Learning-based Variable Compliance Control for Robotic Assembly

Tianyu Ren
ultrain@126.com
State Key Laboratory of Tribology, Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China

Yunfei Dong
d_yunfei@163.com
State Key Laboratory of Tribology, Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China

Dan Wu*,[1]
wud@tsinghua.edu.cn
State Key Laboratory of Tribology, Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China

Ken Chen
kenchen@tsinghua.edu.cn
State Key Laboratory of Tribology, Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China

Abstract
The assembly task is of major difficulty for manufacturing automation. Wherein the peg-in-hole problem represents a group of manipulation tasks that features continuous motion control in both unconstrained and constrained environments, so that it requires extremely careful consideration to perform with robots. In this work, we adapt the ideas underlying the success of human to manipulation tasks, variable compliance and learning, for robotic assembly. Based on sensing the interaction between the peg and hole, the proposed controller can switch the operation strategy between passive compliance and active regulation in continuous spaces, which outperforms the fixed compliance controllers. Experimental results show that the robot is able to learn a proper stiffness strategy along with the trajectory policy through trial and error. Further, this variable compliance policy proves robust to different initial states and it is able to generalize to more complex situation.

Keywords Control, Manufacturing, Novel Fabrication Techniques

Paper type Research paper

I. INTRODUCTION

High precision assembly is essential for manufacturing and so far, it is a labor-intensive industry with poor automation improvement. Through the problem has been extensively researched in fields of robotic and manufacturing, manual assembly is irreplaceable in many production lines. Most of the existing approaches highly depend on the high precision of robot manipulators with high stiffness and accurate sensory measurements. The vision-based methods seem intuitive and of high precision [1, 2], they are prone to cause hazardous assembly force even with tiny position error [3]. To constrain the contact force, force control algorithms are used for assembly. These methods, in general, use force sensors to detect external forces and moments, and design control strategies based on the mathematical model of the task dynamics [4-6]. In order to remove the hard work of representing the complex contact models, Inoue et al [7] proposed a robot skill acquisition approach by training a recurrent neural network with reinforcement learning. However, the generated discrete actions with high-stiffness constrain the smoothness and safety of the assembly process. While the remote center of compliance (RCC) device can regulate the stiffness of the robot end and may work well with specific insertion operation [8], it requires major mechanical adjustment for different objects.

For peg-in-hole, many works prove that a human is much better at this kind of contact motions [9]. A human baby, for example, can easily assemble LEGO blocks at the age of two with even no vision feedback and limited manipulation capacity. Human can achieve good manipulation performance through compliant control of their limbs with variable and task-dependent stiffness [10]. Yun [9] compared simulation results from various stiffness ranges of a robot manipulator and concluded that the passive compliance can greatly help with the peg-in-hole task and yield more stability in contact motion. However, finding the appropriate stiffness policy in accordance with the position policy for the manipulation task is a hard problem [11].

Reinforcement learning (RL) [12] is a possible solution to this problem with its important feature that can accomplish optimal performance without knowledge of the models of the robot &environment system [13]. With the RL algorithm of policy improvement with path integrals (PI^2), Buchli et al [14]

* Corresponding author.
E-mail address: wud@tsinghua.edu.cn (D. Wu).

[1] Postal address: Room A829, Lee Shau Kee Science and Technology Building, Tsinghua University, Beijing, 100084, China.

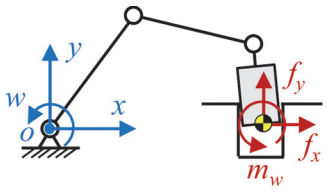

Fig. 1. Considered robotic assembly system. The peg-in-hole problem is implemented in the assembly plane *oxy*. *x*, *y*, *w* present the positive directions of the robot motion (rotation), while f_x, f_y, m_w present the positive total external force (moment).

and Abu-Dakka et al [15] accomplished variable stiffness skill learning on robot manipulators. The strategies of both the reference trajectory and the stiffness of the robot are generated by a planner from a stochastic optimal process with path integrals. They are successfully implemented in via-point tracking and several simple tasks. In other works, this method was used to learn other compliant skills including pool stroke [16] and dynamic pancake-flipping task [17]. Though the policy is updated episode by episode, it remains unchanged in each episode and it is not able to actively react to perturbations. In previous studies, we tried to apply the PI^2 algorithm for variable stiffness assembly and found that the robot is always stuck in the hole even with substantial training. This is mainly results from the repeatability positional error of the robot compared to the small assembly tolerance. The trajectory generated by the planner can never be implemented accurately by the robot.

In insertion tasks, we expect the robot to learn how to react to contact forces by variable stiffness motion. However, the non-linear function approximators in RL algorithms means that convergence is no longer guaranteed in theory and the stability in practical learning is very problematic. Based on Deep Q Network (DQN) [18] and deterministic policy gradient (DPG) [19], Lillicrap et al [20] presented Deep DPG (DDPG), an actor-critic, model-free RL algorithm that can operate in continuous action spaces, which is especially appealing for learning the interaction strategy between the robot and environments.

The goal of this research is to incorporate the variable compliance control and deep reinforcement learning as a learning-based controller as human for robotic assembly, and to show the importance of the adjustable stiffness in peg-in-hole tasks. The rest of this paper is structured as follows. Section II firstly formulates the peg-in-hole problem and analyzes the help the compliant motion strategies to this task. Section III briefly reviews the used RL algorithm and further explains how it is applied to variable compliance assembly. Section IV presents the experimental evaluation of our methods on a 7-DOF robot. Finally, Section V concludes this research and addresses future works.

II. Insertion with Compliant Control

A. Peg-in-hole insertion task

Peg-in-hole is the benchmark problem in the category of insertion task and it represents a batch of challenges in robot manipulation. It is generally divided into two phases: a) search phase and b) insertion phase [21]. In the first phase, the robot need to locate the hole with the peg. While in the second phase, the task becomes more complicated. The robot has to align the axis of the peg to that of the hole and insert the peg to the desired depth. During this phase, slight misalignment will lead to high friction and even result in jamming. In general, the potential friction and jamming is hard to model, and therefore it is unlikely to find optimal policies with model-based methods. This study focused on the more challenging insertion phase where force control is explicitly required.

