International Journal of Automation and Computing
www.ijac.net

18(3), June 2021, 457-467
DOI: 10.1007/s11633-021-1290-3

Skill Learning for Robotic Insertion Based on One-shot Demonstration and Reinforcement Learning

Ying Li[1,2] De Xu[1,2]

[1]Research Center of Precision Sensing and Control, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract: In this paper, an efficient skill learning framework is proposed for robotic insertion, based on one-shot demonstration and reinforcement learning. First, the robot action is composed of two parts: expert action and refinement action. A force Jacobian matrix is calibrated with only one demonstration, based on which stable and safe expert action can be generated. The deep deterministic policy gradients (DDPG) method is employed to learn the refinement action, which aims to improve the assembly efficiency. Second, an episode-step exploration strategy is developed, which uses the expert action as a benchmark and adjusts the exploration intensity dynamically. A safety-efficiency reward function is designed for the compliant insertion. Third, to improve the adaptability with different components, a skill saving and selection mechanism is proposed. Several typical components are used to train the skill models. And the trained models and force Jacobian matrices are saved in a skill pool. Given a new component, the most appropriate model is selected from the skill pool according to the force Jacobian matrix and directly used to accomplish insertion tasks. Fourth, a simulation environment is established under the guidance of the force Jacobian matrix, which avoids tedious training process on real robotic systems. Simulation and experiments are conducted to validate the effectiveness of the proposed methods.

Keywords: Force Jacobian matrix, one-shot demonstration, dynamic exploration strategy, insertion skill learning, reinforcement learning.

Citation: Y. Li, D. Xu. Skill learning for robotic insertion based on one-shot demonstration and reinforcement learning. *International Journal of Automation and Computing*, vol.18, no.3, pp.457–467, 2021. http://doi.org/10.1007/s11633-021-1290-3

1 Introduction

Recently, precision assembly and manipulation have attracted much attention, and been widely used in micro-electromechanical systems (MEMS) and industrial applications[1–4]. The contact force between components, provided with the force sensor, should be kept within a limited range to guarantee the safety.

Peg in hole assembly is a common assembly task and automatic assembly methods have attracted much attention. Generally, the assembly control methods can be classified into model-based methods and model-free methods. The former have been widely used in assembly tasks. In order to accomplish dual peg in hole assembly, Zhang et al.[5] analyzed the contact states and established the relationship between contact state and contact force. The jamming state was analyzed quantitatively and the corresponding control strategies were developed. In order to assemble three objects together, Liu et al.[6] modelled the contact states between each two components as a probability distribution. The three objects were adjusted simultaneously. Chen et al.[7] developed an error recovery method for wiring harness assembly. The dynamic model of mating connectors on printed circuit board (PCB) is established and the smooth insertion is achieved with moment control. The components in the above methods are rigid. If the components are deformable, the insertion tasks will become more difficult and the contact states will be more complex. Xing et al.[8] presented an efficient assembly method for multiple components connected parallel by spring. An optimization method was developed based on the spring model. Efficiency is an important factor and a passive alignment principle-based method was employed to accomplish assembly tasks with deformable components[9].

However, the above methods are mainly based on mathematical description of contact states which may contain errors. The real contact states are far more complex and the precise model can hardly be obtained. Therefore, model-free precision insertion methods are highly needed.

Recently, reinforcement learning (RL), combined with deep learning, has shown its great potential in the field of artificial intelligence[10] and continuous control has been

Research Article
Manuscript received October 9, 2020; accepted March 2, 2021; published online March 24, 2021
Recommended by Associate Editor Nazim Mir-Nasiri
Colored figures are available in the online version at https://link.springer.com/journal/11633

realized in simulated physics tasks[11]. RL becomes a promising approach for robotic precision assembly[12, 13]. Inoue et al.[14] proposed a Q-learning-based method for assembly with robot arm, and long short term memory (LSTM) layers were used to approximate the Q-function. Li et al.[15] presented a robot acquisition method for assembly process. Unlike others, the reward function employed a two-classification support vector machine (SVM) model to determine whether the assembly is successful. However, the actions in the above methods are discrete and continuous actions are more suitable for the assembly process. Fan et al.[16] presented a learning framework for high precision industrial assembly, which combined the supervised learning and deep deterministic policy gradient (DDPG) based RL. Trajectory optimization served as a semi supervisor to provide initial guidance for the actor-critic. In order to improve training efficiency, Vecerik et al.[17] developed a DDPG-based insertion method, which introduced the human demonstrations into the learning process. Guided by the behavior cloning loss, the actor network can imitate the actions from demonstrations. As for deformable objects, Luo et al.[18] proposed a mirror descent guided policy search (MDGPS) method to insert a rigid peg into a deformable hole. Moreover, available priori knowledge can further improve the performance of RL. For example, Thomas et al.[19] developed a computer aided design (CAD) based RL method for robotic assembly. CAD data can be used to guide the RL by a geometric motion plan.

The above RL-based methods can be used in real robotic assembly tasks. However, there are still some problems should to be solved. Firstly, expert demonstrations can improve the training efficiency[17], but it is tedious and not safe to collect much demonstration data on real robotic systems. It is valuable to use as little demonstration data as possible to guide the training process of RL. Secondly, in order to obtain the optimal action policy, abundant exploration is needed. An efficient exploration strategy can accelerate the training process and random exploration in action space is not enough[14]. Therefore, an efficient exploration strategy for robotic assembly is highly needed. Thirdly, the model is usually trained for specific components[19] and should be retrained when meeting new components in real assembly tasks. The adaptability is an important factor and should be improved to meet the requirements of the real assembly tasks. Furthermore, training RL model on real robotic system is time-consuming and low-efficiency. There is a gap between simulation and real robotic systems. In other words, the models trained in simulation environments cannot be directly used on real robotic systems[20]. The training efficiency can be improved if the gap is bridged.

In this paper, a DDPG-based insertion skill learning framework is proposed for robotic assembly. The main contributions of this work are as follows. 1) The final executed action consists of an expert action learned from one demonstration and a refinement action learned from RL, to improve the insertion efficiency. 2) An episode-step based exploration strategy is proposed to explore state space more efficiently, which views the expert action as a benchmark and adjusts the exploration intensity dynamically. 3) A skill saving and selection mechanism is proposed to improve the adaptability of our method. Trained models for several typical components are saved in the skill pool and the most appropriate model will be selected for insertion tasks for a new component. 4) A simulation environment is established with the help of force Jacobian matrix, which avoids tedious training process on real robotic system.

