International Journal of Intelligent Robotics and Applications (2020) 4:202–216
https://doi.org/10.1007/s41315-020-00138-z

REGULAR PAPER

Model accelerated reinforcement learning for high precision robotic assembly

Xin Zhao[1] · Huan Zhao[1] · Pengfei Chen[1] · Han Ding[1]

Received: 28 December 2019 / Accepted: 25 May 2020 / Published online: 2 June 2020

Abstract
Peg-in-hole assembly with narrow clearance is a typical robotic contact-rich task in industrial manufacturing. Robot learning allows robots to directly acquire the assembly skills for this task without modeling and recognizing the complex contact states. However, learning such skills is still challenging for robot because of the difficulties in collecting massive transitions data and transferring skills to new tasks, which inevitably leads to low training efficiency. This paper formulated the assembly task as a Markov decision process, and proposed a model accelerated reinforcement learning method to efficiently learn assembly policy. In this method, the assembly policy is learned with the maximum entropy reinforcement learning framework and executed with an impedance controller, which ensures exploration efficiency meanwhile allows transferring skills between tasks. To reduce sample complexity and improve training efficiency, the proposed method learns the environment dynamics with Gaussian Process while training policy, then, the learned dynamic model is utilized to improve target value estimation and generate virtual data to argument transition samples. This method can robustly learn assembly skills while minimizing real-world interaction requirements which makes it suitable for realistic assembly scenarios. To verify the proposed method, experiments on an industrial robot are conducted, and the results demonstrate that the proposed method improves the training efficiency by 31% compared with the method without model acceleration and the learned skill can be transferred to new tasks to accelerate the training for new policies.

Keywords Robotic assembly · Reinforcement learning · Peg-in-hole · Model acceleration

1 Introduction

Robots have been widely used in industrial tasks, such as assembly, grinding, and polishing. Classical programming methods are teaching robots to perform repeat motion by defining key frames using teach pendant or off-line programming software with CAD models. However, the position control-based method is not suitable for high precision assembly with narrow clearance, e.g. peg-in-hole insertion, due to low tracking accuracy and environmental uncertainties. Traditional methods are based on contact state models. These methods recognize the contact state according to the contact state model, and then use the corresponding control strategy to complete the assembly task. Commonly used contact state models include analytical model, such as quasi-static model (Whitney 1982; Kim et al. 1999), dynamic model (Xia et al. 2006), etc., and statistical model, such as Support Vector Machine (SVM) (Jakovljevic et al. 2012), fuzzy classifier (Jakovljevic et al. 2014), Gaussian Mixture Model (GMM) (Jasim et al. 2017), Hidden Markov Model (HMM) (Hannaford and Lee 1991) and so on. This kind of methods analyzes the assembly process with physical principles intrinsically and succinctly. However, these methods must identify and tune many parameters. This procedure is time-consuming or even impossible for some complex scenarios (Li et al. 2019).

In recent years, the learning-based method brought new opportunities to directly learn assembly policy without contact state models (Xu et al. 2019a). Because of its simple implementation and good versatility, this kind of method has been widely concerned in recent years. Among them, Learning from Demonstration (LfD) (Billard et al. 2008) and Reinforcement Learning (RL) (Sutton and Barto 2018) are two widely used methods. Considering that human beings

✉ Huan Zhao
huanzhao@hust.edu.cn

[1] State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, People's Republic of China

can easily complete various complex assembly tasks, LfD methods devote to imitate human assembly skills. Kronander (Kronander et al. 2014; Kronander 2015) modeled the joint probability distribution of the sensed wrench and angular velocity of the end effector with GMM, and then, Gaussian Mixture Regression (GMR) is used to generate the probability distribution of angular velocity conditioned on the sensed wrench. Based on this idea, Tang (Tang et al. 2016; Tang 2018) further proved the stability of the GMR controller and developed a handheld device to collect demonstrations. Kramberger et al. (2017) proposed a Dynamic Movement Primitive (DMP) based method to generalize orientational trajectories as well as the accompanying force-torque profiles for assembly tasks. Gao et al. (2019) encoded the motion and force information as a joint probability distribution model and then used a task planner and an adaptive control policy to accomplish assembly tasks. However, the performance of LfD methods completely depends on the quality of the demonstrations. To achieve good performance, the operator must demonstrate as many states as possible, which increases the operator's workload.

Different from the LfD method, the RL method learns skills through trial and error similar to the way humans learn. According to whether the task dynamic model is used in training, RL methods are divided into model-free and model-based methods. Model-free methods directly learn policy from data without considering the task dynamic model. Inoue et al. (2018) trained a recurrent neural network with Deep Q Network (DQN) (Silver et al. 2016) to learn a high-precision peg-in-hole assembly policy. Li et al. (2019) presented a deep RL method for a complex assembly process. In this method, the reward function uses a trained classification model to recognize whether the assembly is successful. However, actions in these methods are discrete, which inevitably leads to discontinuous movements. To solve this problem, Xu et al. (2019b) proposed a model-driven Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015) algorithm for multiple peg-in-hole assembly tasks. Ren et al. (2018) generated virtual trajectory and stiffness with a DDPG based variable compliance controller. However, Since DDPG is a deterministic policy method, to explore more states in training, noise must be added in the action or parameter spaces, which makes it difficult to stabilize and brittle to hyper-parameter settings. To solve this problem, Haarnoja et al. (2017) proposed a maximum entropy-based algorithm to learn stochastic policy. Later, Haarnoja et al. extended this algorithm and proposed soft actor-critic (SAC) (Haarnoja et al. 2018), a state-of-the-art off-policy actor-critic deep RL algorithm which provides sample-efficient learning while retaining the benefits of entropy maximization and stability. However, those model-free approaches need a significant amount of data samples, and collecting such excessive data is time-consuming and even may lead to physical damage in real robotic assembly tasks, which seriously limits the application of these methods in practice.

In contrast to model-free RL methods, model-based RL methods (Polydoros and Nalpantidis 2017) first learn the environment dynamics with deterministic models such as Locally Weighted Linear Regression (LWLR), Receptive Field Weighted Regression (RFWR), Decision Trees, etc. or stochastic models such as Gaussian Processes (GP), Locally Weighted Bayesian Regression (LWLR), etc., and then optimize trajectories with an optimal controller such as iterative Linear Quadratic Regulator (iLQR) or iterative Linear Quadratic Gaussian (iLQG). In these methods, the required number of samples can be reduced. Thomas et al. (2018) combined motion planning with Guided Policy Search (GPS) to learn assembly skills with prior knowledge from CAD models. Luo et al. (2018) proposed a Mirror Descent Guided Policy Search (MDGPS) based method for inserting a rigid peg into a deformable hole task. However, the performance of this kind of methods heavily depends on the accuracy of the dynamics model. Actually, learning such an accurate model in assembly tasks is difficult, which in turn affects the policy convergence. In addition, the learned skill is trained for the specific task and is difficult to be transferred to new scenarios. In a word, RL based methods can learn assembly skills by trial and error without contact state model and human demonstrations. However, it still challenging to robustly and efficiently learn transferable assembly skills.

In this paper, by integrating model-free and model-based learning techniques, a model accelerated RL method is proposed for high-precision assembly tasks, which robustly and efficiently learns assembly skills while minimizing real-world interaction requirements. In the training process, the assembly policy is learned with maximum entropy RL framework and executed by an impedance controller, which can guarantee exploration efficiency and safety and allow transferring the learned skill to new tasks. Meanwhile, the environment dynamics are learned with GP to improve target value estimation and generate virtual transitions to argument data samples, which in turn improves the training efficiency.

The rest of this paper is organized as follows: Sect. 2 introduces the peg-in-hole task and impedance control. Section 3 formulates the problem and introduces the learning algorithm. Section 4 introduces the model acceleration method and summarizes the proposed method in detail. Section 5 evaluates the proposed method with experiments on an industrial robot. Finally, Sect. 6 concludes the paper.

2 Robotic assembly task with impedance control

The peg-in-hole task can be divided into two stages, i.e. searching and insertion. In the searching stage, the robot moves to the position within the clearance region of the hole by means of vision system assistance or blind search method (S., R.C., M., S.B. 2001). After contact with the hole, it enters the more complicated insertion stage. In this stage, because the clearance is usually smaller than the motion accuracy of the robot, the insertion task cannot be fulfilled with only position control. For this reason, it is often necessary to install a force sensor on the robot end-effector and use the feedback force to realize the compliance control of the manipulator.

