# Feedback Deep Deterministic Policy Gradient With Fuzzy Reward for Robotic Multiple Peg-in-Hole Assembly Tasks

Jing Xu, *Member, IEEE*, Zhimin Hou, Wei Wang, Bohao Xu, Kuangen Zhang, and Ken Chen

***Abstract*—The automatic completion of multiple peg-in-hole assembly tasks by robots remains a formidable challenge because the traditional control strategies require a complex analysis of the contact model. In this paper, the assembly task is formulated as a Markov decision process, and a model-driven deep deterministic policy gradient algorithm is proposed to accomplish the assembly task through the learned policy without analyzing the contact states. In our algorithm, the learning process is driven by a simple traditional force controller. In addition, a feedback exploration strategy is proposed to ensure that our algorithm can efficiently explore the optimal assembly policy and avoid risky actions, which can address the data efficiency and guarantee stability in realistic assembly scenarios. To improve the learning efficiency, we utilize a fuzzy reward system for the complex assembly process. Then, simulations and realistic experiments of a dual peg-in-hole assembly demonstrate the effectiveness of the proposed algorithm. The advantages of the fuzzy reward system and feedback exploration strategy are validated by comparing the performances of different cases in simulations and experiments.**

***Index Terms*—Continuous actions control, feedback exploration, fuzzy reward, intelligent assembly, multiple peg-in-hole, reinforcement learning.**

## I. Introduction

INDUSTRIAL robots are playing a more and more significant role in modern industrial systems for costing less and having a greater range of capabilities to handle complex tasks during manufacturing process. Peg-in-hole assembly is a common task, but automating the multiple peg-in-hole assembly process remains a challenge [1]. Active force control strategies are practical in realistic robotic assembly applications. A two-element method and screw theory were presented in [2], and they can analyze the geometric states and the contact model of a three-dimensional (3-D) triple peg-in-hole insertion. In [3], Zhang *et al.* proposed a more practical fuzzy force control strategy for a rigid dual peg-in-hole assembly task based on a simplified analysis method. However, the above control algorithms are challenge to be applied in multiple peg-in-hole assembly scenarios for the complex contact model. Considerable time and efforts are required to tune the controller parameters with respect to the new assembly task. Therefore, an advanced algorithm without depending on the analysis of the physical contact model is required to perform multiple peg-in-hole assembly tasks well.

In contrast to robot programming, which depends on the model analysis, human workers can accomplish peg-in-hole assembly tasks with their learned experience [4]. Inspired by this, industrial robots can perform complex peg-in-hole assembly through learning assembly skills. The standard reinforcement learning (RL) algorithm is implemented in robotic applications by developing an agent that can learn the desired policy through maximizing the rewards received from the environment [5]. Therefore, an RL can be applied to enable robots to learn assembly skills through trials instead of only transferring human skills to robot program [6]. The RL technique was used to learn a single peg-in-hole assembly operation in [7]. Long short-term memory was trained combining with RL to learn a high-precision peg-in-hole assembly policy in [8]. Nevertheless, the above approaches output the discrete actions by discretizing the action space, which still has many limitations in continuous actions control tasks [9] that the output actions to control robots are continuous and high-dimensional.

Deep learning as a powerful nonlinear approximation approach has promoted the considerable performance of deep reinforcement learning (DRL) algorithms [10] in realistic robotic applications. Especially, convolutional neural networks have made robots achieve successful grasps of objects from images [11]. DRL has been applied to tune the controller parameters automatically for the industrial assembly tasks [12] and control a quadrotor by the learned policy network in simulation [13]. In particular, deep deterministic policy gradient (DDPG)

Manuscript received February 23, 2018; revised May 6, 2018 and August 18, 2018; accepted September 2, 2018. Date of publication September 5, 2018; date of current version March 1, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 51675291 and Grant U1613205, and in part by the State Key Laboratory of China (SKLT2018C04). Paper no. TII-18-0499. *(Corresponding author: Jing Xu.)*

J. Xu, Z. Hou, W. Wang, K. Zhang, and K. Chen are with the State Key Laboratory of Tribology, Beijing Key Laboratory of Precision/Ultra-Precision Manufacturing Equipment Control, Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China (e-mail: jingxu@tsinghua.edu.cn; houzm16@mails.tsinghua.edu.cn; wangw17@mails.tsinghua.edu.cn; zhangke12@tsinghua.org.cn; kenchen@tsinghua.edu.cn).

B. Xu is with the Department of Mechanical Engineering, Dalian Jiaotong University, Dalian 116028, China (e-mail: xvbohao@163.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2018.2868859

[9] is demonstrated to perform stably and efficiently on many continuous and high-dimensional action control tasks in simulation environment. Nevertheless, the following challenges still remain when DDPG algorithms are implemented in realistic industrial assembly processes.

1) In multiple peg-in-hole assembly tasks, it is costly to collect enough experience data to train the agent in such a large search space, mapping the observed high-dimensional and continuous states to high-dimensional and continuous actions [14].
2) The reward function of DRL algorithms evaluates the actions that the agent performed [15]. However, it is difficult to find a perfect function to evaluate whether the actions could satisfy the long-term desirability.
3) The tradeoff of the exploration/exploitation strategies, the techniques for choosing actions during the learning process, largely decides the efficiency of DRL algorithms. The previous exploration approaches for continuous actions control tasks only try to explore the better action and novel state efficiently [16]. However, in realistic robotic tasks, the safety of the exploration action need to be considered first [17].

To utilize the superiority of the DDPG algorithm for performing multiple peg-in-hole assembly tasks well, we present a model-driven deep deterministic policy gradient (MDDPG) algorithm that enables the robot to learn a high-level assembly policy in realistic scenarios. The training process of the MDDPG algorithm is driven by a traditional force control strategy rather than learning from a zero basis, which can decrease the number of training experiments and avoid risky actions. We propose a fuzzy reward system rather than designing a complex reward function. The proposed fuzzy reward system incorporating prior knowledge evaluates the assembly process, taking more comprehensive factors into account than previous works, which not only improves the learning efficiency but also avoids the agent becoming trapped in a local minimum.

In addition, we also present a feedback exploration strategy, which can tune the exploration process according to the performance of the last training step. The primary exploration strategy in continuous action space is $\varepsilon-$greedy exploration [18] by picking a random action noise with probability $\varepsilon$. In contrast to fixed exploration methods, an active exploration strategy is presented in [19] by applying the Gaussian noise to the continuous action parameters. However, the aforementioned exploratory approaches rely on collecting sufficient samples from environments and probably result in meaningless or risky actions, which is impermissible in realistic assembly scenarios. To eliminate the meaningless exploration trials and risky exploration actions, the feedback exploration strategy incorporates expert knowledge into modifying exploration process.

A simplified assembly simulation model is constructed to test the effectiveness of our algorithm and compare with the previous algorithms without a fuzzy reward system and feedback exploration strategy. Moreover, a realistic dual peg-in-hole assembly experiment is implemented to demonstrate the availability and generality in realistic scenarios.