The rest of this paper is organized as follows. Section 2 introduces the system configuration and problem formation. The insertion skill learning framework is detailed in Section 3. Sections 4 and 5 present the simulation and experiment results, respectively. Finally, this paper is concluded in Section 6.

2 System configuration and problem formulation

2.1 System configuration

The automated precision assembly system is designed as shown in Fig. 1. It consists of a 4 degree of freedom (DOF) adjustment platform, a 3-DOF manipulator, three microscopic cameras, lighting system and a host computer. The optical axes of three microscopic cameras are approximately orthogonal to each other and the three microscopic cameras can move along their moving platform to adjust the distance between objective lens and objects for capturing clear images. The microscopic camera 1–3 provide mid-view, side-view and up-view of objects.

The world coordinate {W} is established on the base of manipulator. The manipulator can move along X_w, Y_w and Z_w axes. The platform coordinate {P} is established

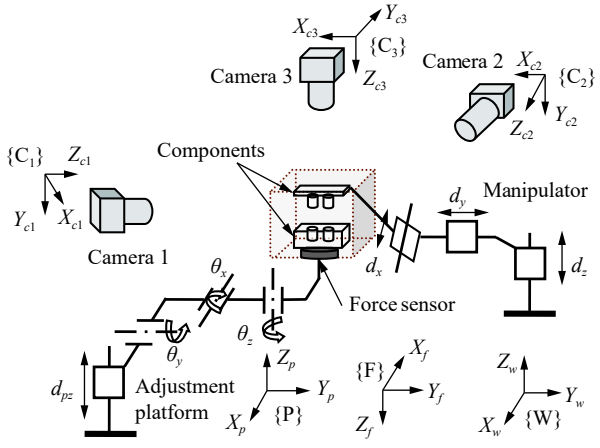

Fig. 1 System configuration and task description

on the adjustment platform. The 4-DOF adjusting platform consists of three rotation DOFs around X_p, Y_p and Z_p axes, respectively, and a translation DOF along Z_p axis. The camera coordinates $\{C_1\}$, $\{C_2\}$ and $\{C_3\}$ are established on the three cameras, respectively. The force coordinate $\{F\}$ is established on the force sensor.

2.2 Problem formulation

For precision assembly, the goal is to learn the insertion policy through interacting with environment. The insertion process can be modeled as a Markov decision process (MDP). At each time step t, the agent observes a state $s_t \in S$, executes an appropriate action $a_t \in A$, and receives a reward R_{Mt}. Then the state is transferred to $s_{t+1} \in S$. The desired insertion policy μ, mapping from states to actions, is obtained by maximizing the sum of expected discounted rewards.

In the insertion task, the state s is defined as

$$s_t = [f_x, f_y, f_z, p_z]^{\mathrm{T}} \tag{1}$$

where f_x, f_y, and f_z are the contact forces along X_f, Y_f, and Z_f axes, respectively; p_z is the insertion depth along Z_w axis. The action is defined as

$$a_t = [d_x, d_y, d_z]^{\mathrm{T}} \tag{2}$$

where d_x and d_y are the compliant adjustments along X_w and Y_w axes, respectively; and d_z is the insertion step along Z_w axis.

3 Insertion skill leaning

The insertion skill learning framework is based on DDPG, whose framework is shown in Fig. 2. The training process is divided into two stages: expert action learning and self-learning. The final action is composed of two parts: an expert action and a refinement action, which are obtained in the two stages, respectively. In the expert action learning stage, only one expert demonstration is collected and the expert action is learned. In the self-learning stage, the actor and critic networks are further trained with RL through interacting with environment. Besides, efficient exploration strategy is developed to accelerate the training process. After training, the agent obtains the insertion skill and accomplishes the insertion tasks. A skill saving and selection mechanism is designed to improve the adaptability of insertion tasks with different components.

3.1 Learning framework

In RL training process, random actions might be harmful to the safety of the robotic system. It is beneficial to learn a stable and safe insertion skill from expert

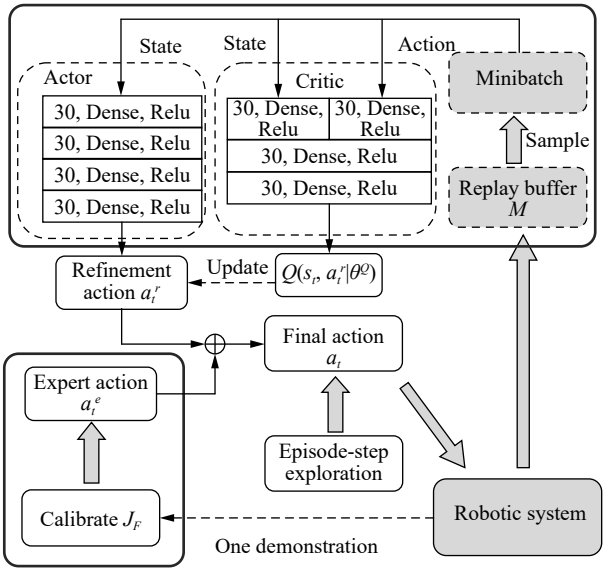

Fig. 2 Framework of insertion skill learning

demonstrations. Therefore, a novel framework is proposed to leverage demonstrations and acquire better efficiency.

1) Learning expert action from one-shot demonstration.

A common method to accelerate the RL training process is to pretrain the networks with demonstrations from experts. A large number of demonstrations are usually needed in the pretrain state to obtain adequate performance. However, data collection on real robotic systems is tedious and time-consuming. Therefore, a novel method is proposed to learn a stable and safe expert action from only one demonstration which records the states and the corresponding actions.