Impedance control is a widely used compliance control method, which keeps the robot end-effector and the environment maintaining a contact relationship like the spring damping system by controlling the position and velocity of the robot. In impedance control, the relationship between force and position can be described by impedance equation:

$$\mathbf{M}_d\ddot{\mathbf{e}} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e} = \mathbf{T}^T\left(\mathbf{x}_e\right)\left(\mathbf{f}_e - \mathbf{f}_d\right) , \tag{1}$$

where $\mathbf{M}_d$, $\mathbf{K}_D$ and $\mathbf{K}_P$ are the positive-definite mass, damping and stiffness matrix respectively, $\mathbf{e}$ is the difference between the actual end-effector pose $\mathbf{x}_e$ and the desired pose $\mathbf{x}_d$, the orientation of the end effector is represented by Euler angle $\boldsymbol{\varphi} = \left[\varphi_x, \varphi_y, \varphi_z\right]$ in form of XYZ, $\mathbf{f}_e$ and $\mathbf{f}_d$ are the actual and desired contact force respectively, $\mathbf{T}$ is the transformation matrix between analytic and geometric Jacobian defined as (Siciliano et al. 2010):

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \sin\varphi_y \\ 0 & 0 & 0 & 0 & \cos\varphi_x & -\sin\varphi_x\cos\varphi_y \\ 0 & 0 & 0 & 0 & \sin\varphi_x & \cos\varphi_x\cos\varphi_y \end{bmatrix} .$$

After the robot end-effector contacts with the environment, to control the desired contact force accurately under steady states, namely, $\mathbf{f}_e = \mathbf{f}_d$, the stiffness matrix is usually set $\mathbf{K}_P = \mathbf{0}$. Therefore, Eq. (1) becomes

$$\mathbf{M}_d\ddot{\mathbf{e}} + \mathbf{K}_D\dot{\mathbf{e}} = \mathbf{T}^T\left(\mathbf{x}_e\right)\left(\mathbf{f}_e - \mathbf{f}_d\right) . \tag{2}$$

In assembly tasks, it is generally considered that the environment is static and its position cannot be accurately obtained. Therefore, the desired velocity and acceleration are set $\ddot{\mathbf{x}}_d = \dot{\mathbf{x}}_d = 0$. For convenient applications in the control of industrial robots, the controller command should be converted to its discrete format as

$$\begin{aligned} \ddot{\mathbf{x}}_c(t) &= \mathbf{M}_d^{-1}\left[\mathbf{T}^T\left(\mathbf{x}_e\right)\left(\mathbf{f}_e - \mathbf{f}_d\right) - \mathbf{K}_D\dot{\mathbf{x}}_c(t-1)\right] , \\ \dot{\mathbf{x}}_c(t) &= \dot{\mathbf{x}}_c(t-1) + \ddot{\mathbf{x}}_c(t)T , \\ \mathbf{x}_c(t) &= \mathbf{x}_c(t-1) + \dot{\mathbf{x}}_c(t)T , \end{aligned} \tag{3}$$

where T is the system communication period between the impedance controller and the robot motion controller, $\mathbf{x}_c(t)$ is the pose command in Cartesian space. After $\mathbf{x}_c(t)$ is obtained, the joint position command can be calculated using the inverse kinematics, and the robot can be driven by sending it to the robot controller.

With the impedance controller, the peg can adjust in a small range to complete insertion, however, when the error is large, especially when the angular error is large, it will enter the jamming or wedging state. In this situation, the pose error cannot be eliminated with force controller. To solve this problem, a RL-based method is proposed in this paper. The framework of the proposed method is shown in Fig. 1. In this method, the desired contact force $\mathbf{f}_d$ is controlled with the policy learned from trial and error, and then the desired force $\mathbf{f}_d$ is used as the input of the impedance controller, finally, the robot position is adjusted to eliminate the pose error to complete the assembly task. In the following, the RL algorithm for the assembly policy is first introduced, and then, a model acceleration method is presented to improve the training efficiency.

3 Reinforcement learning for robotic assembly

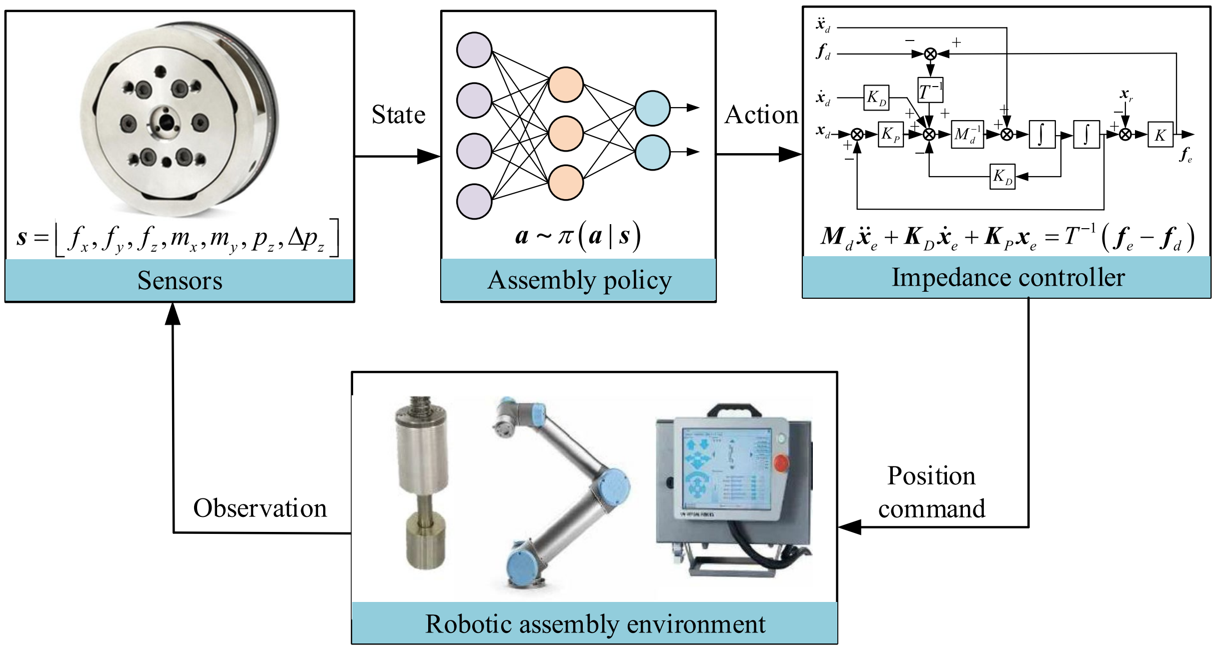
3.1 Problem formulation

The RL problem is usually formulated as a Markov decision process, which is defined as $\mathcal{M} = (S, A, P, R, \gamma)$, where S and A are the set of agent's states and actions respectively, $P\left(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t\right)$ is the probability of transition to state $\mathbf{s}_{t+1}$ from state $\mathbf{s}_t$ by executing action $\mathbf{a}_t$, also known as system dynamics. At each time step t, the robot chooses and executes an action $\mathbf{a}_t$ according to the assembly policy $\pi\left(\mathbf{a}_t|\mathbf{s}_t\right)$, and receives a reward $R\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\right)$, $\gamma \in (0, 1]$ is the discount factor. The goal of RL is to find a policy π to maximize the expectation of discounted returns from the initial state.

$$J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}\left[\sum_{i=0}^{\infty} \gamma^i R\left(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}_{i+1}\right)\right] . \tag{4}$$

To learn assembly policy with RL algorithms, the state and action space, and reward function must be defined first.

Because the contact force can characterize the contact state, therefore, it must be considered in the system state. The peg-in-hole assembly model is shown in Fig. 2. The contact force measured by the force sensor installed on the



State
Action
$s=[f_x,f_y,f_z,m_x,m_y,p_z,\Delta p_z]$
Sensors
$a\sim\pi(a|s)$
Assembly policy
$M_d\ddot{x}_e+K_D\dot{x}_e+K_P x_e=T^{-1}(f_e-f_d)$
Impedance controller
Observation
Position command
Robotic assembly environment

Fig. 1 Framework of the RL-based assembly method

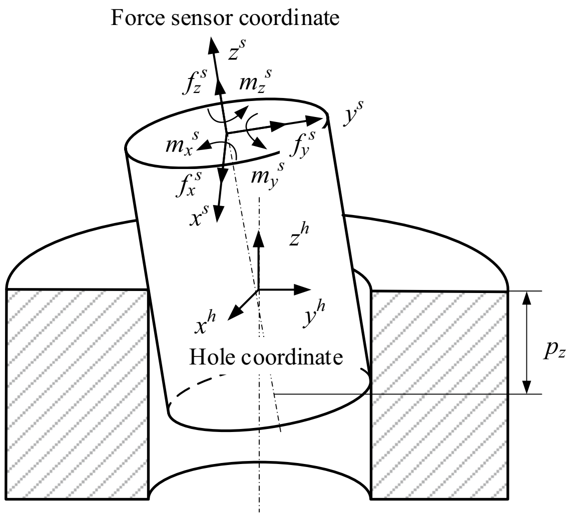


Fig. 2 State in insertion task

peg is $\mathbf{h}^s = \left[f_x^s, f_y^s, f_z^s, m_x^s, m_y^s, m_z^s\right]^T$. This force is represented in the force sensor coordinate $x^s - y^s - z^s$, however, in the peg-in-hole task, all the movements are represented in the hole coordinate i.e. $x^h - y^h - z^h$, therefore, the contact force should be transformed into the hole coordinate

$$\mathbf{h} = \begin{bmatrix} R_s & 0 \\ 0 & R_s \end{bmatrix} \mathbf{h}^s , \tag{5}$$

where R_s is the rotation of the force sensor relative to the hole coordinate. It is worth noting that only the direction of force is converted to the world coordinate system, the origin of the moment is still selected at the origin of the force sensor.