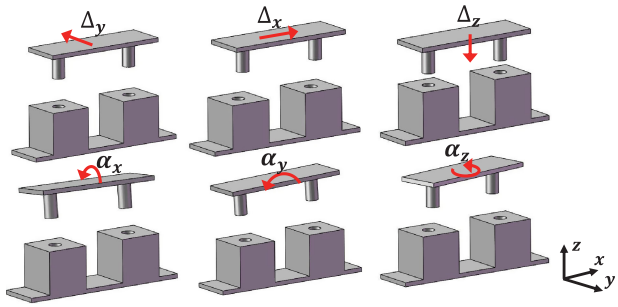



Fig. 1. Elementary movement of the pegs.

The remainder of this paper is structured as follows. Section II explains the context of the proposed algorithm. The implemented fuzzy reward system is described in Section III. The details of the feedback exploration are described in Section IV. The setup of the assembly simulation environments and the results are described in Section V. The performance in realistic experiments is provided in Section VI. The conclusions are drawn in Section VII.

## II. Algorithm

In this section, we first define the formulation of the multiple peg-in-hole assembly task. Then, we describe the foundation of the proposed MDDPG algorithm, the network setup and the network training skills.

### A. Problem Formulation

For the multiple peg-in-hole assembly task, the target is to develop an agent that learns the assembly policy through interacting with the assembly environment. The process of executing the assembly tasks can be formulated as a Markov decision process. At each time step $t$, the agent obtains a state $\mathbf{s}_t \in \mathcal{S}$, chooses an assembly action $\mathbf{a}_t \in \mathcal{A}$, and receives a scalar reward $r_t$ representing the evaluation of an action-state pair. The behaviors of the agent denote the assembly policy $\pi$ mapping states to actions. The agent improves the learned policy by maximizing the sum of the expected discounted reward $R_t$ over the future steps as in

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-t} r_T = r_t + \gamma R_{t+1} \tag{1}$$

where $\gamma \in [0, 1]$ is called the discount rate, the current reward $r_t$ depends on the performance of the action $\mathbf{a}_t$ at the state $\mathbf{s}_t$, and $T$ denotes the terminal time step of the assembly episode. The reward signal defines the goal in the assembly task, which is to decrease the assembly time and smooth the contact forces.

The assembly action consists of six motion components to move the pegs, as shown in Fig. 1. Therefore, the action $\mathbf{a}_t$ is a 6-D vector defined as follows:

$$\mathbf{a}_t = [\Delta_x, \Delta_y, \Delta_z, \alpha_x, \alpha_y, \alpha_z] \tag{2}$$

where $\Delta$ and $\alpha$ are the translation and rotation component, respectively; the subscripts $x$, $y$, and $z$ denote the axes of the base coordinate.

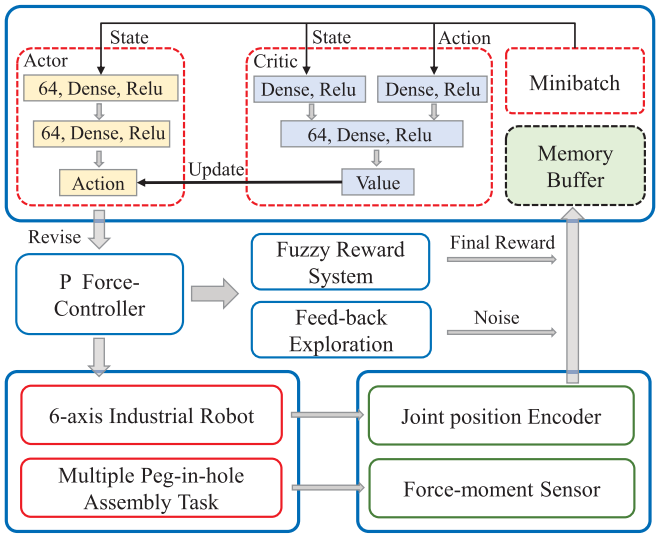



Fig. 2. Framework of the MDDPG algorithm, including network architecture, assembly environment, fuzzy reward system, and feedback exploration mechanism.

The state $\mathbf{s}_t$ consists of 12 elements defined as follows:

$$\mathbf{s}_t = [P_x, P_y, P_z, O_x, O_y, O_z, F_x, F_y, F_z, M_x, M_y, M_z] \quad (3)$$

where $P$ and $O$ are the current position and orientation of the pegs in the base coordinate, respectively, which can be calculated according to the joint angles; $F$ and $M$ are the forces and moments measured from the force-moment sensor.

## B. Network Setup

The framework of the MDDPG algorithm is illustrated in Fig. 2, which is built based on the actor-critic approach trained by the theory of deterministic policy gradient (DPG) [20].

The critic network approximates the action-value $Q(\mathbf{s}_t, \mathbf{a}_t|\theta^Q)$ of the executed action $\mathbf{a}_t$ at the state $\mathbf{s}_t$. The action-value indicates the long-term desirability after taking the states that are likely to follow into account. The critic network is optimized by minimizing the loss

$$L(\theta^Q) = E_{\pi'}\left[\left(Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t|\theta^Q) - y_t\right)^2\right] \quad (4)$$

where $y_t = r_t(\mathbf{s}_t, \mathbf{a}_t) + \gamma Q^{\pi}(\mathbf{s}_{t+1}, \pi(\mathbf{s}_{t+1})|\theta^Q)$ is the target action value calculated using the Bellman equation.

The actor network is used to approximate the assembly policy mapping the state to a deterministic action. The actor network is updated by computing the policy gradient with respect to the actor parameters $\theta^{\pi}$ by the chain rule, which is the gradient of the policy performance $J$ proven in [20]

$$\begin{aligned}\nabla_{\theta^{\pi}} J &\approx \mathbb{E}_{\pi}\left[\nabla_{\theta^{\pi}} Q^{\pi}(\mathbf{s}_t, \pi(\mathbf{s}_t|\theta^{\pi})|\theta^Q)\right] \\ &= \mathbb{E}_{\pi}\left[\nabla_{\mathbf{a}} Q^{\pi}(\mathbf{s}_t, \pi(\mathbf{s}_t)|\theta^Q)\nabla_{\theta^{\pi}}\pi(\mathbf{s}_t|\theta^{\pi})\right]. \quad (5)\end{aligned}$$

Although most of the DDPG algorithms can perform well using the above optimization approach [9], the data efficiency and safety guarantees are the main limitations of applying such algorithms in realistic physical tasks. To search the optimal policies, the conventional DDPG algorithms explore all possible actions, which requires a substantial amount of experimental data and also probably generates some risky actions in realistic environments. As shown in Fig. 2, the training process of the MDDPG algorithm is driven by the traditional force control strategy. The output action $\mathbf{a}_t$ from the actor network is dimensionless and every component is limited in the range of $[-b, +b]$, which is not used to control robot directly, where $b$ denotes the degree of learning using the proposed MDDPG algorithm. In our paper, the action $\mathbf{a}_t$ is utilized to revise the basic actions $\mathbf{a}_t^e$ output from the $P$ force controller given the current measured forces and moments $\mathbf{F}_t$ by

$$\mathbf{a}_t^e = \mathbf{K}_p \circ (\mathbf{F}_t - \mathbf{F}_{\text{ref}}) \quad (6)$$

$$\bar{\mathbf{a}} = \mathbf{a}_t^e \circ \mathbf{a}_t + \mathbf{a}_t^e \quad (7)$$

where $\bar{\mathbf{a}}$ including translation and rotation offsets values corresponding to the definition in Fig. 1 is the final output action to control the robot executing the assembly process ($\circ$ represents the Hadamard product), $\mathbf{K}_p$ is a vector including six proportional coefficients corresponding to six action components, respectively, and $\mathbf{F}_{\text{ref}}$ is the reference forces and moments. The basic actions can ensure that the actor network avoids exploring the meaningless and dangerous policies, which corresponds to the agent having handled the basic assembly skills rather than the zero basis.