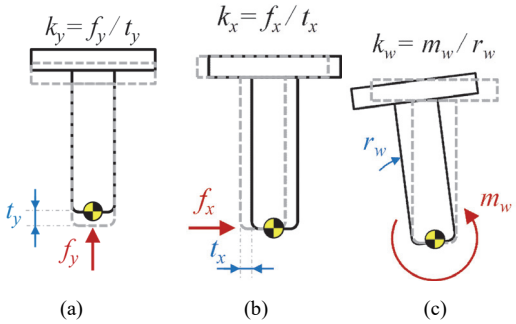

Fig. 2. Passive response of the peg to external forces and moments, and the definition of stiffness in three directions. (a) Response to external vertical forces. (b) Response to external lateral forces. (c) Response to external moments.

In high precision assembly, the clearance between the peg and hole is generally smaller than the uncertainty of position control so that neither traditional kinematic planning nor position control can undertake this task. In the studies of assembly process, the peg motion is usually decoupled into two orthogonal planes for analysis [4-9, 18]. Therefore, without losing generality, a 2D peg-in-hole problem is considered in this work, wherein a square peg and a hole are implemented in the assembly plane *oxy* (Fig. 1). A 3-DOF robot manipulator grasps the peg and interacts with the hole. Based on the feedback of state, a state-action controller is designed to undertake the assembly task.

B. Cartesian Compliant Control

In order to improve positioning accuracy, classical robot controllers use position feedback control with high gains. Unfortunately, high-gain control is not favorable for many manipulations where sophisticated physical interaction is included [14]. In contrast, impedance control [22] is one of the most popular approaches to Cartesian compliance control [23] and it seeks to maintain a mass-damper-spring relationship between the external force $\boldsymbol{f}$ and position displacement $\Delta\boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{x}$ in Cartesian space:

$$\boldsymbol{f} = \boldsymbol{M}\Delta\ddot{\boldsymbol{x}} + \boldsymbol{D}\Delta\dot{\boldsymbol{x}} + \boldsymbol{K}\Delta\boldsymbol{x}, \quad (1)$$

where $\Delta\boldsymbol{x}$ represents the difference between the actual position $\boldsymbol{x}$ and the moving reference point $\boldsymbol{x}_d$ called the *virtual trajectory*. $\boldsymbol{M}, \boldsymbol{D}, \boldsymbol{K}$ are the positive-definite virtual inertia, damping and stiffness matrices.

C. Insertion with Cartesian compliant control

For insertion tasks, the robot is generally supposed to move slowly and smoothly in the constraint environment so that the

operation can be regarded as a quasi-static process. The terms related to acceleration and velocity in Eq. (1) are relatively insignificant. Consequently, a stiffness controller instead of the classical impedance controller is compatible for this task. When the compliant robot is blocked by obstacles, it will deviate from the virtual trajectory and tends to be pulled back according to the designated stiffness. Stiffness selection for insertion is widely discussed in previous works and one of the desired strategies was proposed in [24] based on model analysis. It is obvious that by designating a low stiffness in *x*-direction and a low torsional stiffness in *w*-direction, the peg will easily make lateral displacement in response to lateral forces f_x and rotational displacement m_w in response to moments. In fact, this strategy is the basis of RCC design where it is realized by passive compliance mechanism. The idea of RCC shows the importance of selective compliance for specific tasks. While suitable stiffness characteristics seems partly derivable, it is not easy to generalize the compliant control to more complex situations, such as significant orientation misalignment and jamming. So the problem of specifying the target stiffness has not been completely addressed.

III. Learning-based Variable Compliance Controller

A. State-action controller design for insertion task

Compared to the model-based method with sophisticated analysis on the environment, a RL-based controller is much more preferred to find optimal policies through trial and error. We consider the insertion task as an optimal control problem where a insertion controller is used to map states $\boldsymbol{s} \in \mathbb{R}^{D_s}$ observed from environments to a specific robot action $\boldsymbol{a} \in \mathbb{R}^{D_a}$ with deterministic virtual trajectory and stiffness. This state-action process is assumed to have the Markov property characterized as "memorylessness".

In assembly tasks, historical information is indispensable for the agent to understand current situation. For example, it cannot distinguish jamming from a normal insertion with large contact force, based only on one-shot feedback. To address this problem, we need to design the observation state with proper historical information according to the assembly process. During assembly, the robot moves in directions of $\boldsymbol{x} = (x, y, w)$ and senses the generalized external force $\boldsymbol{f} = (f_x, f_y, m_w)$. Each episode starts from a designated starting position which is nearby the hole. In time step t, the insertion controller observes the current state $\boldsymbol{s}_t$ of the system and generate action $\boldsymbol{a}_t$ to react to this observation. The definition of $\boldsymbol{s}_t$ is inspired by the classic Proportional-differential (PD) controller:

$$\boldsymbol{s}_t = (\boldsymbol{f}_t, \Delta\boldsymbol{x}, \Delta\boldsymbol{f}, progress), \tag{2}$$

where $\Delta\boldsymbol{x} = \boldsymbol{x}_t - \boldsymbol{x}_{t-1}$ and $\Delta\boldsymbol{f} = \boldsymbol{f}_t - \boldsymbol{f}_{t-1}$ are the incremental displacement and incremental force respectively. They are used to infer if the peg is jammed in a certain direction, which are combined to define a *jamming indicator*. Instead of explicit position information, we include the percentage of insertion $progress = depth_{insertion} / depth_{hole}$ in the state vector. $progress = 1$ means that the peg is completely pushed onto the hole bottom, which is considered as a success of this trial. In robotic assembly, contact force must be strictly limited to avoid robot overload and workpiece damage. Any force overload will trigger the emergency stop of the robot in operation, and current episode will be labeled as a failure.