In addition, the insertion depth p_z is also needed to indicate the current task completion progress, and the axial displacement increment along the hole $\Delta p_z = p_z' - p_z$ can be used to determine whether the system is jammed. Therefore, the system status is expressed as:

$$\mathbf{s} = \left[f_x, f_y, f_z, m_x, m_y, p_z, \Delta p_z\right] . \tag{6}$$

For cylindrical peg-in-hole insertion task, the axial moment m_z is usually not considered. Because the exact positions of the hole or peg are not used in the state space, the system state will not be affected by the change of the hole's position.

There are many ways to define the action space, such as joint torque, velocity. However, these action space definitions are not only inefficient for training, but also dangerous for robot and parts before policy converges. Therefore, in this paper, the action is defined as the expected contact force $\mathbf{f}_d$ in the impedance control model. In the executing phase, the peg pose is controlled by an impedance controller according to the action $\mathbf{f}_d$. Compared with the ways of directly generating torque or velocity, using the desired contact force as the action and driving the robot through an impedance controller can avoid irrelevant or even dangerous actions. Therefore, it can improve the training efficiency and ensure the safety of the robot and parts before the policy converges. Therefore, the action is defined as:

$$\mathbf{a} = \left[f_x^d, f_y^d, f_z^d, m_x^d, m_y^d\right] . \tag{7}$$

The reward function is defined as:

$$r(\mathbf{s}, \mathbf{a}, \mathbf{s}') = -\frac{1}{N_{max}} + \frac{\Delta p_z}{H} + r_{end} , \tag{8}$$

where N_{max} is the maximum number of steps allowed for one episode, if the number of steps is exceeded, this episode is considered a failure and ends immediately; H is the insertion goal depth, and $\Delta p_z = p_z' - p_z$ is the change of the insertion depth after action $\boldsymbol{a}$ is executed. r_{end} is the reward in the last step of an episode, defined as

$$r_{end} = \begin{cases} 1 & p_z > H \\ -1 & \mathbf{f}_{min} > \mathbf{f}_e \ \ or \ \ \mathbf{f}_{max} < \mathbf{f}_e \\ 0 & others \end{cases} , \tag{9}$$

where $\mathbf{f}_{min}$ and $\mathbf{f}_{max}$ are the minimum and maximum allowed contact force. The reward in Eq. (8) is designed to stay within the range of $\left[-1/N_{max} - 2, -1/N_{max} + 2\right]$. The first two items represent the immediate rewards in each step which encourages the actions that accomplish the task in the minimum number of steps and achieve as much insert progress as possible. These items can avoid the problem of low learning efficiency caused by sparse reward. Besides, the last item encourages the actions that successfully perform the task and punishes the actions that cause large contact force exceeding the allowed range.

3.2 Learning algorithm

The proposed method is based on SAC (Haarnoja et al. 2018), a model-free actor-critic algorithm in maximum entropy RL framework that can efficiently learn policies in high-dimensional continuous action space.

Traditional RL algorithms directly maximize the objective function in Eq. (4). However, in the maximum entropy RL framework, in addition to maximizing the cumulative return, it also needs to ensure the exploration ability of the policy. Therefore, the objective function is defined as:

$$J = \sum_{t=0}^{T} \mathbb{E}_{r_t, \mathbf{s}_t \sim E, \mathbf{a}_t \sim \pi}\left[r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t))\right] , \tag{10}$$

where $\mathcal{H}(\pi(\cdot|\mathbf{s}_i))$ is the entropy of policy π, α is the temperature parameter that determines the relative importance of the entropy term against the reward. Adding entropy term to the objective function can maximize the accumulative reward while maintaining the exploration ability of the policy at the same time.

Soft state action value function and soft state value function are defined as:

$$Q(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + \mathbb{E}_{(\mathbf{s}_{t+1}, \ldots) \sim \rho_\pi} \times \left[\sum_{l=1}^{\infty} \gamma^l \left(r(\mathbf{s}_{t+l}, \mathbf{a}_{t+l}, \mathbf{s}_{t+l+1}) + \alpha \mathcal{H}(\pi^*_{\text{MaxEnt}}(\cdot|\mathbf{s}_{t+l}))\right)\right] , \tag{11}$$

$$V(\mathbf{s}_t) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q(\mathbf{s}_t, \mathbf{a}')\right) \mathrm{d}\mathbf{a}' . \tag{12}$$

The soft Bellman equation is represented as:

$$Q(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V(\mathbf{s}_{t+1})\right] , \tag{13}$$

where $(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots)$ is the trajectory obtained by executing policy π. According to the value function, the optimal policy is

$$\pi^*_{\text{MaxEnt}}(\mathbf{a}_t|\mathbf{s}_t) = \exp\left(\frac{1}{\alpha}\left(Q(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t)\right)\right) . \tag{14}$$

In SAC, the actor and critic are approximated by corresponding networks $\pi_\theta(\mathbf{a}|\mathbf{s})$ and $Q_w(\mathbf{s}, \mathbf{a})$, parameterized by θ and w respectively. To stabilize the learning procedure, the target value is calculated using a slowly updated target Q-network, denoted by $Q_{w'}(\mathbf{s}, \mathbf{a})$ parameterized by w'. To improve data efficiency, experience replay buffer $\mathcal{D}$ is built to store the experience transition data $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$. In each training step, a mini-batch is randomly sampled from $\mathcal{D}$, and the critic network is updated by minimizing the following loss function:

$$J_Q(w) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_w(\mathbf{s}_t, \mathbf{a}_t) - y_t\right)^2\right] , \tag{15}$$

where y_t is target value and more details will be introduced in Sect. 4.

In traditional actor-critical algorithms, the policy network is updated with the gradient of $Q_w(\mathbf{s}, \mathbf{a})$. However, since the policy defined by Eq. (14) is intractable, it is necessary to re-parameterize the policy. One option is the Gaussian policy. To limit the action to a specific interval $\mathbf{a}_t \in [\mathbf{a}_{min}, \ \mathbf{a}_{max}]$, the action is expressed as:

$$\begin{aligned} \mathbf{u}_t &= \boldsymbol{\mu}_\theta(\mathbf{s}_t) + \boldsymbol{\varepsilon}_t \circ \boldsymbol{\sigma}_\theta(\mathbf{s}_t) , \\ \mathbf{a}_t &= \mathbf{a}_s \circ \tanh(\mathbf{u}_t) + \mathbf{a}_{mean} , \end{aligned} \tag{16}$$

where $\boldsymbol{\mu}_\theta(\mathbf{s}_t)$, $\boldsymbol{\sigma}_\theta(\mathbf{s}_t)$ are the two output heads of the policy network, represent mean and variance, respectively; $\boldsymbol{\varepsilon}_t$ is the noise vector, sampled from the Normal distribution, $\circ$ is the Hadamard product, $\mathbf{a}_s$ and $\mathbf{a}_{mean}$ are the scale and mean of action:

$$\mathbf{a}_s = \frac{\mathbf{a}_{max} - \mathbf{a}_{min}}{2}, \quad \mathbf{a}_{mean} = \frac{\mathbf{a}_{max} + \mathbf{a}_{min}}{2}, \tag{17}$$

where $\mathbf{a}_{max}$ and $\mathbf{a}_{min}$ are the upper and lower limits of actions. In this parameterization, the log-likelihood of action $\mathbf{a}_t$ is:

$$\log \pi(\mathbf{a}_t|\mathbf{s}_t) = \log \mathcal{N}(\mathbf{a}_t|\boldsymbol{\mu}_\theta(\mathbf{s}_t), \boldsymbol{\sigma}_\theta(\mathbf{s}_t)) - \sum_{i=1}^{D} \log\left(a_{s,i}\left(1 - \tanh^2\left(a_{t,i}\right)\right)\right), \tag{18}$$

where $a_{s,i}$ and $a_{t,i}$ are the i-th elements of $\mathbf{a}_s$ and $\mathbf{a}_t$ respectively.

In this way, random action $\mathbf{a}_t$ can enhance the exploration ability, while ensuring security. In the training procedure, the policy network parameters are updated to minimize the Kullback–Leibler (KL) divergence:

$$J_\pi(\theta) = D_{KL}\left(\pi(\cdot|\mathbf{s}_t) \middle\| \exp\left(\frac{1}{\alpha}\left(Q(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t)\right)\right)\right). \tag{19}$$

With Eq. (12) and (14), the objective can be re-written as:

$$J_\pi(\theta) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \mathbf{a}_t \sim \pi_\theta}\left[\alpha \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) - Q_w(\mathbf{s}_t, \mathbf{a}_t)\right]. \tag{20}$$

The parameter w' of the target network $Q_{w'}(\mathbf{s}, \mathbf{a})$ is softly updated by:

$$w' = \tau w + (1 - \tau) w', \tag{21}$$

where $0 < \tau \ll 1$ is the update rate.

In addition, the temperature coefficient α determines the weight of entropy in the objective (10), therefore, it affects the learned assembly policy. During the training process, it is updated with the following objective

$$J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi}\left[-\alpha \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) - \alpha \tilde{\mathcal{H}}\right], \tag{22}$$

where $\tilde{\mathcal{H}}$ is the entropy target.