In the insertion process, the relationship between contact force and relative translation movements can be modeled with a force Jacobian matrix $J_F \in \mathbf{R}^{2\times 2}$:

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = J_F \begin{bmatrix} f_x \\ f_y \end{bmatrix}. \tag{3}$$

J_F can be calibrated with the least square method according to the demonstration. The expert action is represented as

$$a_t^e = [d_{ex}, d_{ey}, d_{ez}]^{\mathrm{T}} \tag{4}$$

where d_{ex} and d_{ey} are the adjustments along X_w and Y_w axes, respectively; and d_{ez} is the insertion step along Z_w axis. d_{ex} and d_{ey} can be calculated by (5).

$$\begin{bmatrix} d_{ex} \\ d_{ey} \end{bmatrix} = -\alpha J_F \begin{bmatrix} f_x \\ f_y \end{bmatrix} \tag{5}$$

where $\alpha \in [0, 1]$ is a constant. In practice, d_{ez} can be set as

a small constant value for convenience.

Then an expert action $a_t{}^e$ is gotten with only one demonstration. The demonstration data is mainly used to obtain the properties of components by calibrating J_F. It is not important whether the demonstration is optimal or not. Therefore, our method is more efficient and convenient than the traditional pretrain-based methods.

2) Neural networks based refinement action

The proposed framework contains two main networks: an actor network and a critic network. The actor network takes a state as input and outputs the refinement action $\mu(s_t \mid \theta^\mu)$ with parameter θ^μ. There are five fully connected layers in the actor network. The ReLU activation function is used in the first four layers and tanh activation function is used in the last output layer, whose output is the refinement action $a_t{}^r$. The final output action a_t is combined by the expert action $a_t{}^e$ and refinement action $a_t{}^r$,

$$a_t = a_t{}^e + a_t{}^r. \tag{6}$$

The actions along X_w and Y_w axes of final action a_t are normalized within [−1, 1] and action along Z_w axis of a_t are normalized within [0, 1]. The expert action $a_t{}^e$ is explainable, safe but not optimal. The refinement action works to improve the insertion efficiency. Then the final action a_t can meet the requirement of safety and high efficiency.

The critic network takes the state and the refinement action $a_t{}^r$ as input and outputs the action value $Q(s_t, a_t{}^r \mid \theta^Q)$ with parameter θ^Q. Two fully connected layers are employed to fuse the state and the action. And two other fully connected layers are used to approximates action value.

A target actor network $\mu'(s_t \mid \theta^{\mu'})$ with parameter $\theta^{\mu'}$ and a target critic network $Q'(s_t, a_t{}^r \mid \theta^{Q'})$ with parameter $\theta^{Q'}$ are employed to calculate the target values. Their structures are the same as the actor and critic networks, respectively.

3) Preliminaries

During training, the agent samples a minibatch of N state transitions from the replay buffer M to update the parameters of actor and critic networks.

The critic network is trained by minimizing the loss L with parameter θ^Q:

$$L\left(\theta^Q\right) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2 \tag{7}$$

where y_i is computed by

$$y_i = R_{Mi} + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}\right)|\theta^Q\right). \tag{8}$$

R_{Mi} is the reward which is detailed in Section 3.2; γ is a discount factor.

The action network is trained by maximizing $J(\theta^\mu)$ with parameter θ^μ:

$$J\left(\theta^\mu\right) = E\left[Q\left(s_t, \mu\left(s_t|\theta^\mu\right)\right)\right]. \tag{9}$$

The parameters of the actor network are updated by computing the policy gradient with the chain rule:

$$\nabla_{\theta^\mu} J\left(\theta^\mu\right) = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix}\nabla_a Q\left(s, a|\theta^Q\right)|_{s=s_i, a=\mu(s_i)}\times \\ \nabla_{\theta^\mu}\mu\left(s|\theta^\mu\right)|_{s=s_i}\end{bmatrix}. \tag{10}$$

The parameters of target networks are updated by slowly tracking the learned networks:

$$\begin{cases}\theta^{Q'} = \tau\theta^Q + (1-\tau)\,\theta^{Q'} \\ \theta^{\mu'} = \tau\theta^\mu + (1-\tau)\,\theta^{\mu'}\end{cases} \tag{11}$$

where τ is a factor between 0 and 1.

4) Self-learning stage

During self-learning, the state transitions are collected by interacting with the environment and stored in the memory replay buffer M. The pseudo code of the self-learning stage is given Algorithm 1.

The actor network is trained with $\nabla_{\theta^\mu} J\left(\theta^\mu\right)$ and the critic network is trained with loss $L(\theta^Q)$ as given in (9) and (7) with batch-size N. The dynamic exploration strategy used in the training process is given in Section. 3.3.

3.2 Safety-efficiency reward

For precision assembly, safety is important and the contact force should be kept within a safe range. Besides, the efficiency is another key factor and it is expected to finish the process with as few insertion steps as possible. Therefore, the designed reward function consists of two parts: the safety reward R_{1t} and the efficiency reward R_{2t} as given in (12).

$$\begin{cases}R_{1t} = 1 - \dfrac{f_{rt}}{f_T} \\ R_{2t} = -\left|d_{zt} - R_{1(t-1)}D_T\right|/D_T\end{cases} \tag{12}$$

where f_T is the maximum allowed radial contact force; D_T is the maximum allowed insertion depth; f_{rt} is the radial contact force after executing the t-th action.

$$f_{rt} = \sqrt{f_{xt}^2 + f_{yt}^2}. \tag{13}$$

Then the reward function R_{Mt} is calculated by

$$R_{Mt} = R_{1t} + R_{2t}. \tag{14}$$

The reward R_{1t} means that the agent will receive a small reward if the contact force is large. $f_{r(t\text{-}1)}$ can be

viewed as the contact force before executing the t-th action. The term $R_{1(t\text{-}1)}D_T$ provides an expected insertion depth. Larger the current contact force is, smaller the insertion depth should be. And the reward R_{2t} indicates that larger the difference between the real and expected insertion depths is, smaller the reward will be.

