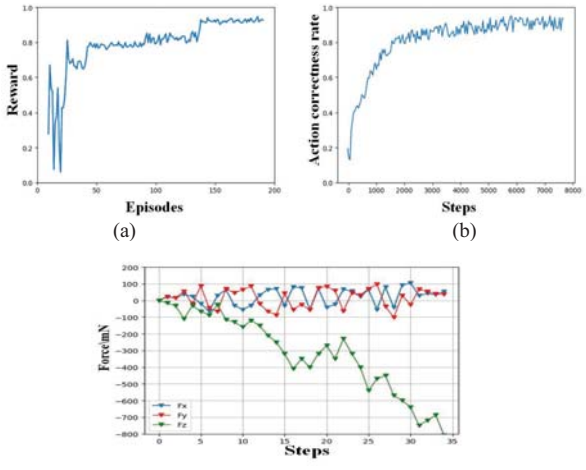
(a) (b)

Fig. 6. Score curve over RL training steps: (a) with pre-training. (b) without pre-training.

2) Insertion Skill Learning Performance

To test the performance of the proposed method, we conducted experiments with different combinations of a peg and a hole for different clearances defined in Table I.

Fig. 7 shows the learning progress in the case of a clearance of 15μm. As is shown in Fig. 7(a), the reward obtained by robot grew progressively as the insertion episode number increased. As is shown in Fig. 7(b), the *action correctness rate* is defined to evaluate the ratio of correct actions among all actions excluded those from exploration. Here the correct action means the action that provides a positive reward. Every time the robot took an action step and optimize once, the action correctness rate was calculated based on all the recorded historical actions. In the beginning of the RL learning, the actions of random exploration were more frequently executed, so the action correctness rate was low. As the RL learning steps increased, the action correctness rate became larger gradually, which meant that the robotic insertion skill was improved by selecting correct actions. The force curve was shown in Fig. 7(c), forces along the X_w and Y_w axis were limited to a small range, so that the insertion is very compliant.



(a) (b)

(c)

Fig. 7. Performance of the proposed method during learning insertion with clearance of 15μm (a) Reward, means and 90% confidence bounds in moving window of 10 episodes. (b) Action correctness rate, (c) force curve during insertion.

3) Influence of Modulation of Insertion Step Length

After the neural network training, we compared the performances with and without modulation of insertion step length, determined by K value. The network selected action from the following five actions with K value:

1)[0, 0, $-Kd_z$, 0, 0]

2)[$+d_x$, 0, $-Kd_z$, $-\theta_x$, 0]

3)[$-d_x$, 0, $-Kd_z$, $+\theta_x$, 0]

4)[0, $+d_y$, $-Kd_z$, 0, $-\theta_y$]

5)[0, $-d_y$, $-Kd_z$, 0, $+\theta_y$]

The set of comparison experiments were carried out. The insertion method in [17] was the comparative method. To evaluate the performance of the proposed method, 50 insertion trials with the peg and hole were carried out for each method. Insertion was executed under the condition that the initial pose of the peg and hole was manually tuned to the same setting, so that the pose variation did not affect the comparison result. Three group of experiments are taken with the two settings: A: initial positional error of 150 μm, orientation error of 0.5° and clearance of 10μm. B: initial positional error of 100μm, orientation error of 1.0° and clearance of 8μm. The results are given in Table II.

According to Table II, firstly, for the two different pose settings, the proposed method could reliably complete these insertion tasks, and is robust to the variation of clearance and pose alignment errors. Secondly, Table II also shows the robot can complete insertion within fewer steps by adjusting insertion step length with K value. Thirdly, the proposed method finished the insertion with a narrower clearance of 8μm, while its performance had tiny reduction. The result shows that more robust and efficient skills can be acquired by the proposed method, compared to the traditional insertion method in [17].

TABLE II. COMPARATIVE RESULTS BASED ON 50 TRIALS

Method	Setting	Average insertion steps	Average insertion time (s)
Proposed method without modulation of insertion step length	A	33	25.5
	B	36	27.3
Proposed method with modulation of insertion step length	A	**28**	**21.0**
	B	**30**	**22.8**
Method in [17]	A	54	45.9
	B	65	56.5

VI. CONCLUSION

This paper proposes a deep reinforcement learning approach for precision insertion skill, which enables the robot to learn and optimize its insertion skill without relying on explicit and complex programming. The LSTM neural network is utilized to approximate the Q-function in the prioritized dueling DQN framework. Pre-training is utilized to accelerate the skill learning. The modulation of insertion step length is realized based on the instant reward. The experiments demonstrate the feasibility of the proposed method with peg-in-hole tasks.

REFERENCES

[1] E. Avci, K. Ohara, C. N. Nguyen, et al., "High-speed automated manipulation of microobjects using a two-fingered microhand," *IEEE Trans. Ind. Eletron.*, vol. 62, no. 2, pp. 1070-1079, 2015.

[2] F. Qin, F. Shen, D. Zhang, et al., "Contour primitives of interest extraction method for microscopic images and its application on pose measurement," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 48, no. 8, 2018.

[3] F. Qin, D. Zhang, D. Xu, et al., "Robotic skill learning for precision assembly with microscopic vision and force feedback," *IEEE Transactions on Mechatronics*, DOI: 10.1109/TMECH.2019.2909081, 2019. (IF:3.9, SCI)

[4] J. Zhang, D. Xu, Z. T. Zhang, and W. S. Zhang, "Position/force hybrid control system for high precision alignment of small gripper to ring object," *Int. J. Autom. Comput.*, vol.10, no.4, pp.360–367, Aug.2013.

[5] V. Gullapalli, A. G Barto, et al., "Learning admittance mappings for force guided assembly," in *IEEE International Conference on Robotics and Automation*, 1994, pp. 2633–2638.

[6] Levine, Sergey, et al., "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, " *The International Journal of Robotics Research*, 37(4-5), 421-436.2018.

[7] J. Matas, S. James, et al., "Sim-to-Real Reinforcement Learning for Deformable Object Manipulation," in *Conference on Robot Learning, pp.* 734-743, 2018.

[8] C. Bouchard, M. Nesme, et al., "6D frictional contact for rigid bodies," in *Graphics Interface Conference. Canadian Information Processing Society*, pp.105–114,2015

[9] Subramanian, K., Isbell Jr, et al., "Exploration from demonstration for interactive reinforcement learning," in *International Conference on Autonomous Agents & Multiagent Systems,* pp.447–456,2016

[10] Hester T, *et al*. "Deep Q-learning from demonstrations," *Conference on Artificial Intelligence*, 2018.

[11] R. S. Sutton, A. G. Barto, "Introduction to reinforcement learning," *Cambridge: MIT Press,* vol.10,1998.

[12] Mnih V, Kavukcuoglu K, Silver D, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529, 2015.

[13] L. J. Li, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning,* vol. 8.3, no. 4 293-321, 1992

[14] Z. Wang, N. D. Freitas, et al., "Dueling Network Architectures for Deep Reinforcement Learning," in *International Conference on Machine Learning, pp.* 1995-2003, 2016.

[15] T. Schaul, J. Quan, et al., "Prioritized Experience Replay," *In International Conference on Learning Representations,* 2016.

[16] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.

[17] S. Liu, D. Xu, D. Zhang, and Z. Zhang, "High precision automatic assembly based on microscopic vision and force information," *IEEE Trans. Autom. Sci. Eng.,* vol. 13, no. 1, pp. 382–393, Jan. 2016.

[18] Kingma D P, Ba J., "Adam: A Method for Stochastic Optimization," *Computer Science*, 2014.