4 Model acceleration

In conventional methods, the target value in Eq. (15) is calculated with TD(0), which is based on the one-step reward and bootstrapping from the estimated value of the next state with the target network $Q_{w'}(\mathbf{s}, \mathbf{a})$. However, the inaccuracy of the target network leads to an inaccurate target value which in turn reduces training efficiency. Besides, SAC is a model-free algorithm, the parameters are updated only with real transition data $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}_{i+1}, r_i)$ without considering system dynamics $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, which leads to significant large demand for transition data. However, in robotic applications, collect such amount of data is time-consuming and tedious even may lead to physical damage. Therefore, improve data efficiency is of great significance. To solve these problems, this paper learns the environment dynamics model and then use it to improve the estimation of future reward and augment the transition data.

4.1 Gaussian process for dynamics modeling

Existing researches have proposed a variety of model classes for dynamics modeling. Among them, the GP is the state-of-the-art approach, because the GP is a distribution over functions with a continuous domain and there is not any assumption about the function that maps current states and actions to future states (Polydoros and Nalpantidis 2017). This fact makes GP a powerful method for dynamics modeling.

A GP is specified by a mean function $m(\cdot)$ and a covariance/kernel function $k(\cdot)$:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right). \tag{23}$$

To predict the state $\mathbf{s}_{t+1}$ after action $\mathbf{a}_t$ is performed in state $\mathbf{s}_t$, $\tilde{\mathbf{x}} = (\mathbf{s}_t, \mathbf{a}_t)$ is used as training input, and the difference $\boldsymbol{\Delta}_t = \mathbf{s}_{t+1} - \mathbf{s}_t$ is taken as training target. In this way, the prior mean function is set to $m(\cdot) \equiv 0$, which simplifies the calculation. The kernel function is defined as:

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}')^T \boldsymbol{\Lambda}^{-1} (\tilde{\mathbf{x}} - \tilde{\mathbf{x}}')\right), \tag{24}$$

where $\boldsymbol{\Lambda} = diag\left(\left[l_1^2, \ldots, l_D^2\right]\right)$, l_i is the characteristic length-scales, σ_f is the signal variance. Given a set of training input data $\tilde{\mathbf{X}} = \left[\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_n\right]$ and the corresponding target value $\mathbf{y} = \left[\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2, \ldots, \boldsymbol{\Delta}_n\right]$, the hyper-parameters can be learned by maximizing the marginal likelihood (Williams and Rasmussen 2006).

The environment dynamic model can be used to predict the results of actions $\mathbf{a}_t$ performed in state $\mathbf{s}_t$. In the GP model, the posterior distribution of successor states is Gaussian distribution:

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = \mathcal{N}(\mathbf{s}_{t+1}|\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}), \tag{25}$$

where

$$\boldsymbol{\mu}_{t+1} = \mathbf{s}_t + \mathbb{E}_f\left[\boldsymbol{\Delta}_t\right], \quad \boldsymbol{\Sigma}_{t+1} = \mathrm{var}_f\left[\boldsymbol{\Delta}_t\right]. \tag{26}$$

The mean and variance of the GP predictor are

$$\begin{aligned}\mathbb{E}_f\left[\boldsymbol{\Delta}_t\right] &= \mathbf{k}_*^T\left(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I}\right)^{-1}\mathbf{y}\ ,\\ \mathrm{var}_f\left[\boldsymbol{\Delta}_t\right] &= \mathbf{k}_{**}-\mathbf{k}_*^T\left(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I}\right)^{-1}\mathbf{k}_*\ ,\end{aligned} \tag{27}$$

where $\mathbf{k}_* = k(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_t)$, $\mathbf{k}_{**} = k(\tilde{\mathbf{x}}_t, \tilde{\mathbf{x}}_t)$ and $\mathbf{K}$ is the kernel matrix with elements $K_{ij} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$, σ_ε is the noise variance.

4.2 Model acceleration with λ-return

To improve target value estimation, the short-term value return is estimated by unrolling the learned model dynamics, and the long-term value is estimated with the learned target critic network. According to Eq. (11), the n-step TD target of state $\mathbf{s}_t$ under policy π_θ is

$$\begin{aligned} y_t^n = &\sum_{i=0}^{n}\gamma^i\left[r(\mathbf{s}_{t+i},\mathbf{a}_{t+i},\mathbf{s}_{t+i+1}) - \alpha\gamma\log\pi(\mathbf{a}_{t+i+1}|\mathbf{s}_{t+i+1})\right]\\ &+\gamma^{n+1}Q_{w'}(\mathbf{s}_{t+n+1},\mathbf{a}_{t+n+1})\ ,\end{aligned} \tag{28}$$

where $P(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t) \sim \mathcal{GP}$. Since the action $\mathbf{a}_t$ is sampled from the current policy $\pi_\theta(\mathbf{s}_t)$, it is an on-policy procedure, importance weights are not needed.

Since the rollout is simulated by the dynamic model rather than interacted with the environment, the distribution mismatch of the dynamic model will directly affect the value estimation accuracy, and the estimation error will accumulate with the simulation step increase. To solve this problem, the λ-return is used to average the n-steps target along with the rollout:

$$y_t^\lambda = (1-\lambda)\sum_{i=1}^{n}\lambda^{i-1}y_t^{i-1} + \lambda^n y_t^n\ , \tag{29}$$

In this way, the estimated target value can be improved, which in turn improves the training efficiency.

4.3 Model acceleration with imagination

Another obstacle for applying deep RL methods in real assembly tasks is the large demand for transition data. However, after the dynamic model is learned, virtual transition data can be generated by simulating the model to improve data efficiency.

In conventional methods, the transition $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$ sampled from the replay buffer $\mathcal{D}$ only considers the action $\mathbf{a}_t$ selected and executed in the past. To explore more states, the proposed method selects new action $\mathbf{a}'_t$ with the current policy and predicts the successor state and reward with the learned dynamic model and reward function, and then, augments transition data with these virtual transitions.

Algorithm 1 Network training with model accelerated SAC

```
1: Initialize actor and critic network with random weights
2: Initialize target value network with the same weight with critic
3: Initialize replay buffer D
4: for episode = 0 to $M$ do
5:     Apply action $a_t$ selected with $\pi_\theta$ until the episode ends, collect transitions
       $(s_t, a_t, s_{t+1}, r_t)$ and store in D
6:     Fit dynamic model GP
7:     for $t = 0$ to $T$ do
8:         Sample mini batch $B$ from D
9:         Estimate target value $y_{t,n}^{\lambda}$ with $\pi_\theta$, GP according to Eq. (29)
10:        Update $w$, $\theta$, $w'$ and $\alpha$ according to (15), (20), (21) and (22)
11:        for $i = 0$ to $N$ do
12:            Select action $a'_t$ with $\pi_\theta$ in $s_t$ of $B$
13:            Predict successor state $s'_{t+1}$ and reward $r'_{t+1}$ with GP
14:            Estimate target value $y'^{\lambda}_{t,n}$ with $\pi_\theta$, GP according to Eq. (29)
15:            Update $w$, $\theta$, $w'$ and $\alpha$ according to (15), (20), (21) and (22)
16:        end for
17:    end for
18: end for
```

Moreover, because these virtual data are generated by the current policy, the training is on-policy. In addition, all data are generated online, there is no need for extra replay buffer to save virtual data. The model acceleration method is shown in Fig. 3.

4.4 Algorithm summary

The architecture of the proposed model accelerated RL algorithm for robotic assembly task is shown in Fig. 4. To further reduce the critic estimation error, similar to TD3 (Fujimoto et al. 2018), two critic networks are used in SAC. For each time step, the critics and actor are updated with the minimum target and critic value of actions selected with the policy. In the training process, the dynamic model is updated after an assembly trial is completed, and the network parameters are trained T times with real data sampled from replay buffer and $T \times N$ times with virtual data generated by the dynamic model. The exact procedure is listed in Algorithm 1.