For the multiple peg-in-hole assembly task, the desirability of the assembly action should depend on both the assembly time and the contact forces rather than only optimizing the assembly time as in [8]. Therefore, as shown in Fig. 2, a fuzzy reward system in consideration of more factors is proposed to offer the rewards rather than the previous reward functions, as described in Section III. In addition, we utilize a feedback exploration strategy to tune the exploration process in real-time according to the measurements and prior knowledge of assembly tasks, as described in Section IV.

## C. Network Training

It is necessary to use two parallel threads during the training process of the proposed algorithm in realistic assembly scenarios since the data collection and training processes have different frequencies. Therefore, to efficiently perform the data collection process, a memory buffer with a finite size is used to store the experience data tuple $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ in a first-in-first-out manner. At each training step, mini-batch-size samples randomly taken from the memory buffer are used to update the actor and critic networks if the number of the data tuple in memory buffer is above mini-batch-size. In addition, to guarantee fast convergence and stability, the following training skills are implemented in the training process.

1) Copies of the actor and critic networks as the target networks are used to offer the target values for training. The target values change slowly to improve the stability of the training process as in supervised learning, which is essential in realistic scenarios.
2) The observed states from environments include four components with different physical units (N, Nm, and mm).

To avoid ineffective learning for the different state ranges, every state component is normalized in the range of $[-1, +1]$. Moreover, the batch normalization is implemented in all dense layers of the networks.

3) In the critic network, two different layers in the first hidden layer are utilized to represent the state and action rather than concatenating them directly, as shown in Fig. 2. Different numbers of nodes in the two layers can control the importance of the state and action in different cases.

## III. Fuzzy Reward System

In this section, we describe the context of the implementation of the proposed fuzzy reward system, which can take more factors into account and describe more complex reward logic through embedding the prior knowledge of the assembly task to set the fuzzy rules.

For the multiple peg-in-hole assembly task, the output scalar reward $r$ consists of two components

$$r = r_1 + r_2 \tag{8}$$

where $r_1$ is the positive reward at the end of each episode to reward the agent for accomplishing the assembly task successfully and $r_2$ is the negative reward at each time step to punish a low assembly speed and large contact forces. If the assembly task can be finished successfully with $K$ time steps when the depth of the pegs reaches the goal and the contact forces never exceeds the safe boundary, then the agent can receive a positive reward limited in the range of $[0, +1)$ as in

$$r_1 = 1 - \frac{K}{K_{\max}} \tag{9}$$

where $K_{\max}$ is the given maximum step number.

The fuzzy reward system is utilized to provide the negative reward $r_2$ depending on four factors: the current depth of pegs $D_z$, the current translation offset component $\Delta_z$ along the $z$ axis of the base coordinate, the maximum contact forces $F_{\max}$, and the maximum moments $M_{\max}$. If each input factor value is converted into fuzzy values in five ranges during the fuzzier stage, then the membership function of the fuzzy reward system will include 625 fuzzy rules. To simplify the membership function, a two-layer fuzzy system with equal effects is designed that consists of three fuzzy sets: two first-layer fuzzy sets ($Translation$–$Depth$, $Force$–$Moment$) and a second-layer fuzzy set, as depicted in Fig. 3.

Each crisp input value of all three fuzzy sets is divided into five ranges. $VB$, $B$, $N$, $G$, and $VG$ denote very bad, bad, normal, good, and very good, respectively. Thus, each fuzzy set includes 25 rules, and the total number of fuzzy rules is reduced to 75. The fuzzy rules of the three fuzzy sets are shown in Fig. 4. The crisp outputs of the first-layer fuzzy sets are limited in the range of $[-1, 0)$, which are the crisp inputs of the second-layer fuzzy set. The crisp output of the second-layer fuzzy set as the final negative reward is also limited in the range of $[-1, 0)$, as shown in Fig. 4(c). The output negative reward is $-1$, corresponding to the maximum penalty for the worst inputs, where the agent performed the worst assembly policy at this time step.

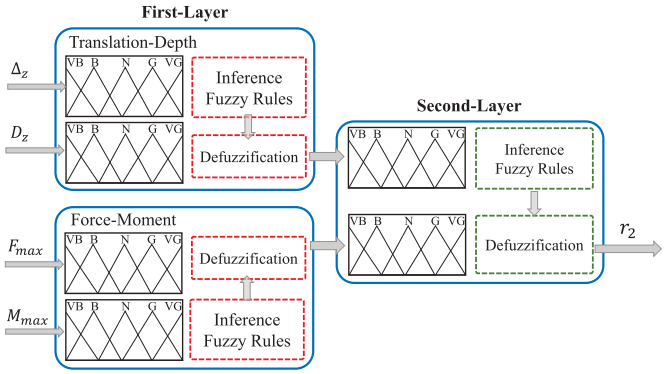



Fig. 3. Architecture of fuzzy reward system and details of each fuzzy set.

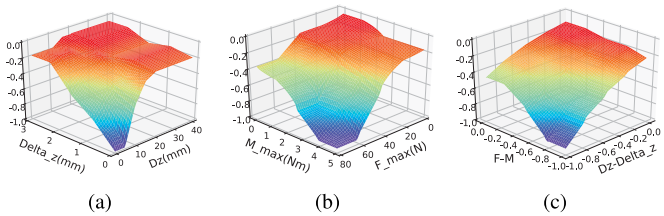



Fig. 4. Fuzzy rules of the fuzzy reward system. (a) Rules of fuzzy set with the input of translation and depth. (b) Rules of fuzzy set with the input of forces and moments. (c) Rules of the second-layer fuzzy set.

The defuzzifier of all three fuzzy sets in the proposed fuzzy reward system using the weighted average functions is defined as follows:

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{25} C^i \prod_{j=1}^{n} g_j^i(x_j)}{\sum_{i=1}^{25} \prod_{j=1}^{n} g_j^i(x_j)} \tag{10}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T$ and $f(\mathbf{x})$ denote the crisp input and output, respectively; n is the number of inputs; $g(\mathbf{x})$ is the triangular membership function, which is different for the three fuzzy sets; $C^i$ denotes the value of the output defined with respect to the fuzzy rule $i$, which is decided by the prior knowledge.