Algorithm 1. Self-learning with dynamic exploration

Initialize $\sigma_a \leftarrow 0.1$
Initialize replay buffer M
For episode =1, 2, ⋯, **do**:
  Reset the initial state s_0
  For t=1, 2, ⋯, **do**:
    Compute the expert action $a_t{}^e$ and refinement action $a_t{}^r$
    Compute the actions a_t and $a_t{}^c$ with (6) and (15)
    Execute action $a_t{}^c$, observe reward R_{Mt} and the next state s_{t+1}
    If s_{t+1} is a termination state **do**:
      break
    End if
    Store transition (s_t, $a_t{}^r$, R_{Mt}, s_{t+1}) in M
    Sample a random minibatch of N transitions from M
    Calculate gradients and update parameters
    Update σ_a with (17)
    $s_t \leftarrow s_{t+1}$
  End for
  Calculate the cumulative reward and update σ_a with (16)
End for

3.3 Dynamic exploration strategy

When training the RL model, the state space should be explored to improve the performance of the action policy. However, random exploration might be harmful to the safety of the robotic system. For example, the radial contact force may exceed the allowed range. An appropriate exploration strategy can encourage the agent to explore the state space more efficiently. Therefore, we develop an episode-step exploration strategy and the exploration intensity is adjusted online according to the current performance of agent.

Gaussian noise is added to the action for random exploration.

$$a_t^c = a_t + N\left(0, \sigma_a I\right) \tag{15}$$

where σ_a is the standard deviation; $a_t{}^c$ is the output action with Gaussian noise.

The parameter σ_a determines the exploration intensity. Generally, the exploration should be increased when the performance of action policy is unsatisfactory. The average episode reward can indicate the performance of action policy. Then a simple but effective episode-based exploration method is given as

$$\sigma_a = \begin{cases} \sigma_{t1}, \text{ if } \dfrac{1}{N_s}\displaystyle\sum_{t=0}^{N_s} R_{Mt} < 0 \\ \sigma_{t2}, \text{ otherwise} \end{cases} \tag{16}$$

where N_s is the number of steps in the episode; σ_{t1} and σ_{t2} are two thresholds where $\sigma_{t1} > \sigma_{t2}$; σ_a is adjusted after each episode.

The episode-based method is insufficient because σ_a is only updated after one episode finishes, which is delayed. Then a step-based exploration method is developed which works as a supplement to the episode-based method.

In general, the performance of action $a_t{}^c$ is expected better than which of the sole expert action $a_t{}^e$. Therefore, the expert action $a_t{}^e$ can be used as an appropriate benchmark to evaluate the performance of the agent after each step. Specifically, if the performance of $a_t{}^c$ is better than that of $a_t{}^e$, the exploration should be decreased for generating stable output. On the contrary, the exploration should be increased for generating a better policy. There is another problem that the real executed action is $a_t{}^c$ rather than $a_t{}^e$, which means the reward R_e generated by $a_t{}^e$ cannot be obtained directly. The reward R_e can be estimated with the state before executing action $a_t{}^c$. The efficiency part R_{2t} can be calculated by (12). The safety part R_{1t} is calculated with the contact force before rather than after executing $a_t{}^c$. Generally, the radial contact force will decrease after executing $a_t{}^e$, which means the reward R_e calculated by the above method is worse than the real ones. Therefore, the reward R_e is competent to work as the benchmark to guide exploration. Then the step exploration method is given as

$$\sigma_a \leftarrow \sigma_a - \sigma_b \tanh\left(R_{Mt} - R_e\right) \tag{17}$$

where σ_b is a constant. And σ_a will be limited within $[\sigma_{\min}, \sigma_{\max}]$.

3.4 Skill saving and selection mechanism

In real robotic assembly tasks, the properties of components are different. The model trained with one kind of component might not be suitable for other components. And it is tedious to train a new model for each new component. In order to solve this problem, an insertion skill saving and selection mechanism is developed and the flow chart is given in Fig. 3. Firstly, several typical components are selected and used to train the corresponding models with the proposed methods. Then the trained parameters are saved in a skill pool S_p. The force Jacobian matrix of the i-th model in S_p is denoted as J_{Fi}.

Given a new component, one demonstration should be firstly conducted and the force Jacobian matrix J_{Fnew} is calibrated. The distance D_i between J_{Fnew} and J_{Fi} is computed by

$$D_i = \left\| J_{Fi} - J_{Fnew} \right\|_F^2 . \tag{18}$$

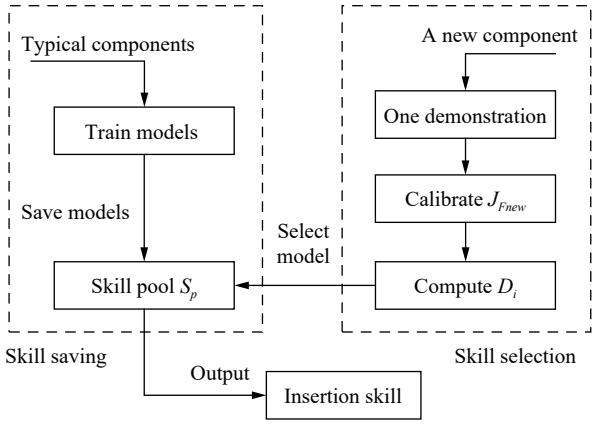

Fig. 3 Flow chart of insertion skill saving and selection mechanism

The model with the minimal distance is chosen as the appropriate model. And the corresponding insertion skill is restored from the skill pool and employed to guide the insertion task with the new component. Therefore, the adaptability for different components is improved.

4 Simulation

This section demonstrates the feasibility of the proposed insertion skill learning method in a peg-in-hole assembly simulation environment.

4.1 Simulation setup

The simulation environment used in this experiment is the same as [21]. The friction coefficient is set to 0.3. The Hookean coefficient is set to 3.3 mN/μm. There are two cylindrical components to be assembled. The heights of the two components are 4 mm. The diameters of the peg and the hole are 4 mm and 4.01 mm respectively.

The training parameters of the proposed insertion skill learning method is given in Table 1. To guarantee the safety of insertion task, the insertion step length d_z is limited within [0, 150 μm] and the adjustments d_x and d_y are limited within [−5 μm, 5 μm]. The maximum allowed radial contact force f_T is set to 80 mN. The maximum insertion steps are set to 1 000. If radial contact force f_r exceeds f_T, the insertion task fails. The insertion task succeeds when $|f_z| > 1\,000$ mN.

The initial orientation and position errors are set within 0.3 degree and 10 μm, respectively. During self-learning, initial states are sampled randomly within the pose errors range. The test dataset includes 100 initial states with random pose errors. And the trained model is evaluated in the test dataset.