$$\begin{cases} \boldsymbol{f}^{upperLmt} = (f_x^{Lmt}, f_y^{Lmt}, m_w^{Lmt}) \\ \boldsymbol{f}^{lowerLmt} = (-f_x^{Lmt}, -f_y^{Lmt}, -m_w^{Lmt}) \end{cases}. \tag{3}$$

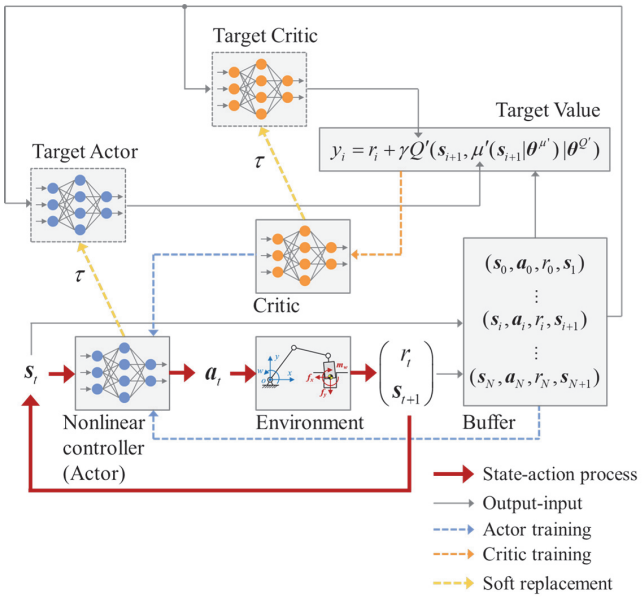

Fig. 3. Schematic illustration of the insertion controller, and its training process with DDPG. Wherein, the proposed controller is implemented by neural networks termed actor in actor-critic methods.

In order to realize variable compliance control, the action $\boldsymbol{a}_t$ generated by insertion controller is defined as a combination of elementary movements and designated stiffness:

$$\boldsymbol{a}_t = (\Delta\boldsymbol{x}_d, \boldsymbol{k}), \tag{4}$$

wherein $\Delta\boldsymbol{x}_d = (\Delta x_d, \Delta y_d, \Delta w_d)$ is the increment of the virtual trajectory, and $\boldsymbol{k} = (k_x, k_y, k_w)$ presents the visual Cartesian stiffness of the robot. For safety and stability, $\boldsymbol{a}_t$ is limited to a specific interval.

$$\begin{cases} \boldsymbol{a}^{upperLmt} = ((\Delta x_d^{Lmt}, \Delta y_d^{Lmt}, \Delta w_d^{Lmt}), (k_x^{Lmt}, k_y^{Lmt}, k_w^{Lmt})) \\ \boldsymbol{a}^{lowerLmt} = ((-\Delta x_d^{Lmt}, -\Delta y_d^{Lmt}, -\Delta w_d^{Lmt}), (0, 0, 0)) \end{cases}. \tag{5}$$

To explore new control policies, we add exploration noise $\boldsymbol{\delta} \in \mathbb{R}^{D_a}$ to the action independently from the learning algorithm:

$$\boldsymbol{a}_t = \boldsymbol{a}_t + \boldsymbol{\delta}. \tag{6}$$

After the execution of $\boldsymbol{a}_t$, a reward r_t from the environment is provided to insertion controller for parameter training. Then the system moves to a next state $\boldsymbol{s}_{t+1}$. According to the desired

Algorithm 1 Network training with DDPG

Initialize actor and critic network with random weights.
Initialize two target networks with the same weights.
Initialize replay buffer R with a maximum size in a FIFO manner.
for episode=1 **to** M **do**
  Reset the environment and receive initial observation $\boldsymbol{s}_1$.
  for step=1 **to** T **do**
    Choose action $\boldsymbol{a}_t$ and add exploration noise. (see (5)(6))
    Execute the action and receive reward r_t; get next state $\boldsymbol{s}_{t+1}$.
    Store transition $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ in R.
    Update four networks. (see (10)(11)(12)(9))
  end for
end for

behavior in the insertion task, we define the reward function as follows:

$$r = -w_{dis}\frac{\Delta y}{\Delta y_d^{Lmt}} - w_f\left(\left\|\frac{\boldsymbol{f}}{\boldsymbol{f}^{upperLmt}}\right\|_\infty\right)^2 - w_{stf}\left(\left\|\frac{\boldsymbol{k}}{\boldsymbol{k}^{upperLmt}}\right\|_\infty\right)^2 + r_{end} \quad (7)$$

The first three items represent the immediate rewards in each step that are respectively used to punish the displacement away from the hole, large contact force and high stiffness. We can regulate the relative weights of these terms by adjusting w_{dis}, w_f, w_{stf}. The starting point of the reward design is to encourage downward movement of the peg and meanwhile discourage actions that will cause large contact forces or require high stiffness motion. If the trial succeeds or falls before reaching the maximum number of step, r_{end} is provided to the network in the last step of this episode.

$$\begin{cases} r_{end} = 1, & \text{if the trial is successful.} \\ r_{end} = -1, & \text{if the trial is failed.} \end{cases} \quad (8)$$

The state-action process for the insertion controller is shown as a closed loop in Fig. 3. Besides projecting states to actions, the controller is supposed to improve itself during task execution. Here an actor-critic architecture [25] is considered for simultaneous execution and improvement. A neural network $\mu(\boldsymbol{s}|\boldsymbol{\theta}^\mu)$ termed as *actor* specifying the current policy implements the insertion controller. For improvement, a *critic* network $Q(\boldsymbol{s},\boldsymbol{a}|\boldsymbol{\theta}^Q)$ is used to evaluate and update the actor. Wherein, $\boldsymbol{\theta}^\mu$ and $\boldsymbol{\theta}^Q$ are respectively the parameters of the actor and critic to be updated. Finally, a learning-based insertion controller with continuous state-action spaces is established.