The fuzzy reward system could meet the flexible requirements instead of the complex implementations utilizing an accurate function [8]. For instance, during the assembly process, $\Delta_z$ and $D_z$ contribute different importance weights to the final reward at the different stages according to the previous assembly experience, as shown in Fig. 4(a). Moreover, the fuzzy reward system can easily be tuned in terms of the different cases.

## IV. Feedback Exploration Strategy

This section introduces the implementation of the proposed feedback exploration strategy, and the structure is depicted in Fig. 5. The performance of previous exploration results is estimated in cooperation with the prior knowledge to guide the regulation mechanism, which can modify the exploration process.

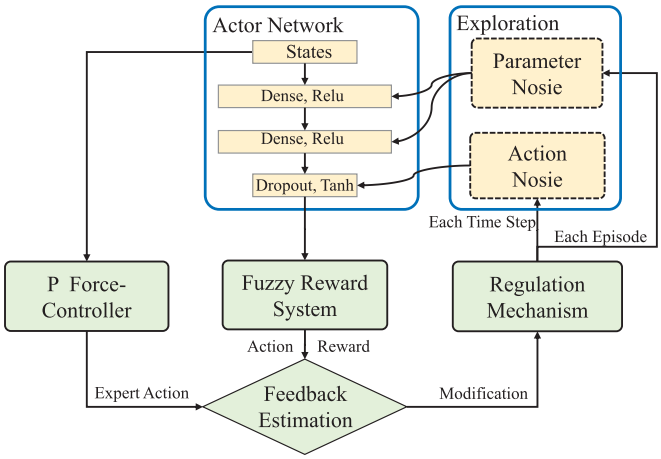



Fig. 5. Implementation of feedback exploration strategy.

The implementation of the main exploration methods is directly adding random action noise to the actor policy. However, the feedback performance cannot immediately improve the decision ability of the agent because the parameters of the actor network are updated step by step. To efficiently explore the optimal assembly policies, in addition to the action space noise, we also add parameter space noise to the parameters of the actor network.

### A. Action Space Noise Exploration

In multiple peg-in-hole assembly tasks, to ensure the safety of assembly actions, the *P* force controller is used as the expert advice to guide the modification of the exploration process. The distance $d_s(\cdot)$ between the perturbed action $\tilde{\mathbf{a}}_t$ and the expert action $\mathbf{a}_e$ can be measured in action space at each time step as follows:

$$d_s\left(\mathbf{a}_e, \tilde{\mathbf{a}}_t\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(a_e^i - \tilde{a}_t^i\right)^2} \tag{11}$$

where $N$ denotes the dimension of the action and $i$ is the index. The perturbed assembly action $\tilde{\mathbf{a}}_t$ is injected with $OU(\mu, \xi, \sigma^2)$ noise, which is correlated and controlled by the parameter $\xi$, $\mu$ is the desired average, and $\sigma$ is the variance. In the regulation mechanism, $\mu$ and $\sigma$ can be tuned at each time step according to the measured distance $d_s(\cdot)$ and the difference value $\Delta r$

$$\Delta r = r_t - r_e \tag{12}$$

where the current reward $r_t$ of the perturbed action can be calculated from fuzzy reward system $r_t = f(F_{\max}^t, M_{\max}^t, D_z^t, \Delta_z^t)$. The expert reward $r_e$ cannot be measured after executing the perturbed action. The value calculated by the forces $F_{\max}^{t-1}$ and moments $M_{\max}^{t-1}$ at the last time step, the expert action $\Delta_z^e$ and the depth $D_z^e$ are used to approximate the expert reward $r_e$.

Therefore, $\mu$ and $\sigma$ can be tuned by the following:

$$\sigma_{t+1} = \sigma_t - B_\sigma \cdot \text{sigmoid}\left(\frac{\Delta r}{d_s + 1}\right) \tag{13}$$

$$\mu_{t+1} = B_\mu \cdot \tanh\left(\Delta_z^t - \Delta_z^e\right) \tag{14}$$

where $B_\sigma$ and $B_\mu$ relate to the upper bound according to the realistic assembly tasks. We only modify the average value of action noise with respect to the translation component $\Delta_z$, and the values of other components are set to zero because of their tiny change. This regulation implementation means that the exploration should be decreased when the agent performs well and that the exploration should also be increased when it performs worse than the expert.

### B. Parameter Space Noise Exploration

To avoid more consistent exploration resulting from the action space exploration, we also inject additive Gaussian noise $\mathcal{N}(0, \sigma^2 I)$ into the parameters of the current policy as the parameter space exploration noise at the end of each episode by

$$\tilde{\theta}^\pi = \theta^\pi + \mathcal{N}\left(0, \sigma^2 \mathbf{I}\right) \tag{15}$$

where $\theta^\pi$ denotes the parameters of the actor network and $\tilde{\theta}^\pi$ denotes the perturbed parameters.

We utilize the performance of the current episode as the short-term average reward $\bar{r}_m$, calculated by

$$\bar{r}_m = \sum_{t=0}^{T} \gamma^{T-t} r_t + r_1 \tag{16}$$

where $m$ is the index number of the episode, and $T$ denotes the total time step of each episode. In addition, we measure the long-term average reward $\bar{\bar{r}}_m$ of all the previous episodes as a reference. When $\bar{r}_m > \bar{\bar{r}}_m$, the exploration process should be decreased because the current performance has exceeded the average. Additionally, when $\bar{r}_m < \bar{\bar{r}}_m$, the exploration process should be increased because the current performance is worse than the average. The variance $\sigma$ can be tuned by

$$\sigma_{m+1} = \begin{cases} 1.01 \cdot \sigma_m, & d_p(\pi_p, \tilde{\pi}) < \delta_{m+1} \\ \sigma_m / 1.01, & d_p(\pi_p, \tilde{\pi}) \geq \delta_{m+1} \end{cases} \tag{17}$$

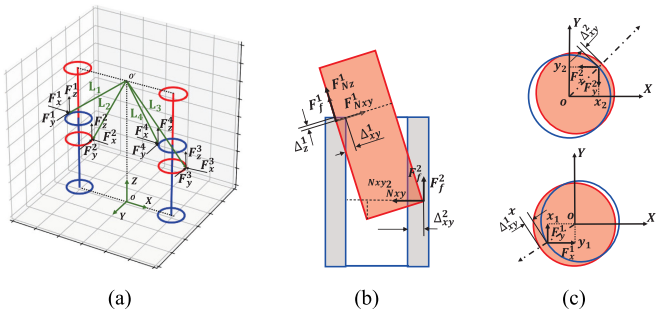
where $d_p(\pi_p, \tilde{\pi})$ denotes the distance between the current policy $\pi_p$ from the actor network and the perturbed policy $\tilde{\pi}$ as in

$$d_p\left(\pi_p, \tilde{\pi}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_\mathbf{s}\left[\left(\pi_p^i - \tilde{\pi}^i\right)^2\right]} \tag{18}$$

where $N$ denotes the dimension of the policy, $\mathbb{E}$ denotes the expectation of the mini-batch-size samples, and $\delta_{m+1}$ denotes the threshold value, which can be updated by

$$\delta_{m+1} = \max\{0,\ \vartheta \Delta \bar{r}_m - \varepsilon\} \tag{19}$$

where $\vartheta$ denotes the update rate in our exploration strategy and $\varepsilon$ is a small constant to ensure a decrease in exploration at each