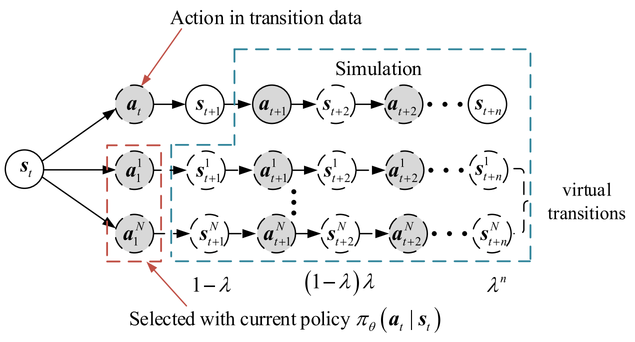


Fig. 3 λ-return and virtual transition with the dynamic model

5 Experiments

5.1 Experimental setup

To validate the feasibility of the proposed learning-based assembly method, experiments are conducted on an industrial robot. The layout of the system architecture for experiments is illustrated in Fig. 5. The robot is UR5 from Universal robotics, and a force sensor is mounted on the end effector to measure the contact force between the peg and hole. The assembly controller and impedance controller are implemented on a PC with Intel Core i7- 5500U CPU @ 2.40 GHz CPU, 16 GB RAM, and Ubuntu 14.04 operating system. The PC collects force/torque from a force sensor with MCC 1608-FS-Plus data acquisition device and communicates with the robot controller with TCP/IP protocol at

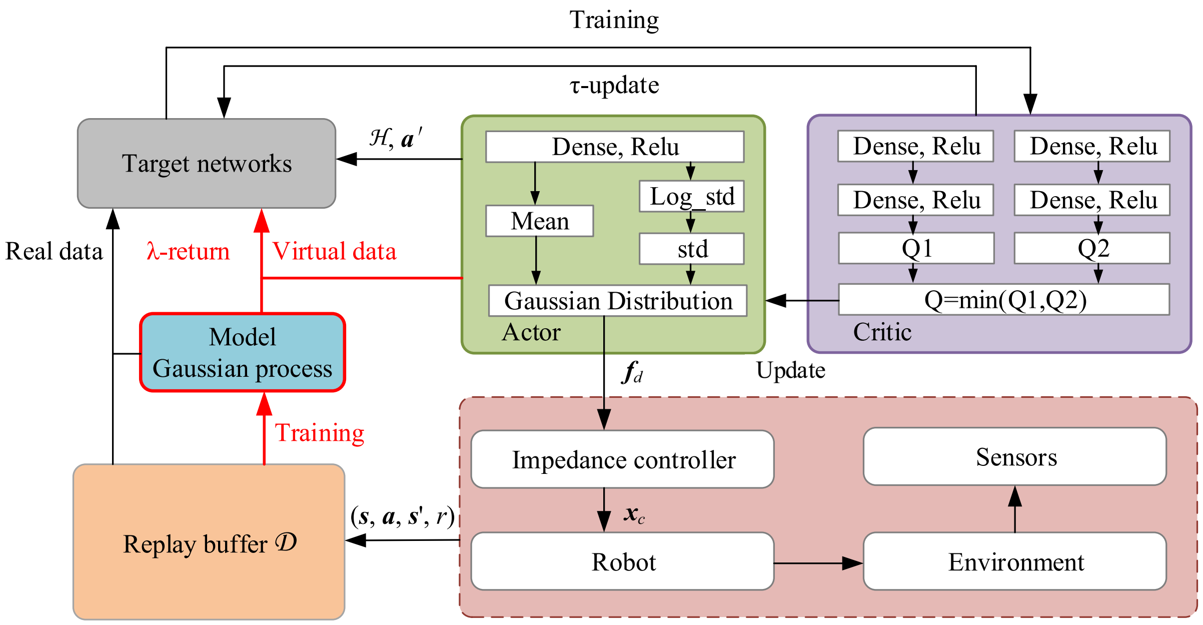


Fig. 4 The architecture of the proposed method

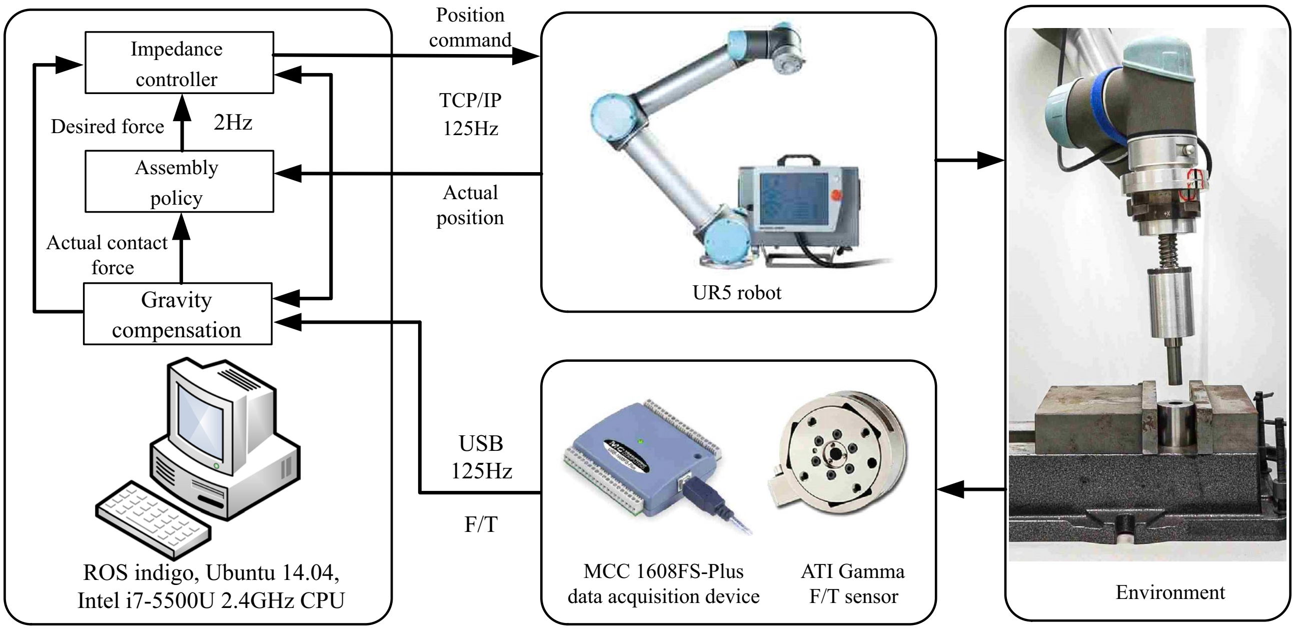


Fig. 5 Experimental layout

Table 1 Hyper-parameters for training

Parameters	Value
Batch size	64
Learning rate	0.0003
Target critic update rate τ	0.005
Replay buffer size	3000
Hidden layer size	32
Discount factor γ	0.99
Entropy target $\tilde{\mathcal{H}}$	-7
Training times per episode T	50
Expansion steps n	4
Imagination times N	3
Trace decay factor λ	0.8

the rate of 125 Hz. After acquiring the F/T measurement and robot position, the actual contact force is calculated by the real-time gravity compensation module. All the programming is implemented with the Robot Operating System (ROS) framework, and the assembly controller and impedance controller are in different nodes, the impedance controller runs at 125 Hz, and the assembly controller runs at 2 Hz. The peg and hole used in the experiment are made of steel, and their diameters are 15.00 mm and 15.02 mm, respectively, the depth of the hole is 35 mm. Different from the two-dimensional assembly experiment in Ref. (Ren et al. 2018), the more complex three-dimensional assembly task is adopted in this experiment.

During training, the robot starts from a random pose within ± 0.1 mm lateral error and $\pm 4°$ angular error. The mass matrix $\mathbf{M}_d$ and damping matrix $\mathbf{K}_D$ in the impedance controller are set to diag([0.67, 0.67, 0.67, 0.017, 0.017, 0.017]) and diag([266.7, 266.7, 533.3, 3.3, 3.3, 3.3]) respectively. The lower and upper limits of action in Eq. (17) are set to ([− 5 N, − 5 N, 20 N, − 0.5 Nm, − 0.5 Nm]) and ([5 N, 5 N, 30 N, 0.5 Nm, 0.5 Nm]) respectively. The maximum number of steps N_{max} is 60. For the safety of the robot and force sensor, the maximum allowed force and moment are 70 N and 4 Nm respectively. In one episode, the robot executes

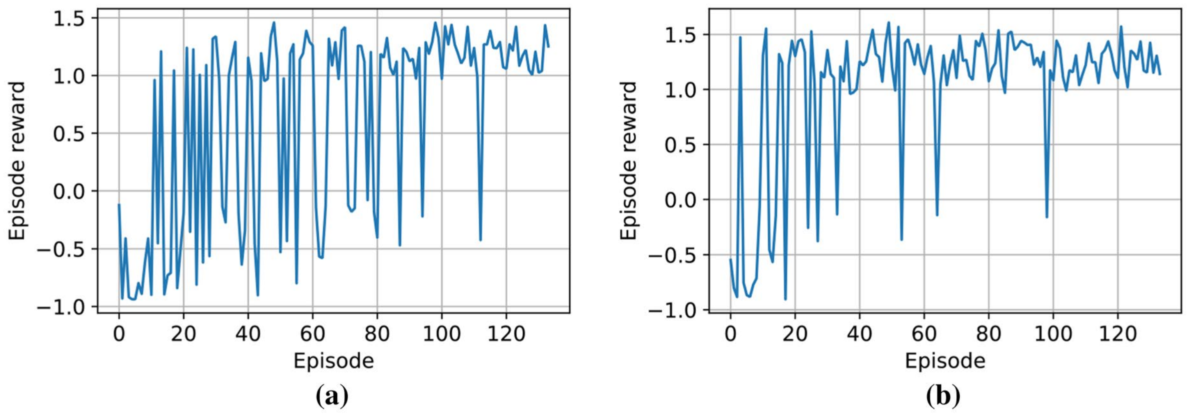


Fig. 6 Training reward curve. **a** Episode reward without model acceleration; **b** Episode reward with model acceleration