B. Learning with DDPG

To update the insertion controller online in the practical system, we need a stable and efficient learnig algorithm. DDPG combines the advantage of DQN and DPG such that it is able to learn the value functions with nonlinear function approximators in a stable and robust way, which is an important stride for reinforcement learning. This algorithm is made up of a straightforward actor-critic architecture, making it easy to implement. In order to learn across a set of uncorrelated transitions, DDPG uses a replay buffer to store transitions $(\boldsymbol{s}_i,\boldsymbol{a}_i,r_i,\boldsymbol{s}_{i+1})$ sampled from the training process and update the actor and critic at each timestep by sampling a minibatch uniformly from the buffer. In classical Q learning methods, the Q update is prone to divergence since the network being updated is also used to calculate the target value. Here, a copy of the actor and critic networks, $\mu'(\boldsymbol{s}|\boldsymbol{\theta}^{\mu'})$ and $Q'(\boldsymbol{s},\boldsymbol{a}|\boldsymbol{\theta}^{Q'})$, are created and used for calculating the target value. The parameters of these target networks are "softly" updated by:

$$\boldsymbol{\theta}' = \tau\boldsymbol{\theta} + (1-\tau)\boldsymbol{\theta}' \quad \text{with } \tau \ll 1 \text{ is the learning rate.} \quad (9)$$

The idea of minibatch training and soft replacement significantly improves the efficiency and stability of learning. During training, the critic network parameterized by $\boldsymbol{\theta}^Q$ is optimized by minimizing the loss function:

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2), \quad (10)$$

where

$$y_i = r_i + \gamma Q'(\boldsymbol{s}_{i+1}, \mu'(\boldsymbol{s}_{i+1}|\boldsymbol{\theta}^{\mu'})|\boldsymbol{\theta}^{Q'}). \quad (11)$$

γ is the discount factor. The training data set used in (10) and (11) is sampled from a random minibatch of N transitions from the replay buffer R. The actor is then updated by using the sampled gradient with respect to its network parameters $\boldsymbol{\theta}^\mu$:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}. \quad (12)$$

In the same training step, target networks are updated through soft replacement described in (9). The algorithm used for variable compliance skill learning is summarized in Algorithm 1.

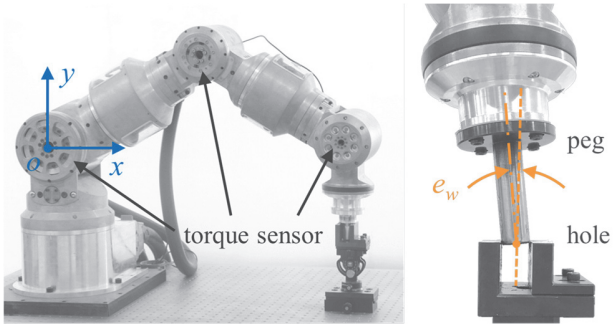

Fig. 4. Description of the robot and peg-in-hole components. Torque sensors integrated in the robot joints are used to estimate contact force and moment in Cartesian space. The initial angle error e_w is defined as the angle between the axes of the peg and hole in the assembly plane in staring position of the task.

IV. Experiments and Analysis

A. System description

The experimental system including a 7-DOF torque-controlled robot and peg-in-hole workpieces as shown in Fig. 4. Each joint of the robot is integrated with a torque sensor so that the controller is able to estimate Cartesian contact force (f_x, f_y, m_w) by force Jacobian [26, 27]. Similarly, the position and orientation of the peg (x, y, w) is calculated though

TABLE I
SPECIFIC PARAMETERS OF THE LEARNING AGENT

Parameter	Value
γ	0.99
τ	0.01
$(f_x^{Lmt}, f_y^{Lmt}, m_w^{Lmt})$	(40 N, 40N, 5 Nm)
$(\Delta x_d^{Lmt}, \Delta y_d^{Lmt}, \Delta w_d^{Lmt})$	(1 mm, 1 mm, 3°)
$(k_x^{Lmt}, k_y^{Lmt}, k_w^{Lmt})$	(4000 N/m, 4000 N/m, 200 Nm/rad)
w_{dis}	1
w_f	1
w_{stf}	0.1

forward kinematics. For controlling the robot, we use a Cartesian impedance controller without inertia shaping [22]. The position error of the robot is about ±0.5 mm in translation and ±0.5° in rotation. The 2D hole is implemented by an adjustable slot with two parallel walls perpendicular to the assembly plane *oxy* (Fig. 4). Both the hole and peg are made of steel with different sizes. Their diameters (or width) are D=23.04 mm and d=23 mm, respectively. The depth of the hole is H=36 mm. The peg is machined with a fillet of 0.5 mm to avoid serious scratch during exploration.

The learning-based insertion controller is implemented on a PC (CPU 3.5 GHz, RAM 32GB) communicating with the robot controller by Socket. The PC receives observations from the robot, trains the networks, and sends actions to the robot controller in every 500 ms. This cycle time is selected though trails in experiments to ensure a quasi-static operation. The neural networks of the actor and critic share the same structure of two fully-connected hidden layers with 64 units. Adam algorithm [28] is used to update the parameters with the learning rates of 1×10^{-3} and 3×10^{-4} for the actor and critic respectively. The memory capacity of the replay buffer is 3000 and the size of the minibatch is 128. Other parameters of the learning algorithm are shown in Table I.

B. Training

In the training phase, the robot starts insertion from a constant position that is located exactly at the top of the hole with the peg tip partly sinking into the hole to avoid the searching operation. In order to learn a robust skill in different situations and further acquire a general policy, initial orientation error with a uniform distribution U(-3°, 3°) is introduced to the initial state. For exploration in each step, we add noise $\boldsymbol{\delta}$ sampled from a normal distribution with standard deviation of 10% of the action range to the actor output.