(a) (b) (c)

Fig. 6. Implementation of simulation model. (a) Simplified dual peg-in-hole model. (b) Analysis of the largest deformation in the plane. (c) Two projection sections cross upper contact point 1 and lower contact point 2.

episode. The difference value $\Delta\bar{r}_m$ can be calculated by

$$\Delta\bar{r}_m = \bar{r}_m - \bar{\bar{r}}_m \tag{20}$$

## V. Simulation

This section demonstrates the feasibility of the MDDPG algorithm and the effectiveness of the fuzzy reward system and feedback exploration strategy in a dual peg-in-hole assembly simulation model. Particularly, a simplified force simulator is built to simulate the dual peg-in-hole assembly task.

### A. Simulation Setup

As shown in Fig. 6(a), the four red circles represent the dual pegs and the four blue circles represent the dual holes, the base coordinate $(OXYZ)$ is set at the bottom of the holes. We assume that the pegs and holes are strict point contacts, the pegs are strictly rigid and the holes are elastic. The point with the largest deformation is considered to be the contact point. The contact pressure can be approximated by

$$F_N = E\delta \tag{21}$$

where $\delta$ is the deformation at the contact point along the given direction and $E$ denotes the stiffness coefficient of the holes, which is set to 200 in this simulation. The friction model is simplified such that the contact friction $F_f$ can be calculated by

$$F_f = \mu F_N \tag{22}$$

where $\mu$ denotes the friction coefficient, which is set to 0.01.

Two projection sections are required to cross the upper contact point $(x_1, y_1, z_1)$ and lower contact point $(x_2, y_2, z_2)$ of the left peg-hole pair in Fig. 6(a). As shown in Fig. 6(b), the contact forces at the contact point $(x_1, y_1, z_1)$ can be calculated by

$$\begin{cases} F_x^1 = \dfrac{y_1}{\sqrt{y_1^2 + x_1^2}} \cdot F_{Nxy}^1 \\ F_y^1 = \dfrac{x_1}{\sqrt{y_1^2 + x_1^2}} \cdot F_{Nxy}^1 \\ F_z^1 = F_f^1 + F_{Nz}^1 = \mu F_{Nxy}^1 + F_{Nz}^1 \end{cases} \tag{23}$$

where $F_{Nxy}^1$ and $F_{Nz}^1$ are the contact pressures, calculated by the deformations $\Delta_{xy}^1$ and $\Delta_z^1$ in the X-O-Y plane and along the Z-axis of the base coordinate, respectively. In the same way, the contact forces at the contact point $(x_2, y_2, z_2)$ can be calculated, but the deformation $\Delta_z^2$ at the lower contact point can be omitted because of its small value.

TABLE I
Training Hyperparameters

| Parameters | Value |
|---|---|
| mini-batch size | 64 |
| actor learning rate | 1e-3 |
| critic learning rate | 1e-2 |
| target update rate | 1e-3 |
| memory pool size | 1e4 |
| reward discount | 0.95 |

After all the contact forces at the different contact points are calculated, the total contact forces $\mathbf{F} = [F_x, F_y, F_z]$ and moments $\mathbf{M} = [M_x, M_y, M_z]$ of the pegs at the point $O'$ can be calculated, respectively, by the following:

$$\begin{cases} \mathbf{F} = \sum_j^n \mathbf{F}^j + \mathcal{N}\left(0, \sigma_F^2 \mathbf{I}\right) \\ \mathbf{M} = \sum_j^n \mathbf{L}_j \times \mathbf{F}^j + \mathcal{N}\left(0, \sigma_M^2 \mathbf{I}\right) \end{cases} \tag{24}$$

where $j$ is the index of the contact point and $n$ is the total number of contact point; $\mathbf{L}_j$ is the coordinate vector (from the point $O'$ to the contact point $j$) in the base coordinate. To simulate the sensor noise in realistic scenarios, we add Gaussian noise $\mathcal{N}$ to the calculated results.

### B. Simulation Implementation

The dimensions of the pegs and holes are defined in Table II. In the dual peg-in-hole assembly simulation environments, the goal depth is set to 40 mm and the maximum number of steps $K_{\max}$ is set to 50. The proportional coefficient vector $\mathbf{K}_p$ ($[0.003, 0.003, 0.02, 0.01, 0.01, 0.01]$) is pretested and $b$ is set to 0.2 in simulation. In each episode, the initial position of the pegs is selected randomly in a limited range, and the initial position should ensure that the contact forces can be detected. At each time step, the forces and moments could be estimated through the built force simulator given the current position and orientation of pegs. The agent outputs the actions to move the pegs according to the current state and the next state could be realized easily. Gaussian noise is injected into the actions to simulate the implementation process by robot. For all the agents in different cases, every training process runs for 50 times after every 10 episodes and the performance is shown after 1000 episodes. The primary training hyperparameters of the networks are decided through testing, as described in Table I.

First, we evaluate the performance of the fuzzy reward system by comparing it with the previous accurate reward function in two different cases: the agents utilize OU noise as the exploration strategy and the agents utilize parameter space noise. As shown in Fig. 7, the agent using the fuzzy reward system can converge faster and learn a better assembly policy with fewer steps and higher accumulative reward in one episode.

Moreover, in the two cases, the agents with the accurate reward function perform worse in the early training stage since it is difficult to indicate whether the agent performs a good action by balancing the assembly time and contact forces. However, the fuzzy reward system can stabilize the training process.

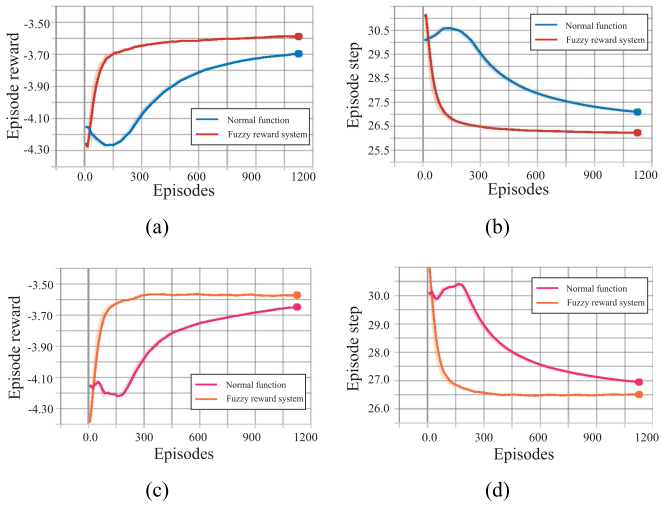



Fig. 7. Performance curves for fuzzy reward system (to show the overall tendency and comparisons between terms clearly, all the curves have been smoothed by graph software named tensorboard). (a) Accumulative reward using OU noise. (b) Number of steps using OU noise. (c) Accumulative reward using parameter space noise. (d) Number of steps using parameter space noise.