Table 1 Training parameters

Parameters	Values	Parameters	Values
Delayed update rate τ	0.1	Discount factor γ	0.99
Learning rate	0.001	Self-learning episodes	200
Batch size N	32	Size of M	200
Constant α	0.15	Thresholds σ_{t1} and σ_{t2}	0.3, 0.1
Thresholds $\sigma_{\min}$, $\sigma_{\max}$	0.1, 0.5	Constant σ_b	0.3

4.2 Expert action learning results

Firstly, one insertion demonstration is conducted. The expert action is learned with the method detailed in Section. 3.1. and evaluated in the test dataset. The success rate is 100%. The mean reward is 0.52. The distribution of radial contact force is shown in Fig. 4. The number of insertion steps is about 54. And the contact force descends below 10 mN after about 10 steps since the beginning of task.

It can be seen the expert action can meet the requirement of safety, but the efficiency is low. Therefore, the performance of the agent should be further improved with the self-learning method introduced in Section. 3.1.

4.3 Self-learning results

To validate the effectiveness of the proposed dynamic exploration strategy, three different strategies are compared: episode-step exploration, episode exploration and step exploration. The initial insertion depths are set randomly. The self-learning stage terminates after 200 episodes. The training results are shown in Fig. 5. The curve converges after about 25 episodes with the episode-step exploration strategy. In contrast, the curve converges after about 65 and 45 episodes with episode exploration and step exploration, respectively. Compared with episode exploration method, the step exploration method can adjust the exploration intensity in a more timely way. And the convergence speed of step exploration is faster than that of episode exploration. Therefore, the proposed episode-step exploration strategy can improve the training efficiency.

After self-learning, the performance of the agent is evaluated in the test dataset. The mean reward is 0.91 and the success rate is 100%. The number of insertion steps is about 31. The contact force descends below 10 mN

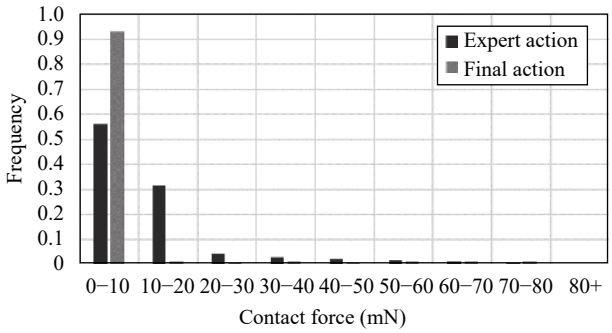

Fig. 4 Distribution of radial contact force

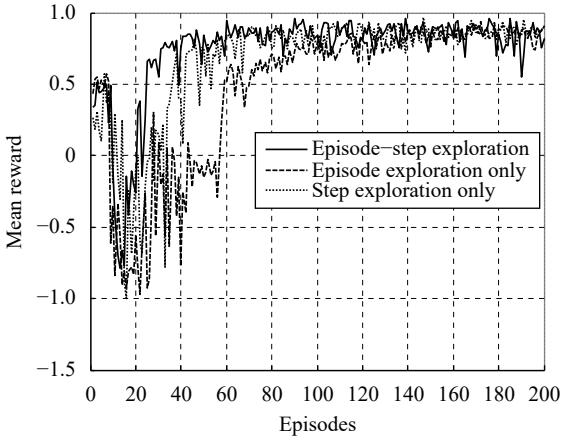

Fig. 5 Reward curves of three different exploration strategies

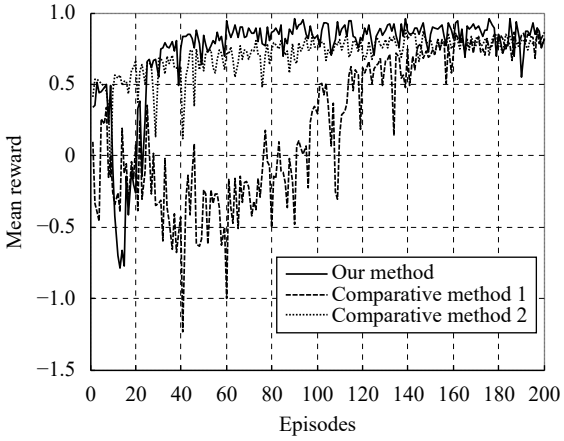

Fig. 6 Reward curves of different methods during training process

after about 3 steps since the beginning of task, which is more efficient than the results of expert action.

The distribution of radial contact force is shown in Fig. 4. The force less than 10 mN occupies over 92% with action a_t after self-learning. In contrast, it occupies only 55% when using expert action $a_t{}^e$. Compared with the expert learning results, the contact force can be kept within a smaller range after self-learning and the performance of the agent has been improved.

4.4 Comparative simulation results

The classic DDPG method[11], denoted as comparative method 1, is chosen as a comparative method, and episode exploration strategy is adopted. The structure of the network is the same as which of our method except the expert action part. The robotic assembly method in [21], denoted as comparative method 2, is chosen as another comparative method. In order to compare its performance with our method's, the fuzzy reward system is replaced with our reward function. And the other parts are the same as which of the original method in [21].

The training process finishes after 200 episodes. The training result is shown in Fig. 6. The convergence speed of comparative method 2 is much faster than which of comparative method 1. But the curve of method 2 is always below which of our method. Then the trained models are evaluated in the test dataset. The final performance of comparative methods 1 and 2 are similar. The success rates are both 100%. The mean rewards are 0.89 and 0.86, respectively. The mean reward of our method is 0.91, given in Section 4.3, which is better than the two comparative methods'. Some comparative results are given in Table 2. The average radial contact force (ACF) is computed in each insertion task. And the mean and standard deviation (STD) of ACFs are computed. The mean and STD of steps are also computed. The four values of our method are smaller than the two comparative methods.

Our method outperforms the two comparative methods and the reasons are given as follows. As for the comparative method 1, the exploration strategy is worse than ours. Besides, our framework contains an expert action and a refinement action, which can accelerate the training process. As for the comparative method 2, the acquisition of expert action is improper. It views the contact force as decoupled, which is not always the truth. And the variance of action space noise decreases all the time. On the contrary, our method can adjust the exploration intensity more flexibly. Furthermore, the two comparative methods should retrain the models when meeting new components. However, in our method, a skill pool is established with the force Jacobian matrix, and the most appropriate model will be selected from the skill pool to directly accomplish the new insertion tasks. And the gap between simulation and real robotic system can be bridged with the method detailed in Section 5.2, which is based on the force Jacobian matrix. But the two comparative methods cannot do that. Therefore, the adaptability of our method is much better.