During exploration, the contact force is restricted by Eq. (3), which is far below the rated load of the peg-in-hole system. In order to further reduce the wear of workpieces, we consider it is a success for peg insertion if $progress > 0.97$. The learned policy is evaluate periodically following five training episodes

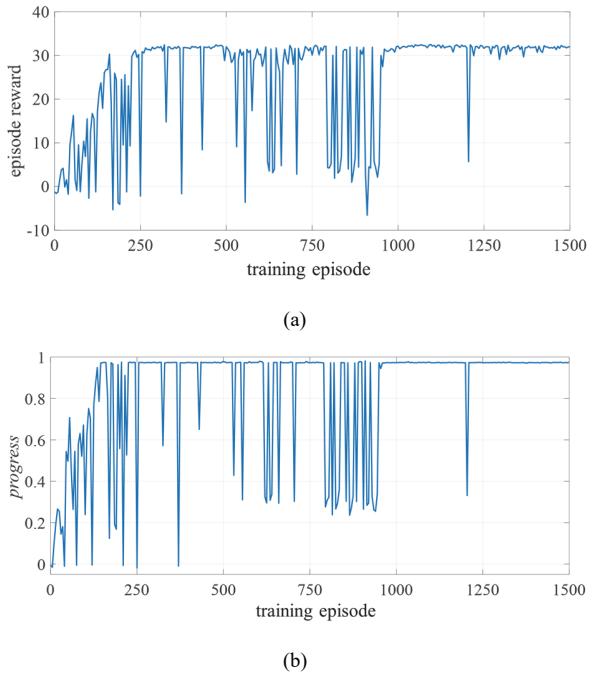

Fig. 5. Learning curve of the neural controller. (a) Cumulative reward of an episode. (b) Achieved insertion *progress*.

by testing it without exploration noise. Fig. 5 shows the learning progress that takes about 500 episodes for reward convergence. The reward curve converges exponentially with occasional fluctuation resulting from action exploration, which presents a typical profile of RL algorithms. In order to acquire a more general knowledge, we train the networks for 1500 episodes. The time consuming is about 9 hours. Comparing Fig. 5 (a) and (b), it is clear that the reward is highly related to the insertion *process* as the final depth is the decisive factor for task success and thus has the highest weight.

C. Evaluation

The trained controller is firstly tested with a simple peg-in-hole task without introducing orientation error. During the evaluation, the observations and actions are recorded and drawn in Fig. 6. It shows that the robot achieves a smooth insertion process and complete the task in 35 steps that represents the top performance with consideration of the step limit Δy_d^{Lmt} (Fig. 6 (i)). The learned stiffness policy is quite intuitive: in the direction of insertion, k_y is set to the maximum stiffness to ensure enough impetus, while motions in other directions are unconstrained by setting the stiffness to zero (Fig. 6 (iv)), which is exactly the idea behind RCC. In this case, the robot loses two degrees of freedom in x and w. The only enabled motion is in y-direction where the virtual trajectory is always heading to the bottom of the hole (see Δy_d in Fig. 6 (iii)).

In the constrained motion of insertion, due to the robot position error, the peg always deviates from its virtual trajectory while its displacement is restricted by the hole. The selective compliance makes the peg yield to the lateral forces before generating large contact force (Fig. 6 (ii)). Note that the rear

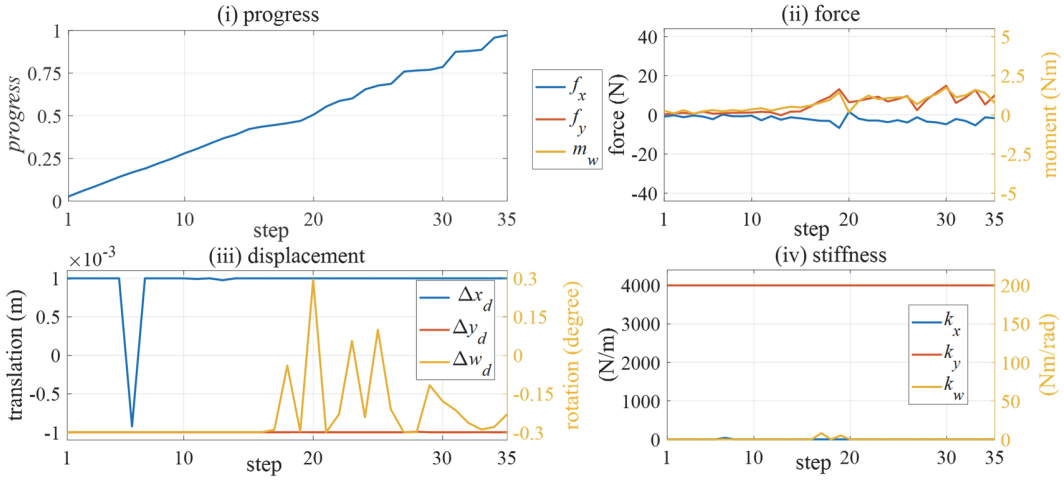

Fig. 6. Insertion process conducted by the proposed controller with e_w=0°. (i) Observation: insertion progress. (ii) Observation: Contact force and moment. (iii) Action: incremental virtual trajectory. (iv) Action: stiffness.

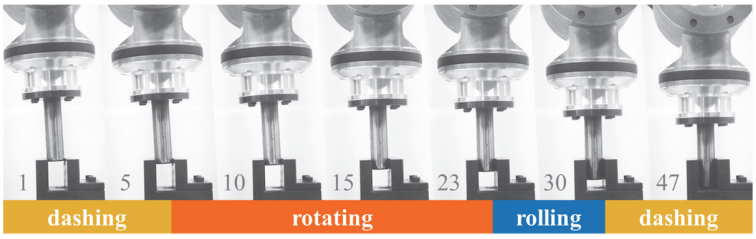

Fig. 7. Stages of the assembly process with large initial angle error. Corresponding step number is marked at the bottom of the picture.