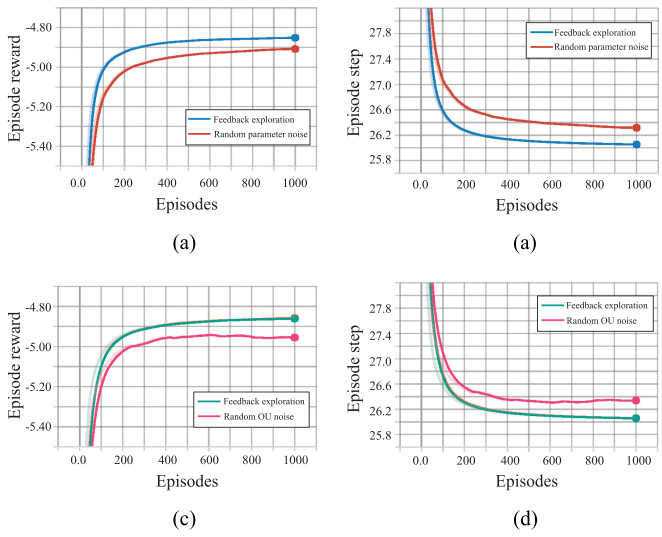



Fig. 8. Performance curves for feedback exploration strategy (to show the overall tendency and comparisons between terms clearly, all the curves have been smoothed by graph software named tensorboard). (a) Accumulative reward using parameter space noise. (b) Number of steps using parameter space noise. (c) Accumulative reward using OU noise. (d) Number of steps using OU noise.

In addition, we evaluate the performance of the feedback exploration strategy by comparing it with the agents using random noise. To demonstrate the generality, we evaluate the performance of two agents using the fuzzy reward system with action space OU noise and parameter space noise. As shown in Fig. 8, the proposed feedback exploration strategy is able to converge slightly faster than that without modifying the exploration process irrespective of what noise we utilize.

Therefore, the proposed MDDPG algorithm is significantly able to learn the high-level assembly policy quickly in the dual peg-in-hole assembly simulation. It has demonstrated that the fuzzy reward system and feedback exploration strategy could make the agent perform efficiently and stably, which is most important for the MDDPG algorithm to be applied in realistic multiple peg-in-hole assembly tasks with fewer training episodes. Moreover, the simulation environment could also decide the primary parameters of the two main networks, which could reduce the trails for realistic experiments.

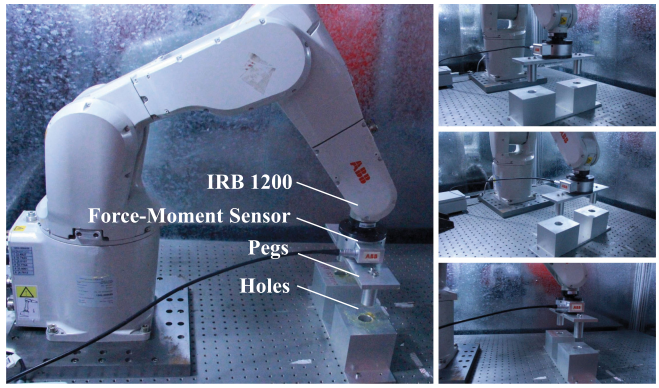



Fig. 9. Description of dual peg-in-hole assembly experimental platform.

TABLE II
DIMENSIONS OF PEGS AND HOLES

| Type | Diameter | Height |
|---|---|---|
| Pegs | 29.96 mm | 60 mm |
| Holes | 30.0 mm | 100 mm |

## VI. EXPERIMENTS

Guided by the selection of hyper-parameters for the proposed algorithm in the simulation environment, the performances of the fuzzy reward system and the feedback exploration strategy will be demonstrated in realistic dual peg-in-hole assembly environments.

### A. Experiment Setup

The architecture of the dual peg-in-hole assembly experimental platform is shown in Fig. 9. The material of the pegs and holes is aluminum. We can assume that the pegs and holes are strictly rigid because the compliance of the whole assembly process mainly depends on adjusting the position/orientation of the pegs actively. Part of the compliance is from the connectors and adapters of the whole experimental system. The dimensions of the component are defined in Table II. The distances between the two pegs' axes and between the two holes are 200 mm. The assembly task is executed by a 6-DOF ABB IRB 1200 robot, and the repeated positioning accuracy of the robot is $\pm 20\ (\mu)$m. A 6-DOF force-moment sensor and the pegs are attached to the end-effector of the robot, and the holes are fixed on the experimental table. The robot base coordinate is set as the base coordinate. The position (mm) and orientation (°) of pegs can be measured easily from robot controller.

The objective of the dual peg-in-hole assembly task is to push the pegs to the desired position in holes. This peg-in-hole assembly task mainly includes two main phases: search and

insertion. The proposed MDDPG algorithm is used to perform the insertion phase. The position of holes can be set by human and measured but with large positional error. During the search phase, the pegs are moved from a fixed initial position to align with the holes and then move down until the distance between the top of holes and the bottom of pegs is less than 0.5 mm. Then the pegs move down along $Z$-axis of the robot base coordinate with a small step size ($-0.06$ mm/step) and added a randomly selected constant noise in 6 components of the robot actions to learn the holes position with respect to the current pegs position. When the forces(N) and moments (Nm) can be measured and within the maximum range, it means that the search phase is performed successfully. Then the proposed MDDPG algorithm is utilized to accomplish the pushing process in insertion phase. The computer (Dell Precision M4800) communicates with the robot controller via TCP/IP protocol, which is developed by socket written in python. The force controller reads the forces/moments values from force-moment sensor according to the specific set of RAPID instructions. In this way, collecting the state data takes about 1.1s at each time step.

## B. Experiment Implementation

*1) Training process:* In the same way as in the simulation, the performances of the proposed fuzzy reward system and feedback exploration strategy are demonstrated through comparisons with the aforementioned approaches in realistic environments. In the dual peg-in-hole assembly experiments, the force-moment sensor should be calibrated first. The goal depth is set to 40 mm, and the maximum number of steps $K_{\max}$ is set to 65. To guarantee the safety of the components in realistic experiments, the maximum safe force is set to 80 N, the maximum safe moment is set to 5 Nm during the assembly process and $b$ is set to 0.2. Therefore, for one episode, the robot executes the assembly action step by step until the pegs reach the goal depth successfully or the assembly process is terminated when the measured forces-moments exceed the maximum safe boundary or the maximum number of steps is above 65.

The training hyper-parameters are selected to be the same as those shown in Table I. However, to make efficient use of the experience data, the training process runs for 50 times per episode. In our experiments, the precise positions of the dual holes are not required. Moreover, to test the robustness of the MDDPG algorithm against the initial position errors, before every assembly episode starting, the dual pegs move to their initial position from a fixed position because the robot has errors from repeated positioning.