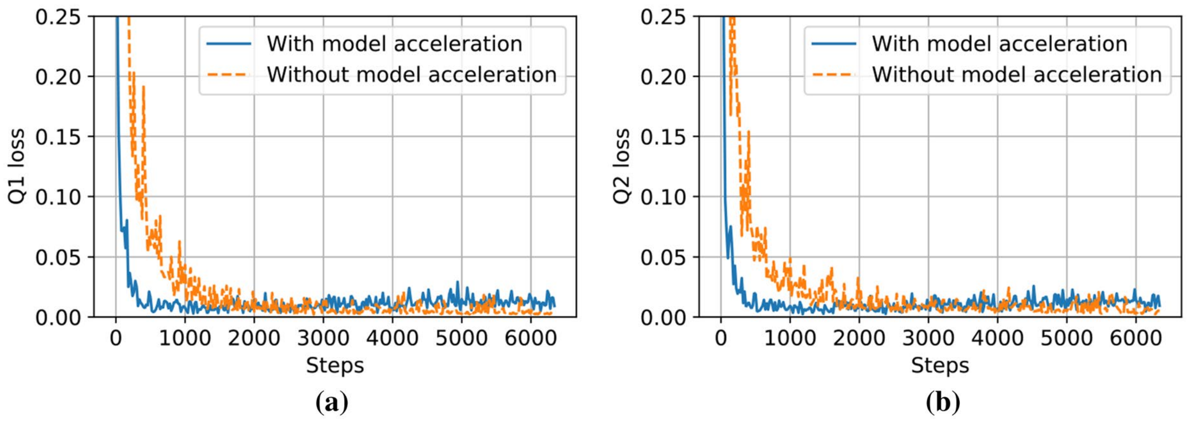


Fig. 7 Critic losses in training. **a** Q1; **b** Q2

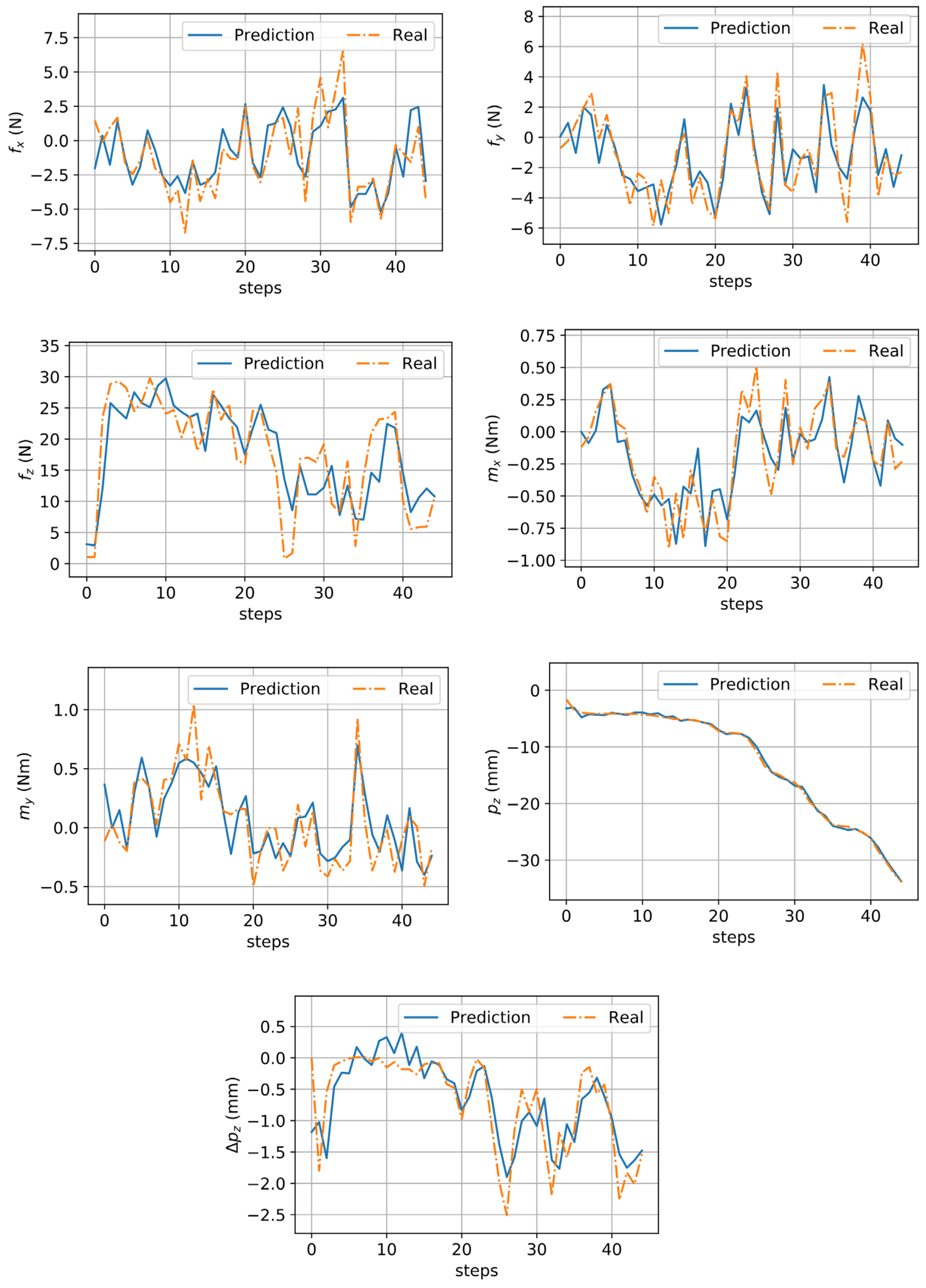


Fig. 8 Prediction result with the dynamic model for one episode

the assembly actions step by step until the task succeeds by inserting the peg into goal depth or fail by whether exceeding the maximum allowed contact forces/torque boundary or the maximum number of steps. The dynamic model is fitted with 400 real transition samples. The other hyper-parameters are listed in Table 1.

5.2 Results

To evaluate the training performance of the proposed method, the insert task is trained for 135 episodes in two methods, i.e. without model acceleration and with model acceleration, the training reward is shown in Fig. 6. It can be fund that, the performances of the two methods are improved with the increase of episodes. After the policy converges, the reward and success rate of the two methods are basically the same, which means the two learned policies have the same performances. In Fig. 6a the policy achieves a high success rate after 95 episodes. However, as shown in Fig. 6b, with the benefits of the dynamic model, the proposed method only needs 65 episodes, which demonstrates the training efficiency is improved by 31%. The average Q value losses are shown in Fig. 7, with model acceleration, the loss values are reduced to 0.01 with 600 training steps, however, without model acceleration, it will take 2000 training steps. The reason for this is, on the one hand, the target value is improved with λ-return using the dynamic model, on the other hand, the current policy is used to select actions and simulated with the dynamic model to explore more states and actions. After 3000 training steps, due to the error of the learned dynamic model, the loss values for the proposed method are a little larger. However, since the loss value is very small, it has little influence on the policy. It is worth noting that although the model acceleration will increase the calculation, however, the learning process is in the initialization of the robot between two assembly executions, therefore, it does not increase the total time of the training process.

In the training process, a GP is used to learn the dynamics, and Fig. 8 shows the prediction result with the learned model for one episode. Because the actual contact point in the insertion will change with the adjustment of peg's pose, and the desired contact force may not be achieved in one control cycle of the assembly controller, the specific value of the next state cannot be accurately predicted, but the results reflect the trend of the states. For this reason, the λ-return is used to reduce the influence of this distribution mismatch in the proposed method.

After the training process, the learned assembly policy is evaluated. An insert process with 5° angular error is shown in Fig. 9, and the insert trajectory is shown in Fig. 10. The robot starts from a random pose as shown in Fig. 9a and it enters the insertion stage when it reaches the top of the hole. During the first 4 steps, the peg inserts down quickly until in the 5-th step contacts with the hole, as shown in Fig. 9b. Due to the large angular error, the peg gets stuck and falls into jamming state. In step 5–45, the peg adjusts pose by executing the actions selected according to the assembly policy with the impedance controller. Finally, the angular error is eliminated in step 45, as shown in Fig. 9c. In the last stage, the peg dashes down to the goal depth and finish the insert task.