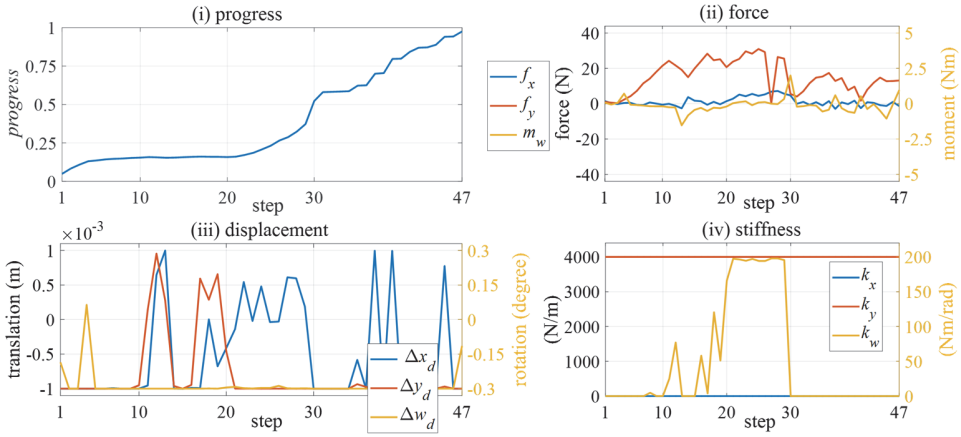

Fig. 8. Insertion process conducted by the proposed controller with e_w=8°. (i) Observation: insertion progress. (ii) Observation: Contact force and moment. (iii) Action: incremental virtual trajectory. (iv) Action: stiffness.

segments of the curve in Fig. 6 (i) shows a stairs shape. This is caused by the nonlinear stiction that is periodically overcome by the robot.

To test the robustness of the learned insertion controller, we perform insertion experiments with different starting orientation error from 1° to 10°. For each e_w, we evaluate the controller performance in 10 episodes and count its success rate. The results are given in Table II. It is clear that the learned controller is able to deal with angle error of no more than 8°, and has some chance to succeed even when $e_w = 9°$. It outperforms the insertion controller presented in [7] that also based on deep reinforcement learning, in terms of the maximum allowable angle error (1.6°). Fig. 7 shows the assembly process with an initial angle error of 8°. It reveals a sophisticated insertion strategy that can be summarized in 4 stages:

1) Dashing. Before step 5, the peg dashes down to the hole bottom until it get stuck. This event can be inferred though the stagnation of the insertion *progress* (Fig. 8 (i)) combined with the continuous increasement of the contact force f_y (Fig. 8 (ii)).

2) Rotating. From step 5 to step 23, the robot does not make

TABLE II
SUCCESS RATE WITH DIFFERENT INITIAL ORIENTATION

e_w	Variable compliance	High compliance	Low compliance	Selective compliance
0°	1.0	1.0	1.0	1.0
1°	1.0	1.0	1.0	1.0
2°	1.0	1.0	0.7	1.0
3°	1.0	0.8	0.0	1.0
4°	1.0	0.2	-	0.4
5°	1.0	0.0	-	0.0
6°	1.0	-	-	-
7°	1.0	-	-	-
8°	1.0	-	-	-
9°	0.2	-	-	-
10°	0.0	-	-	-

TABLE III
STIFFNESS SELECTION OF DIFFERENT COMPLIANCE CONTROLLER

k	Variable compliance	High compliance	Low compliance	Selective compliance
k_x	[0, 4000]	200	4000	200
k_y	[0, 4000]	200	4000	4000
k_w	[0, 200]	10	200	10

much progress in the direction of insertion (see Fig. 7 and Fig. 8 (i)), instead, it rotate the peg with an effective torsional stiffness $k_w > 0$. In the beginning of this phase, the orientation misalign results in an increasing moment as the peg move downwards (see m_w during step 5-12 in Fig. 8 (ii)). Then it is corrected by the active adjustment. During this process, the rotation is periodically enable to avoid overcorrection, which results in several repeated changes in k_w (see Fig. 8 (iv)). At the same time, the insertion motion is halted and even reversed by the insertion controller to avoid further force growth in y-direction (see f_y and Δy_d respectively in Fig. 8 (ii) and (iii)).

3) Rolling. During step 23 to step 30, the peg continues to rotate, however, it makes significant progress in insertion at the same time (see Fig. 7). In this phase, rotation in w-direction is continuously enabled by maximizing the torsional stiffness k_w (Fig. 8 (iv)). This active rotation at beginning plays a positive role for overcoming the external moment and further correcting the orientation of the peg (Fig. 9 (a)). As the angle error reduces to specific value, the equivalent insertion force $f_{eq} \in [-f_y^{Lmt}, f_y^{Lmt}]$ is able to overcome the total external force f_{ext}, and consequently, the peg slides to a deeper positon until getting jammed again. At this time, however, the peg lean on the hole wall with the other side, resulting in a reversed external moment (see m_w during step 28-30 in Fig. 8 (ii)). In response to this overcorrection, the controller reduces k_w to zero and make the peg compliant in w-direction again. This process is

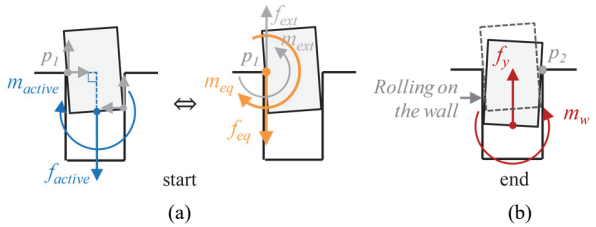

Fig. 9. Illustration of the rolling phase. (a) The Starting state. The robot active force and moment on the left are equivalent to the force and moment acting on the upper contact point p_1. (b)The terminate state. The peg rotates and slides downwards until jamming in a deeper position. p_2 is the new contact point.

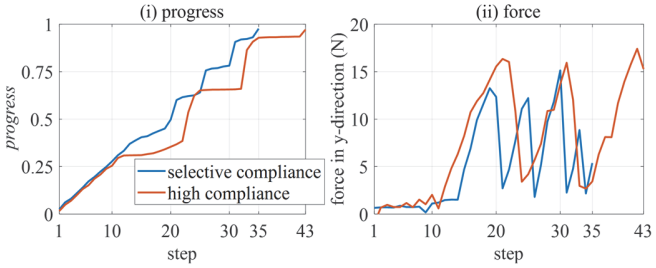

Fig. 10. Performance comparison of the controllers with selective compliance and uniformly high compliance. (i) Observation: insertion progress. (ii) Observation: Contact force in y-direction.

described as the rolling of the peg on the hole wall.