Table 2 Contact forces and steps of different methods

Methods	Mean of ACF (mN)	STD of ACF(mN)	Mean of Steps	STD of Steps
Our method	5.53	1.26	30.21	0.43
Comparative method 1	6.77	1.34	32.58	0.59
Comparative method 2	10.13	1.66	34.32	0.65

4.5 Insertion experiments using different components

In real robotic applications, the properties of components may be different and it is time-consuming to train a new model for each new component. The method, detailed in Section. 3.4, can solve this problem. In order to validate the performance of the method, a set of experi-

ments are conducted. Three typical components are chosen and the Hookean coefficient are $0.5k_0$, k_0 and $2k_0$, respectively. k_0 is the Hookean coefficient used in the aforementioned experiments. These models are trained with the method introduced in Section 3 and are saved to the skill pool S_p.

A new component is chosen as a test component and the Hookean coefficient is $2.2k_0$. The distance D_i, i=1, 2, 3 is computed with (18). And D_3 is the smallest distance and the third model is chosen as the most appropriate model. Then the model is evaluated in the test dataset. The mean reward is 0.924 and the success rate is 100%. The results are given in Table 3. The models 1–3 are the three models in the skill pool. We test the two other models with the new component and the mean rewards decrease obviously. It validates the correctness of choosing the third model for insertion tasks with the new component. Therefore, it is important to choose an appropriate model to obtain better performance. And the proposed method provides a convenient and efficient approach for insertion tasks with a new component.

5 Real robot experiments

5.1 Experiment Setup

An experiment system is established according to the scheme given in Section 2.1, as shown in Fig. 7. In this experiment system, camera 1 and camera 2 are GC2450 cameras and camera 3 is a PointGrey camera. All the three cameras are equipped with Navitar zoom lens with magnification 0.47~4.5×, which capture images 15 frames per second with image size of 2 448×2 050 in pixel. The adjustment platform is composed of a Micos WT-100 for rotation around X_p and Y_p axes, Sigma SGSP-40YAW for around Z_p axis Micos ES-100 for translation along Z_p axis. The rotation and translation resolutions of adjustment platform are 0.001 degree, 0.02 degree, 0.02 degree. and 1 μm, respectively. The manipulator is composed of a Sugura KWG06030G for translation along X_w, Y_w and Z_w axes with resolution 1 μm.

Table 3 Success rate and reward with new component

Models	Success rate	Mean reward
Model 1	0.98	0.75
Model 2	1.0	0.84
Model 3	1.0	0.92

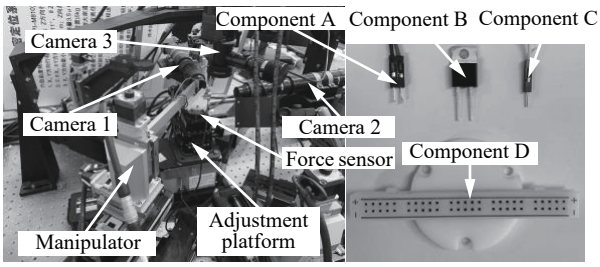

Fig. 7 Experimental system and components

Three kinds of components are employed to verify the effectiveness of the proposed insertion skill learning method. The insertion tasks are to insert the components A, B and C into the component D. The components A, B and C are electronic components. The component D is a bread board and there are many holes on it. The components A, B, and C are separately mounted on the manipulator in sequence, and component D is mounted on the adjustment platform. The diameters of the pegs are 1 mm. The height of the components A, B, and C are 5 mm, 8 mm and 5 mm, respectively. The vision-based pose alignment is conducted before the insertion process[1]. The force sensor provides contact force during insertion tasks.

5.2 Experiment results

Usually, it takes many hours to train the RL model to obtain insertion skills on real robotic systems. It is very time-consuming and the safety cannot be guaranteed. In order to solve this problem, we proposed a method to bridge the gap between simulation and the real robotic system. And the insertion skills obtained in the simulation environment can be directly used on real robotic systems by using our method.

In real robotic systems, the coordinates of force sensor and manipulator may not be identical. J_X is the inverse of J_F. and can indicate the relationship between relative movements and contact force. Then a new simulation environment can be established, which is similar to the real robotic environment.

The components A and B are separately mounted on the manipulator in sequence and the relative movements of the manipulator will cause deformation offset of the components. The corresponding force Jacobian matrices of components A and B are given in (19) and (20).

$$J_{FA} = \begin{bmatrix} -0.133\ 0 & -0.010\ 1 \\ -0.004\ 1 & 0.063\ 1 \end{bmatrix} \mu\text{m/mN} \tag{19}$$

$$J_{FB} = \begin{bmatrix} -0.232\ 8 & -0.032\ 1 \\ -0.036\ 5 & 0.525\ 1 \end{bmatrix} \mu\text{m/mN} \tag{20}$$

where J_{FA} and J_{FB} are the force Jacobian matrices of components A and B, respectively.

The components A and B are trained in the corresponding simulation environments and the corresponding model A and model B are saved in the skill pool. Four experiments are conducted for each component with the learned insertion policy. And all of the eight experiments finished successfully. The distributions of the contact force are given in Fig. 8. In the insertion tasks with component A, the contact force less than 50 mN occupies over

90%. As for component B, the contact force less than 30 mN occupies over 95%. The success of the experiments validates the effectiveness of our method. It can save a lot of time and provide a much more convenient way to train RL model for assembly tasks on real robotic system.