The results of the first experiment show that the proposed method and SAC have the same performances in success rate and episode reward after the policies converge. To further demonstrate the effectiveness of the proposed method, the success rate comparison between the proposed method and the pure impedance control method is

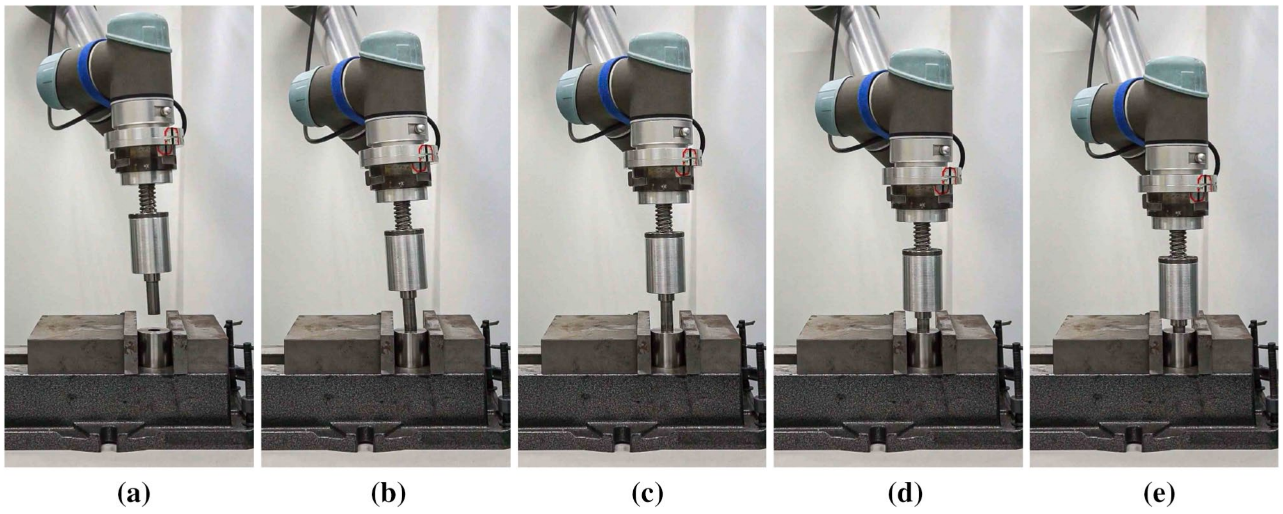


Fig. 9 Insert process with 5° angular error. **a** Starting from a random pose; **b**, **c** align the peg with the hole; **d**, **e** dashing

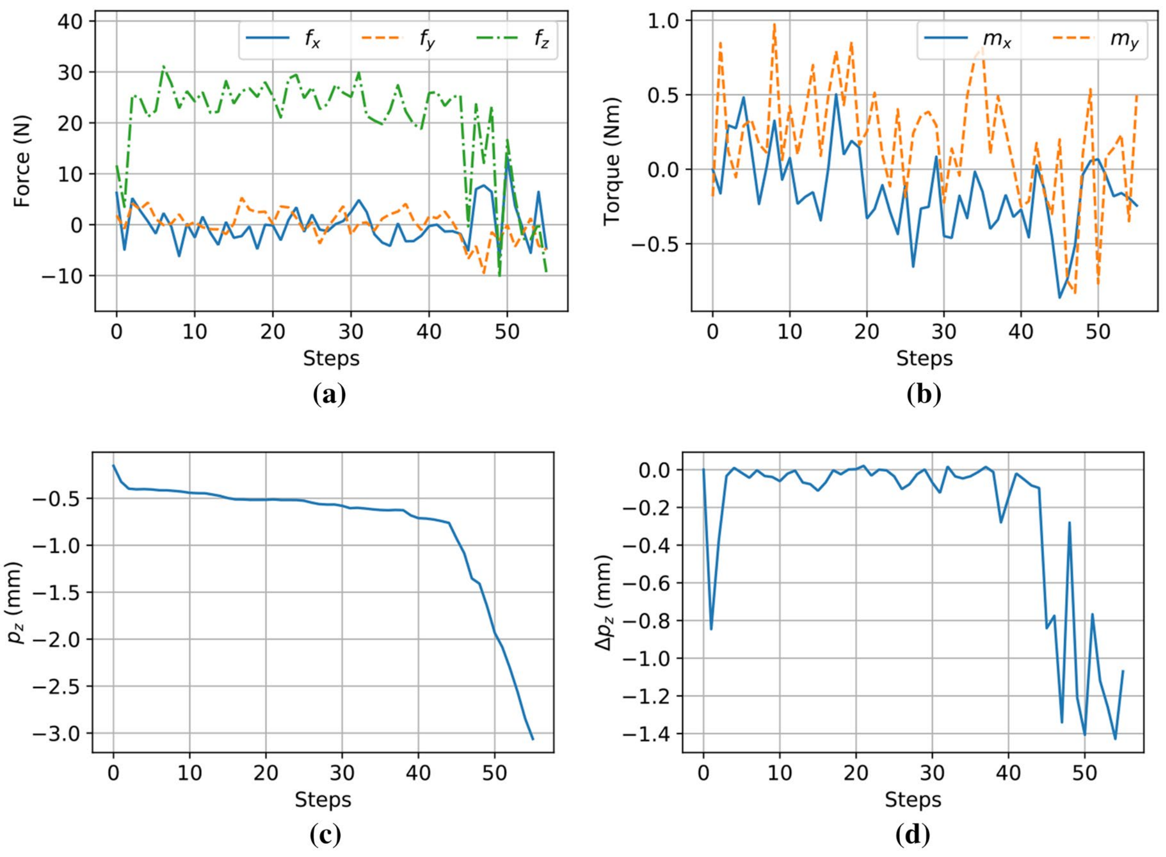


Fig. 10 Trajectory by the learned policy. **a** Force trajectory; **b** Moment trajectory; **c** z-axis position; **d** z-axis position increment

Table 2 Success rate with different angular errors

Angular error (°)	1	2	3	4	5	6	7	8	9
Proposed method	1.0	1.0	1.0	1.0	1.0	0.5	0.3	0.1	0.0
Impedance control	1.0	1.0	0.7	0.4	0.0	0.0	0.0	0.0	0.0

conducted with different angular errors. In the pure impedance control method, the assembly process is only controlled with the impedance controller described in Eq. (3). For each angular error, ten inserts are conducted, and the result is listed in Table 2. It can be found that the pure impedance control method can only deal with the maximum angular error not greater than 4°. This is because jamming happens when the angular error is large. At this time, the error cannot be eliminated by using only impedance control. However, the proposed method can ensure success up to 5° and it is still possible to succeed when the angular error is 8°. Meanwhile, because of the continuous action and stochastic policy adopted in the proposed method, the performance of the proposed method is better than that of the DQN based method proposed in Ref. (Inoue et al. 2018), where the maximum allowable initial angular error is 1.6°.

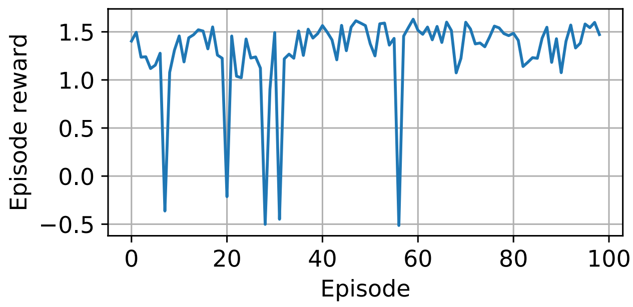


Fig. 11 Training reward curve of transferring the learned policy to a new task

Although the proposed method can speed up the training process, learning such skills from scratch still needs a lot of trials. A standard way to accelerate training is initializing

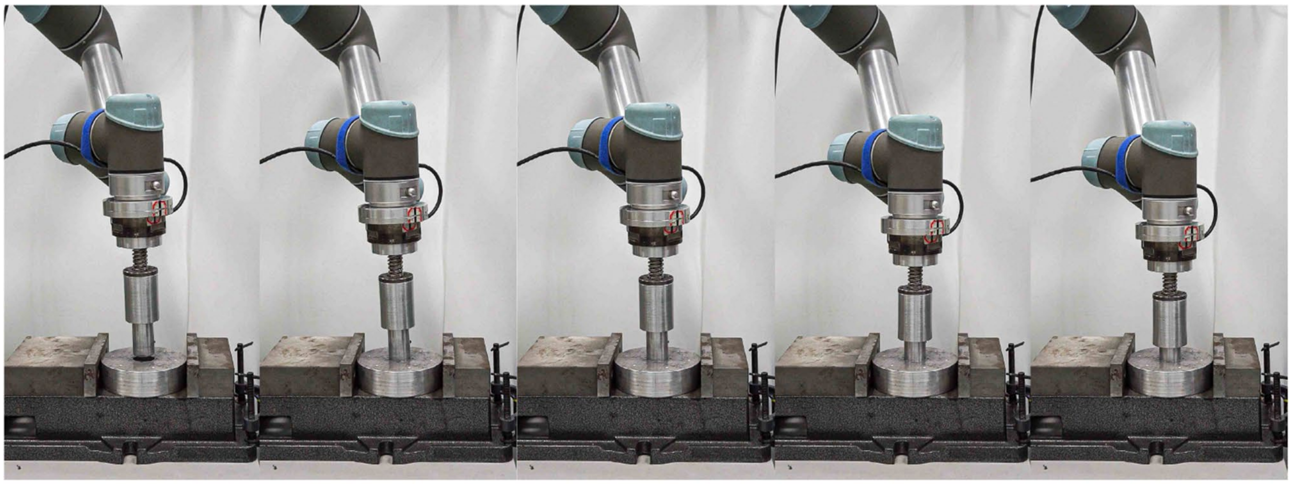


Fig. 12 Insert process for the new parts. **a** Starting from a random pose; **b**, **c** align the peg with the hole; **d**, **e** dashing

the networks with the one trained for another task. Next, we will validate the transferability of the learned skill for accelerating the learning process. In this experiment, a new insertion task is adopted, where the diameters for peg and hole are 29.98 mm and 30.01 mm respectively, and they are all made of aluminum alloy. And the networks' weights are initialized with the learned ones in the last experiment. The initial lateral and angular errors are ± 0.1 mm and $\pm$ 4° respectively, the training reward is shown in Fig. 11 and an insert process is shown in Fig. 12. In the first 30 episodes, the learned policy can still complete the new task with 86.7% success rate, and the performance is gradually improved with the increase of learning steps. Compared with the zero-start situation in Fig. 6, the training efficiency is significantly improved. This is because, on the one hand, the Gaussian policy used in this paper has a strong exploration ability; on the other hand, an impedance controller is used in the low-level control, therefore, the assembly policy is less affected by the size and materials of parts and the learned policy can serve as a good initialization for a new task.