4) Dashing. In the final stage, misalignment of the peg and hole is fully compensated, and the robot complete the insertion task in the same way as the last test without angle error.

With the consideration that no explicit model information and control law is given, the insertion controller do obtain a list of useful policies though reinforcement learning to deal with the complex situations. These policies, if not best, are quite effective. The implementation of the sophisticated active correction, rotating and rolling, presupposes the awareness of jamming during task execution. It is depending on the jamming indicator that the controller can switch from passive compliance to active correction, or vice versa. Depending on the initial state of orientation error, the required number of execution step varies from 34 to 50, which implies an execution time of 17 s to 25 s per insertion. With a larger angle error, the number of execution step will become larger as the controller needs more steps to adjust the peg and align it with the hole.

D. Comparison with fixed compliant control

To illustrate the importance of variable compliance in peg-in-hole task, we also report results with the stiffness regulation removed. To this end, another three insertion controllers with fixed high compliance, low compliance, and selective compliance are designed (Table III). They share the same observation states, rewards, and limitations with the proposed variable compliance controller, but a simplified action vector containing only elementary movements. Each of these controllers is trained with DDPG until convergence, and continuously updated to 1500 episodes for skill generalization. The learned controllers are evaluated with a list of initial angle error. Their success rate in 10 episodes is given in Table III alongside with the variable compliance controller.

The low compliance (high stiffness) controller no doubt has the weakest robustness as it cannot exploit passive compliance to react to misalignment or other disturbances, and it tends to cause large contact force with a single displacement. On the other hand, the controllers with high compliance and selective compliance are able to adjust the peg orientation passively under the external force. The differences between these two controllers are minor. The reason of the higher success rate of selective compliance than that of the uniform compliance is that with a higher stiffness in the insertion direction, the robot is able to generate larger active forces in limited steps, and resolutely get rid of the stiction and friction before being jammed for a long period. Fig. 10 shows the advantages of selective compliance in terms of insertion progress and contact force. Besides accelerating the insertion progress, the high stiffness in *y*-direction helps to restrict the derivation between the virtual trajectory and actual position, and therefore reduces the effect of stiction lag. In fact, the controller with selective compliance takes the advantage of model analysis, revealing similar performance as RCC devices. This comparison indicates that a comprehensive stiffness strategy should not always present passive compliance in the presence of external force, but also bring sufficient support to the robot motion in certain directions to overcome resistance.

Though the selective compliance controller proves the best performance in constant compliance controllers (see Table III), it is eclipsed by the variable compliance controller. The controller with variable compliance is able to change the behavior of the robot between passive compliance and active regulation in continuous spaces, where the designated stiffness can be recognized as the activation level of active motion. In the dashing stage, the robot mainly shows passive properties and tries to exploit the constraint force for alignment. While in the stage of rotating and rolling, the robot actively correct the orientation of the peg based on force sensing with high stiffness motion and therefore makes its way towards deeper position. These well-timed active regulations in the insertion process are critical for the superior robustness of the proposed controller.

E. Generalization to insertion with deformable workpiece

To evaluate the proposed skill learning method in more challenging environments, we apply the trained controller to the insertion problem with a deformable peg. This is composed of the previously used peg and a rubber sleeve pipe with a maximum outside diameter of 29 mm (Fig. 11). The hole is adjusted to the same size of the deformable peg. From the second column of Table IV， it is shown that the controller learned for stiff workpiece does have some chance to successfully insert the deformable peg into the hole. However, the significant difference of between the two models prevents the controller from giving a satisfactory performance. The contact stiffness of the deformable peg is much weaker than the stiffness one, which can be characterized by Young's modulus of the two material: E_{rbr}= 2.8×10^6 Pa for rubber and E_{stl}= 2.1×10^{11} Pa for steel. So that for the same displacement deviation during insertion, compared to the stiffness peg, the deformable workpiece will produce minor contact force that can be used for feedback control. Moreover, the texture and friction coefficient of the peg surface is quite different between the two workpieces.

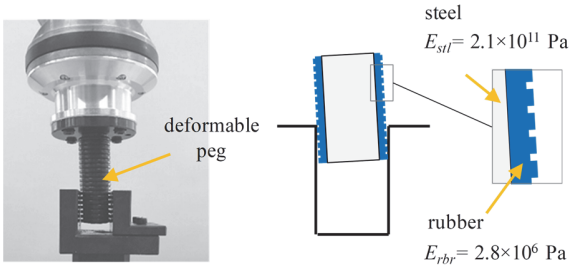

Fig. 11. Description of the deformable peg for the insertion skill.

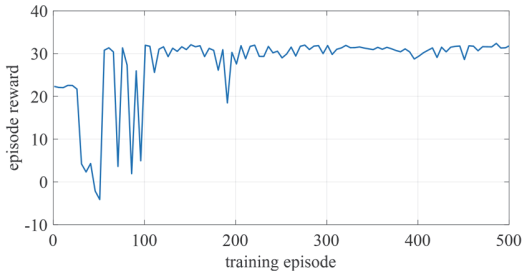

Fig. 12. Learning curve of transfer learning for deformable workpiece.

TABLE IV
SUCCESS RATE WITH DIFFERENT INITIAL ORIENTATION

e_w	Before training	After training
0°	0.5	1.0
1°	0.2	1.0
2°	0.0	1.0
3°	-	1.0
4°	-	1.0
5°	-	0.9
6°	-	0.6
7°	-	0.6
8°	-	0.3
9°	-	0.0

Based on the insight of transfer learning, we generate our controller across tasks by initializing the network parameters for the deformable peg-in-hole task with the learned parameter in the stiffness situation. As shown the Fig. 12, the training reward converges quickly in 3.5 hours due to the generalization ability of the proposed controller. Finally, the robustness of the controller to orientation error is evaluated and the results are given in the third column of Table IV. It is clear that the controller after training works fairly well with the new model, which implies a good flexibility of the proposed method.