In order to further validate the feasibility of the skill saving and selection mechanism, the component C is viewed as a new component used in the insertion tasks. The component C is mounted on manipulator and the relative movements of manipulator will cause deformation offset of the component. The corresponding force Jacobian matrix J_{FC} is firstly obtained and given in (21).

$$J_{FC} = \begin{bmatrix} -0.230\,4 & 0.023\,7 \\ -0.000\,1 & 0.258\,4 \end{bmatrix} \mu\text{m/mN}. \tag{21}$$

There are two trained models, model A and model B, in the skill pool. The distance, $\{D_i|\ i{=}1,\ 2\}$, between J_{FC} and the force Jacobian matrices of two models are computed with (18). The materials of components A and C are similar, and as expected, D_1 is smaller than D_2. Then the model A is selected to complete the insertion tasks with component C. The maximum insertion step is set as 50 μm. Four experiments are conducted with different initial states. Some details of one insertion task are shown in Fig. 9. The contact force f_x and f_y are kept within a safe range during the insertion process. Adjustments d_x and d_y are employed to reduce the contact force. The insertion step d_z decreases as the contact force increases. The distribution of the radial contact force is shown in Fig. 8. The contact force less than 30 mN occupies over 96%. The above results verify the feasibility of the skill saving and selection mechanism which promotes the adaptabil-

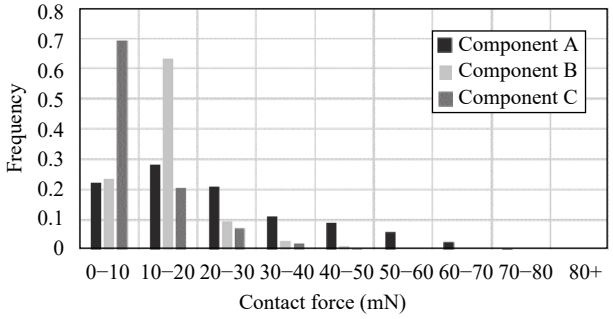

Fig. 8 Distributions of radial contact force with components A, B and C.

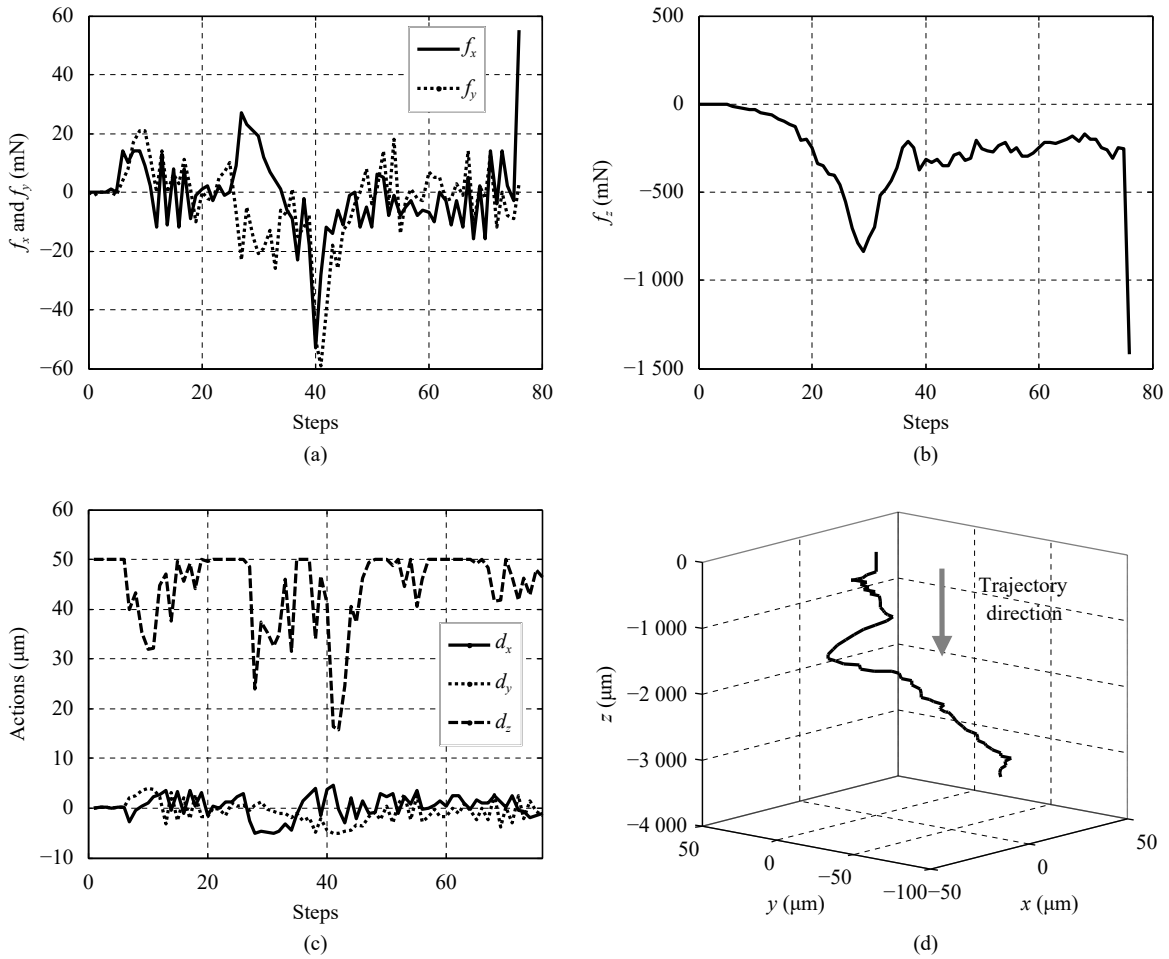

Fig. 9 Results of one insertion task with component C: (a) Contact force f_x and f_y; (b) Contact force f_z; (c) Actions; (d) The whole trajectory.

ity to new components.

6 Conclusions

A DDPG-based skill learning framework is proposed for robotic insertion. Considering both the safety and efficiency, the action to be executed is composed of two parts: an expert action and a refinement action, which are learned from one demonstration and RL, respectively. The episode-step exploration strategy is designed to improve training efficiency of RL. In order to improve the adaptability of the insertion skill learning method, a skill saving and selection mechanism is designed. It is convenient to select an appropriate model from the skill pool to execute insertion tasks when meeting new components. To bridge the gap between simulation and real robotic systems, a simulation environment is established under the guidance of force Jacobian matrix. Then the models can be trained in the simulation environment and be used directly in the real insertion tasks. The results of simulations and experiments show the effectiveness of the proposed insertion skill learning framework.