6 Conclusion

In this paper, a RL method is proposed to learn high precision robotic assembly skills, which integrates model-free and model-based learning techniques. Compared with existing methods, it has the following advantages: (1) the assembly policy is learned with the maximum entropy RL framework, the training process is efficient and robust, meanwhile, the learned skill is allowed be transferred to new tasks; (2) the robot is driven by an impedance controller to ensure the exploration efficiency and safety in training; (3) to improve training efficiency and reduce sample complexity, the environment dynamics are learned with Gaussian Process to improve the target value estimation and argument transition samples. Experiments are conducted on an industrial robot. The results demonstrate that the proposed method can reduce sample complexity and improve the training efficiency compared with the model-free method, and the learned policy can serve as a good initialization for a new task.

Considering that in the process of learning skills, people usually do not directly start from scratch, but first acquire the initial skills through imitation, and then refine the skills through trial and error. Therefore, one of the directions of our future work is to combine imitation learning and reinforcement learning to improve learning efficiency. In addition, this paper only adopted the classical cylindrical peg-in-hole to validate the proposed method, the future work will apply the proposed method to irregular shape or deformable parts assembly tasks.

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grant Nos. 91748114, 51535004 and the Innovation Group Program of Hubei Province, China under Grant No. 2017CFA003.

References

Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot programming by demonstration. Springer handbook of robotics, 1371–1394 (2008)

Chhatpar, S.R., Branicky, M.S.: Search strategies for peg-in-hole assemblies with position uncertainty. Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems.

Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180), 20011465–1470. (2001)

Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. Paper presented at the Proceedings of the 35th International Conference on Machine Learning (2018)

Gao, X., Ling, J., Xiao, X., Li, M.: Learning force-relevant skills from human demonstration. Complexity **2019**, 1–11 (2019). https://doi.org/10.1155/2019/5262859

Haarnoja, T., Tang, H., Abbeel, P., Levine, S.: Reinforcement learning with deep energy-based policiesProceedings of the 34th International Conference on Machine Learning Volume 70, Sydney, NSW, Australia, 2017. JMLR.org, p 1352–1361. (2017)

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research, 2018. PMLR, p 1861–1870. (2018)

Hannaford, B., Lee, P.: Hidden markov model analysis of force/torque information in telemanipulation. Int. J. Robot. Res. **10**(5), 528–539 (1991). https://doi.org/10.1177/027836499101000508

Inoue, T., De Magistris, G., Munawar, A., Yokoya, T., Tachibana, R.: Deep reinforcement learning for high precision assembly tasks. Paper presented at the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018)

Jakovljevic, Z., Petrovic, P.B., Hodolic, J.: Contact states recognition in robotic part mating based on support vector machines. Int. J. Adv. Manuf. Technol. **59**(1), 377–395 (2012). https://doi.org/10.1007/s00170-011-3501-5

Jakovljevic, Z., Petrovic, P.B., Mikovic, V.D., Pajic, M.: Fuzzy inference mechanism for recognition of contact states in intelligent robotic assembly. J Intell Manuf **25**(3), 571–587 (2014). https://doi.org/10.1007/s10845-012-0706-x

Jasim, I.F., Plapper, P.W., Voos, H.: Contact-state modelling in force-controlled robotic peg-in-hole assembly processes of flexible objects using optimised Gaussian mixtures. Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. **231**(8), 1448–1463 (2017). https://doi.org/10.1177/0954405415598945

Kim, I., Lim, D., Kim, K.: Active peg-in-hole of chamferless parts using force/moment sensor. Paper presented at the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems., 1999

Kramberger, A., Gams, A., Nemec, B., Chrysostomou, D., Madsen, O., Ude, A.: Generalization of orientation trajectories and force-torque profiles for robotic assembly. Robot Auton Syst **98**, 333–346 (2017). https://doi.org/10.1016/j.robot.2017.09.019

Kronander, K.J.A.: Control and learning of compliant manipulation skills. doctoral, EPFL (2015)

Kronander, K., Burdet, E., Billard, A.: Task transfer via collaborative manipulation for insertion assembly. Paper presented at the 2014 Workshop on human-robot interaction for industrial manufacturing, robotics, science and systems (2014)

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)

Li, F., Jiang, Q., Zhang, S., Wei, M., Song, R.: Robot skill acquisition in assembly process using deep reinforcement learning. Neurocomputing **345**, 92–102 (2019). https://doi.org/10.1016/j.neucom.2019.01.087

Luo, J., Solowjow, E., Wen, C., Ojea, J.A., Agogino, A.M.: Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects. Paper presented at the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Polydoros, A.S., Nalpantidis, L.: Survey of model-based reinforcement learning: applications on robotics. J Intell Robot Syst **86**(2), 153–173 (2017). https://doi.org/10.1007/s10846-017-0468-y

Ren, T., Dong, Y., Wu, D., Chen, K.: Learning-based variable compliance control for robotic assembly. J. Mech. Robot. (2018). https://doi.org/10.1115/1.4041331

Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics: modelling, planning and control. Springer Science & Business Media. (2010)

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M.: Others: mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484 (2016)

Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT Press, Cambridge (2018)

Tang, T.: Skill learning for industrial robot manipulators., UC Berkeley (2018)

Tang, T., Lin, H.C., Zhao, Y., Fan, Y., Chen, W., Tomizuka, M.: Teach industrial robots peg-hole-insertion by human demonstration. Paper presented at the 2016 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM, 2016–01–01 (2016)

Thomas, G., Chien, M., Tamar, A., Ojea, J.A., Abbeel, P.: Learning robotic assembly from CAD. Paper presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018–01–01

Whitney, D.E.: Quasi-static assembly of compliantly supported rigid parts. J. Dyn. Syst. Meas. Control. Trans. ASME **104**(1), 65–77 (1982). https://doi.org/10.1115/1.3149634

Williams, C.K., Rasmussen, C.E.: Gaussian processes for machine learning, vol. 2. MIT press, Cambridge (2006)

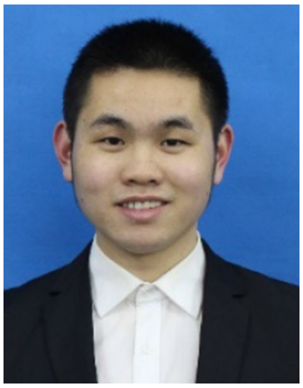
Xia, Y., Yin, Y., Chen, Z.: Dynamic analysis for peg-in-hole assembly with contact deformation. Int. J. Adv. Manuf. Technol. **30**(1–2), 118–128 (2006). https://doi.org/10.1007/s00170-005-0047-4

Xu, J., Hou, Z., Liu, Z., Qiao, H.: Compare contact model-based control and contact model-free learning: a survey of robotic peg-in-hole assembly strategies. arXiv preprint arXiv:1904.05240 (2019a)

Xu, J., Hou, Z., Wang, W., Xu, B., Zhang, K., Chen, K.: Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks. IEEE T Ind Inform **15**(3), 1658–1667 (2019b). https://doi.org/10.1109/TII.2018.2868859

Xin Zhao received his B.E. degree from School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan P.R. China, in 2013. He is currently working toward the Ph.D. degree in Huazhong University of Science and Technology, Wuhan, China. His research interests include CNC interpolation, motion planning and robotic assembly.

Huan Zhao received his B.E. degree from School of Mechanical Science and Engineering, Jilin University, Changchun, China, in 2006, and Ph.D. degree from School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2013. During 2013 - 2015, he was a Postdoctoral Researcher with the Huazhong University of Science and Technology. Since 2015, he has been with Huazhong University of Science and Technology. He is now an associate professor at Huazhong University of Science and Technology. His research interests include force control, visual servoing and machine learning with applications to robotic machining.



Pengfei Chen received the B.S. degree in mechanical engineering from Zhejiang Sci-Tech University, Hangzhou, China, in 2017. He is currently working toward the M.S degree in the department of Mechanical Engineering, Huazhong University of Science and Technology. His research interests include robotic force control and motion planning.

Han Ding received the Ph.D. degree from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1989. Supported by the Alexander von Humboldt Foundation, he was with the University of Stuttgart, Germany from 1993 to 1994. He has been a Professor at HUST ever since 1997 and is now Director of State Key Lab of Digital Manufacturing Equipment and Technology there. Dr. Ding was a "Cheung Kong" Chair Professor of Shanghai Jiao Tong University from 2001 to 2006. He was elected the member of Chinese Academy of Sciences in 2013. Dr. Ding served as an Associate Editor of IEEE Trans Automation Science and Engineering (TASE) from 2004 to 2007. Currently, he is an Editor of IEEE TASE and a Senior Editor of IEEE Robotics and Automation Letters. His research interests include robotics, multi-axis machining and control engineering.