V. Conclusion

In this work, we model the peg-in-hole task as a Markov decision process with a continuous state-action space and thus present a robust model-free controller with variable compliance to govern the insertion process in a closed loop. For controller updating, we employ the recent advances in deep reinforcement learning to ensure stability and efficiency.

The two innovations of our controller, the jamming indicator in observation and variable compliance in action, significantly improves the efficient and robustness of task execution. Experimental results demonstrate a much better performance of proposed controller than the selective compliance controller based on model analysis does.

One of the most notably limitation of is that the learning algorithm requires a large number training episodes. In further works, we will include heuristic knowledge from human manipulation into action selection and exploration to accelerate the convergence rate of controller parameters.

References

[1] Chang, R. J., Lin, C. Y., and Lin, P. S., 2011, "Visual-Based Automation of Peg-in-Hole Microassembly Process," Journal of Manufacturing Science & Engineering, 133(4), p. 041015.

[2] Yoshimi, B. H. and Allen, P. K., "Active, uncalibrated visual servoing," Proceedings of the IEEE International Conference on Robotics and Automation, 1994. Proceedings, 1994, pp. 156-161 vol.1.

[3] Burdet, E. and Nuttin, M., 1999, "Learning Complex Tasks Using a Stepwise Approach," Journal of Intelligent & Robotic Systems, 24(1), pp. 43-68.

[4] Usubamatov, R. and Leong, K. W., 2011, "Analyses of peg - hole jamming in automatic assembly machines," Assembly Automation, 31(4), pp. 358-362.

[5] Jasim, I. F. and Plapper, P. W., 2014, "Contact-state monitoring of force-guided robotic assembly tasks using expectation maximization-based Gaussian mixtures models," International Journal of Advanced Manufacturing Technology, 73(5-8), pp. 623-633.

[6] Zhang, K., Shi, M. H., Xu, J., Liu, F., and Chen, K., 2017, "Force control for a rigid dual peg-in-hole assembly," Assembly Automation, 37(2), pp. 200-207.

[7] Inoue, T., Magistris, G. D., Munawar, A., Yokoya, T., and Tachibana, R., 2017, "Deep Reinforcement Learning for High Precision Assembly Tasks."

[8] Whitney, D. E., 1982, "Quasi-Static Assembly of Compliantly Supported Rigid Parts," Journal of Dynamic Systems Measurement & Control, 104(104), pp. 65-77.

[9] Yun, S. K., "Compliant manipulation for peg-in-hole: Is passive compliance a key to learn contact motion?," Proceedings of the IEEE International Conference on Robotics and Automation, 2008, pp. 1647-1652.

[10] Ganesh, G., Albu-Schäffer, A., Haruno, M., Kawato, M., and Burdet, E., "Biomimetic motor behavior for simultaneous adaptation of force, impedance and trajectory in interaction tasks," Proceedings of the IEEE International Conference on Robotics and Automation, 2010, pp. 2705-2711.

[11] Franklin, D. W. *et al.*, 2008, "CNS learns stable, accurate, and efficient movements using a simple algorithm," Journal of Neuroscience the Official Journal of the Society for Neuroscience, 28(44), pp. 11165-73.

[12] Arnold, B., 1998, "Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)," IEEE Transactions on Neural Networks, 9(5), p. 1054.

[13] Kober, J. and Peters, J., 2013, "Reinforcement Learning in Robotics: A Survey," International Journal of Robotics Research, 32(11), pp. 1238-1274.

[14] Buchli, J., Stulp, F., Theodorou, E., and Schaal, S., 2011, "Learning variable impedance control," International Journal of Robotics Research, 30(7), pp. 820-833.

[15] Abu-Dakka, F. J., Nemec, B., Jørgensen, J. A., Savarimuthu, T. R., Krüger, N., and Ude, A., 2015, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," Autonomous Robots, 39(2), pp. 199-217.

[16] Pastor, P., Kalakrishnan, M., Chitta, S., and Theodorou, E., "Skill learning and task outcome prediction for manipulation," Proceedings of the IEEE International Conference on Robotics and Automation, 2011, pp. 3828-3834.

[17] Kormushev, P., Calinon, S., and Caldwell, D. G., "Robot motor skill coordination with EM-based Reinforcement Learning," Proceedings of the Ieee/rsj International Conference on Intelligent Robots and Systems, 2010, pp. 3232-3237.

[18] Mnih, V. *et al.*, 2015, "Human-level control through deep reinforcement learning," Nature, 518(7540), p. 529.

[19] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M., "Deterministic policy gradient algorithms," Proceedings of the International Conference on International Conference on Machine Learning, 2014, pp. 387-395.

[20] Lillicrap, T. P. *et al.*, 2015, "Continuous control with deep reinforcement learning," Computer Science, 8(6), p. A187.

[21] Sharma, K., Shirwalkar, V., and Pal, P. K., "Intelligent and environment-independent Peg-In-Hole search strategies," Proceedings of the International Conference on Control, Automation, Robotics and Embedded Systems, 2014, pp. 1-6.

[22] Hogan, N., 1985, "Impedance Control : An Approach to Manipulation : Part-I theory," Asme Journal of Dynamic Systems Measurement & Control, 107(4), pp. 481-9.

[23] Albu-Schäffer, A. and Hirzinger, G., 2003, "Cartesian Compliant Control Strategies for Light-Weight, Flexible Joint Robots."

[24] Drake, S. H., 1977, "Using Compliance in Lieu of Sensory Feedback for Automatic Assembly," Massachusetts Institute of Technology.

[25] Peters, J., Vijayakumar, S., and Schaal, S., "Natural Actor-Critic," Proceedings of the European Conference on Machine Learning, 2005, pp. 280-291.

[26] Eberman, B. S. and Salisbury, K., 1989, "Whole-Arm Manipulation: Kinematics and Control," Massachusetts Institute of Technology.

[27] Ren, T., Dong, Y., Wu, D., Wang, G., and Chen, K., 2018, "Collision detection and identification for robot manipulators based on extended state observer," Control Engineering Practice, 79(pp. 144-153.

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

List of figure caption

List of table caption