Acknowledgements

This work was supported by National Key Research and Development Program of China (No. 2018AAA0103005) and National Natural Science Foundation of China (No. 61873266).

References

[1] S. Liu, D. Xu, D. P. Zhang, Z. T. Zhang. High precision automatic assembly based on microscopic vision and force information. *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 382–393, 2016. DOI: 10.1109/TASE.2014.2332543.

[2] J. Zhang, D. Xu, Z. T. Zhang, W. S. Zhang. Position/force hybrid control system for high precision aligning of small gripper to ring object. *International Journal of Automation and Computing*, vol. 10, no. 4, pp. 360–367, 2013. DOI: 10.1007/s11633-013-0732-y.

[3] F. B. Qin, D. Xu, D. P. Zhang, Y. Li. Robotic skill learning for precision assembly with microscopic vision and force feedback. *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 1117–1128, 2019. DOI: 10.1109/TMECH.2019.2909081.

[4] M. Armin, P. N. Roy, S. K. Das. A survey on modelling and compensation for hysteresis in high speed nanopositioning of AFMs: Observation and future recommendation. *International Journal of Automation and Computing*, vol. 17, no. 4, pp. 479–501, 2020. DOI: 10.1007/s11633-020-1225-4.

[5] K. G. Zhang, J. Xu, H. P. Chen, J. G. Zhao, K. Chen. Jamming analysis and force control for flexible dual peg-in-hole assembly. *IEEE Transactions on Industrial Electronics*, vol. 66, no. 3, pp. 1930–1939, 2019. DOI: 10.1109/TIE.2018.2838069.

[6] S. Liu, Y. F. Li, D. P. Xing. Sensing and control for simultaneous precision peg-in-hole assembly of multiple objects. *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 310–324, 2020. DOI: 10.1109/TASE.2019.2921224.

[7] F. Chen, F. Cannella, J. Huang, H. Sasaki, T. Fukuda. A study on error recovery search strategies of electronic connector mating for robotic fault-tolerant assembly. *Journal of Intelligent & Robotic Systems*, vol. 81, no. 2, pp. 257–271, 2016. DOI: 10.1007/s10846-015-0248-5.

[8] D. P. Xing, Y. Lv, S. Liu, D. Xu, F. F. Liu. Efficient insertion of multiple objects parallel connected by passive compliant mechanisms in precision assembly. *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 4878–4887, 2019. DOI: 10.1109/TII.2019.2897731.

[9] J. Takahashi, T. Fukukawa, T. Fukuda. Passive alignment principle for robotic assembly between a ring and a shaft with extremely narrow clearance. *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 1, pp. 196–204, 2016. DOI: 10.1109/TMECH.2015.2448639.

[10] 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, D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. DOI: 10.1038/nature14236.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

[12] J. L. Luo, E. Solowjow, C. T. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, P. Abbeel. Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *Proceedings of the International Conference on Robotics and Automation*, IEEE, Montreal, Canada, pp. 3080−3087, 2019. DOI: 10.1109/ICRA.2019.8793506.

[13] T. Johannink, S. Bahl, A. Nair, J. L. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, S. Levine. Residual reinforcement learning for robot control. In *Proceedings of International Conference on Robotics and Automation*, IEEE, Montreal, Canada, pp. 6023−6029, 2019. DOI: 10.1109/ICRA.2019.8794127.

[14] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, R. Tachibana. Deep reinforcement learning for high precision assembly tasks. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Vancouver, Canada, pp. 819−825, 2017. DOI: 10.1109/IROS.2017.8202244.

[15] F. M. Li, Q. Jiang, S. S. Zhang, M. Wei, R. Song. Robot skill acquisition in assembly process using deep reinforcement learning. *Neurocomputing*, vol. 345, pp. 92–102, 2019. DOI: 10.1016/j.neucom.2019.01.087.

[16] Y. X. Fan, J. L. Luo, M. Tomizuka. A learning framework for high precision industrial assembly. In *Proceedings of International Conference on Robotics and Automation*, IEEE, Montreal, Canada, pp. 811−817, 2019. DOI: 10.1109/ICRA.2019.8793659.

[17] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, J. Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. In *Proceedings of International Conference on Robotics and Automation*, IEEE, Montreal, Canada, pp. 754−760, 2019. DOI: 10.1109/ICRA.2019.8794074.

[18] J. L. Luo, E. Solowjow, C. G. Wen, J. A. Ojea, A. M. Ago-

gino. Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Madrid, Spain, pp. 2062–2069, 2018. DOI: 10.1109/IROS.2018.8594353.

[19] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, P. Abbeel. Learning robotic assembly from CAD. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Brisbane, Australia, pp. 3524–3531, 2018. DOI: 10.1109/ICRA.2018.8460696.

[20] Z. M. Hou, H. M. Dong, K. G. Zhang, Q. Gao, K. Chen, J. Xu. Knowledge-driven deep deterministic policy gradient for robotic multiple peg-in-hole assembly tasks. In *Proceedings of IEEE International Conference on Robotics and Biomimetics*, IEEE, Kuala Lumpur, Malaysia, pp. 256–261, 2018. DOI: 10.1109/ROBIO.2018.8665255.

[21] J. Xu, Z. M. Hou, W. Wang, B. H. Xu, K. G. Zhang, K. Chen. Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks. *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1658–1667, 2019. DOI: 10.1109/TII.2018.2868859.

Ying Li received the B. Sc. degree in control science and engineering from North China Electric Power University (Baoding), China in 2016. He is a Ph. D. degree candidate at Institute of Automation, Chinese Academy of Sciences (IACAS), China.

His research interests include visual measurement, visual control, micro-assembly and machine learning.

E-mail: liying2016@ia.ac.cn.

ORCID iD: 0000-0002-0213-9247

De Xu He received his B. Sc. degree in control science and engineering and M. Sc. degree in control science and engineering from Shandong University of Technology, China in 1985 and 1990, respectively, and received the Ph. D. degree in control science and engineering from Zhejiang University, China in 2001. He is a professor at the Institute of Automation, Chinese Academy of Sciences (IACAS), China.

His research interests include visual measurement, visual control, intelligent control, visual positioning, microscopic vision, and micro-assembly.

E-mail: de.xu@ia.ac.cn. (Corresponding author)

ORCID iD: 0000-0002-7221-1654