As shown in Fig. 10(a), the agent with the accurate reward function only taking the assembly time into account can rapidly converge after training 200 episodes. However, the agent with the accurate reward function including both the assembly time and the contact forces cannot perform well because it is difficult to evaluate the assembly policies by balancing the performance of the assembly time and contact forces in the early training stage. However, as shown in Fig. 10(b), the agent utilizing the fuzzy reward system is able to converge and perform efficiently, similar to the accurate reward function only measuring the assembly time. As shown in Fig. 10(c) and (d), the agent with

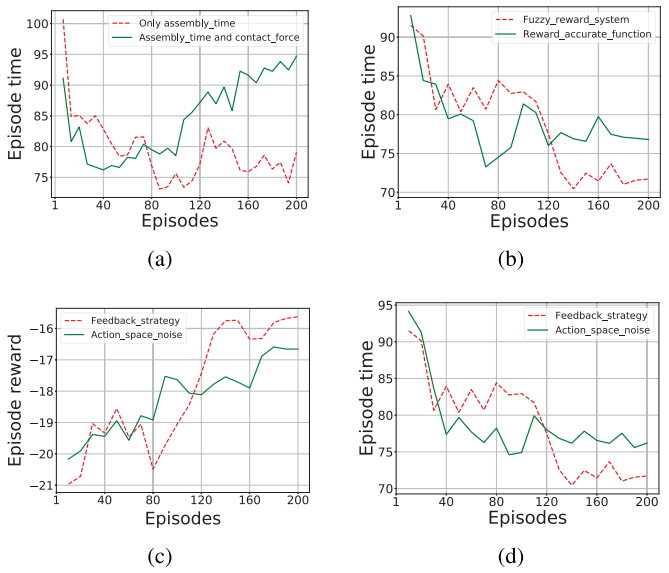


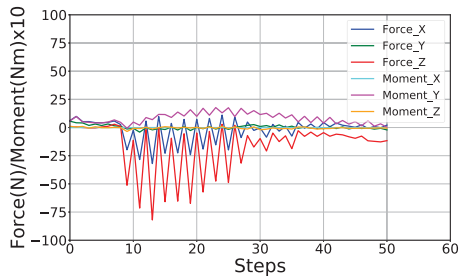

Fig. 10. Performance in realistic assembly experiments (the episode time only includes time of insertion phase). (a) Assembly time using two accurate reward functions. (b) Assembly time with fuzzy reward and accurate reward function. (c) Accumulative reward using feedback exploration strategy. (d) Assembly time using feedback exploration strategy.

the fuzzy reward system after training for only 200 episodes can decrease the assembly time by approximately 20 s, and the accumulative rewards increase by 20% in one episode.

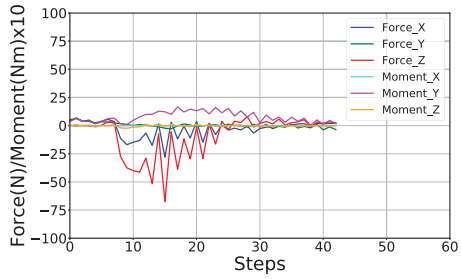
In addition, we compare two agents using the fuzzy reward system, but one agent only utilizes the OU noise in action space and another uses the feedback exploration strategy in Section IV. As shown, the agent with the feedback exploration strategy can perform efficiently. The proposed feedback exploration strategy only performs slightly better since the agent is only trained for approximately 200 episodes and there are fewer exploration episodes. However, the proposed exploration strategy can ensure that the exploration actions are safe and avoid risky exploration actions.

*2) Testing process:* To demonstrate the stability of the proposed algorithm, we execute the dual peg-in-hole robotic assembly task 50 times to test the success rate. In addition, we compare the changes in contact forces and moments during assembly processes accomplished by the learned policies with pretraining. The agent before training equals to the $P$ force controller and the proportional coefficient vector $\mathbf{K}_p$ ($[0.002, 0.002, 0.015, 0.01, 0.01, 0.01]$) is pretested referring to the work [3].

The proposed MDDPG algorithm using the fuzzy reward system and feedback exploration strategy achieves a 100% success rate in the case with the same initial position as the training process, which means that the initial position error is limited within 20 ($\mu$)m. Although the pretraining agent only by the $P$ force controller could also achieve the 100% success rate in our test experiments, we focus on demonstrating the improvement through training. As shown in Fig. 11, the agent after training can perform the same assembly task through fewer steps and decrease the contact forces and moments compared with pretraining.

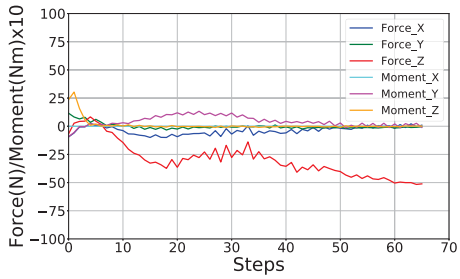



(a)

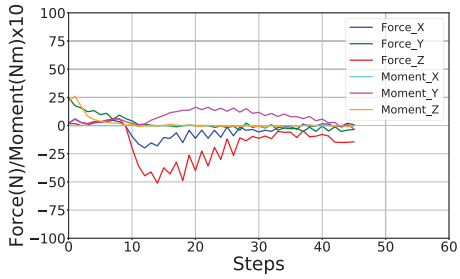



(b)

Fig. 11. Performance of tasks with the same initial position. (a) Performance before training. (b) Performance using the learned policy.




(a)




(b)

Fig. 12. Performance of tasks with new initial position. (a) Performance before training. (b) Performance using the learned policy.

Moreover, to demonstrate the robustness of the proposed algorithm against a new initial position, we rotate the pegs along $Z$-axis of the robot base coordinate by 1.0° and translate along $Y$-axis by 0.1 mm after getting the initial position in search phase. It is equivalent to that the holes are set in a new initial position. As shown in Fig. 12(a), the new initial position leads to a large initial contact forces and moments, and the agent before training only using the traditional $P$ force controller cannot finish the assembly process since it cannot decrease the contact forces $F_z$ and then exceeds the maximum steps in one episode. Nevertheless, the agent after training could accomplish the same task with the same initial position shown in Fig. 12(b). It means that the proposed algorithm can accomplish the assembly task through the learned high-level policy, which is robust to the new initial position.

Similar to the implementation in dual peg-in-hole assembly tasks, the MDDPG algorithm can also be applied in multiple peg-in-hole assembly scenarios, which cannot be accomplished successfully through the traditional force control algorithms relying on the analysis of the contact model. However, the agent with the learned assembly policy probably needs to be trained again in new assembly tasks (with different number of pegs or different size pegs) for learning the new environments. How long the agent should be trained depends on the complexity of the new task. For example, the agent needs to be trained longer in new tasks with number of pegs change than the tasks with only size change, because the peg-in-hole tasks with different number means the intrinsic change of the environment model. Although this is an important limitation of our paper, we believe that it is an appropriate and general way to address the complex multiple peg-in-hole assembly tasks in through the learned experience, which can greatly improve the efficiency of industrial manufacturing process without analyzing the unknown contact states and tuning the program parameters. In addition, importantly, the proposed MDDPG algorithm could become a general and practical approach to perform the different multiple peg-in-hole assembly tasks with the advance of transfer learning and state representation techniques.

## VII. Conclusion

Combined with a feedback exploration strategy and fuzzy reward system, the proposed MDDPG algorithm can perform multiple peg-in-hole assembly tasks well through the learned high-level assembly policy. The effectiveness and robustness of the proposed algorithm have been demonstrated through a designed simplified simulation model and realistic dual peg-in-hole assembly experiments. However, the contact force simulator cannot accurately estimate the contact forces, which means that the simulation training results cannot be applied in realistic experiments. Therefore, in future work, it is important that the agent is trained in a more accurate simulation environment implemented by virtual robot experimentation platform [21] or physics engine Mujoco [22]. The accurate environment could improve the data efficiency of the MDDPG algorithm because the learned policy could be applied in the realistic tasks directly or as the initial policy. Moreover, the learned policies applied to new assembly tasks still require retraining under the proposed algorithm, which limits its application scope. To address this challenge, the proposed algorithm will be improved by referring to the advantages of transfer learning.

## References

[1] A. Wan, J. Xu, H. Chen, S. Zhang, and K. Chen, "Optimal path planning and control of assembly robots for hard-measuring easy-deformation assemblies," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 4, pp. 1600–1609, Aug. 2017.

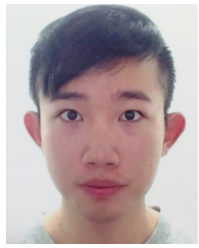
[2] Y. Fei and X. Zhao, "Contact and jamming analysis for three dimensional dual peg-in-hole mechanism," *Mechanism Mach. Theory*, vol. 39, no. 5, pp. 477–499, 2004.
[3] K. Zhang, M. Shi, J. Xu, F. Liu, and K. Chen, "Force control for a rigid dual peg-in-hole assembly," *Assem. Automat.*, vol. 37, no. 2, pp. 200–207, 2017.
[4] T. Tang, H.-C. Lin, and M. Tomizuka, "A learning-based framework for robot peg-hole-insertion," in *Proc, ASME 2015 Dyn. Syst. Control Conf.*, pp. V002T27A002–V002T27A002, 2015.
[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. Cambridge, MA, USA: MIT Press, 1998.
[6] C. Yang, C. Zeng, Y. Cong, N. Wang, and M. Wang, "A learning framework of adaptive manipulative skills from human to robot," *IEEE Trans. Ind. Inform.*, p. 1, 2018, doi: 10.1109/TII.2018.2826064.
[7] M. Nuttin and H. Van Brussel, "Learning the peg-into-hole assembly operation with a connectionist reinforcement technique," *Comput. Ind.*, vol. 33, no. 1, pp. 101–109, 1997.
[8] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 819–825.
[9] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015. [Online]. Available: http://arxiv.org/abs/1509.02971.
[10] 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, 2013.
[11] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, no. 4-5, pp. 421–436, 2018.
[12] L. Roveda, G. Pallucca, N. Pedrocchi, F. Braghin, and L. M. Tosatti, "Iterative learning procedure with reinforcement for high-accuracy force tracking in robotized tasks," *IEEE Trans. Ind. Inform.*, vol. 14, no. 4, pp. 1753–1763, Apr. 2018.
[13] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 2, no. 4, pp. 2096–2103, Oct. 2017.
[14] Y. Li, "Deep reinforcement learning: An overview," *CoRR*, vol. abs/1701.07274, 2017. [Online]. Available: http://arxiv.org/abs/1701.07274
[15] P. Kofinas, G. Vouros, and A. I. Dounis, "Energy management in solar microgrid via reinforcement learning using fuzzy reward," *Adv. Building Energy Res.*, vol. 12, no. 1, pp. 97–115, 2018.
[16] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: Variational information maximizing exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1109–1117.
[17] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.
[18] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," *CoRR*, vol. abs/1511.04143, 2015. [Online]. Available: http://arxiv.org/abs/1511.04143
[19] M. Khamassi, G. Velentzas, T. Tsitsimis, and C. Tzafestas, "Active exploration and parameterized reinforcement learning applied to a simulated human-robot interaction task," in *Proc. IEEE Int. Conf. Robot. Comput.*, 2017, pp. 28–35.
[20] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[21] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[22] V. Kumar and E. Todorov, "Mujoco haptix: A virtual reality system for hand manipulation," in *Proc. Humanoid Robots (Humanoids), IEEE-RAS 15th Int. Conf.*, 2015, pp. 657–663.

**Jing Xu** (M'12) received the B.E. degree in mechanical engineering from Harbin Institute of Technology, Harbin, China, in 2003, and the Ph.D. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2008.

He was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Michigan State University. He is currently an Associate Professor with the Department of Mechanical Engineering, Tsinghua University, Beijing, China. His research interests include vision-guided manufacturing, image processing, and intelligent robotics.

**Zhimin Hou** received the B.E. degree in mechanical engineering from Tongji University, Shanghai, China, in 2016. He is currently working toward the Master's degree in mechnical engineering at Tsinghua University, Beijing, China, advised by Prof. J. Xu.

He was a Visiting Scholar with the Department of Computing Science, University of Alberta, Edmonton, Canada, in 2018, advised by Prof. R. S. Sutton. His research interests include reinforcement learning, machine learning, robotic force control, and intelligent control.



**Wei Wang** received the B.E. degree in mechanical engineering from National Cheng Kung University, Tainan, Taiwan, in 2017. He is currently working toward the Master's degree in mechanical engineering from Tsinghua University, Beijing, China.

His research interests include force control, robotic vision, and intelligent robotics.

**Bohao Xu** received the B.E. degree in mechanical engineering from Yanshan University, Qinhuangdao, China, in 2015. He is currently working toward the Master's degree in mechanical engineering from Dalian Jiaotong University, Dalian, China.

He was a Visiting Student in mechanical engineering from Tsinghua University. Beijing, China, in 2018. His research interests include additive manufacturing, robotic vision, and intelligent robotics.

**Kuangen Zhang** received the B.E. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2016.

His research interests include force control, robotic vision, and intelligent robotics.

**Ken Chen** received the B.S. degree in mechanical engineering from Sichuan University, Chengdu, China, in 1982, and the M.S. and Ph.D. degrees in mechanical engineering from Zhejiang University, Hangzhou, China, in 1984 and 1987, respectively.

From 1991 to 1992, he was a Visiting Professor with the University of Illinois, Chicago. From 1992 to 1995, he was a Postdoctoral Researcher with Purdue University, Indianapolis, IN, USA. He is currently a Professor with the Department of Mechanical Engineering, Tsinghua University, Beijing, China. His research interests include robotics and intelligent control, humanoid robots, microrobots and small robots, medical and space robots, manufacturing automation systems, and hydraulic servo systems.