Fuzzy Logic-Driven Variable Time-Scale Prediction-Based Reinforcement Learning for Robotic Multiple Peg-in-Hole Assembly

Zhimin Hou, *Graduate Student Member, IEEE*, Zhihu Li, Chenwei Hsu, Kuangen Zhang, and Jing Xu, *Member, IEEE*

***Abstract*— Reinforcement learning (RL) has been increasingly used for single peg-in-hole assembly, where assembly skill is learned through interaction with the assembly environment in a manner similar to skills employed by human beings. However, the existing RL algorithms are difficult to apply to the multiple peg-in-hole assembly because the much more complicated assembly environment requires sufficient exploration, resulting in a long training time and less data efficiency. To this end, this article focuses on how to predict the assembly environment and how to use the predicted environment in assembly action control to improve the data efficiency of the RL algorithm. Specifically, first, the assembly environment is exactly predicted by a variable time-scale prediction (VTSP) defined as general value functions (GVFs), reducing the unnecessary exploration. Second, we propose a fuzzy logic-driven variable time-scale prediction-based reinforcement learning (FLDVTSP-RL) for assembly action control to improve the efficiency of the RL algorithm, in which the predicted environment is mapped to the impedance parameter in the proposed impedance action space by a fuzzy logic system (FLS) as the action baseline. To demonstrate the effectiveness of VTSP and the data efficiency of the FLDVTSP-RL methods, a dual peg-in-hole assembly experiment is set up; the results show that FLDVTSP-deep Q-learning (DQN) decreases the assembly time about 44% compared with DQN and FLDVTSP-deep deterministic policy gradient (DDPG) decreases the assembly time about 24% compared with DDPG.**

***Note to Practitioners*—The complicated assembly environment of the multiple peg-in-hole assembly results in a contact state that cannot be recognized exactly from the force sensor. Therefore, contact-model-based methods that require tuning of the control parameters based on the contact state recognition cannot be applied directly in this complicated environment. Recently, reinforcement learning (RL) methods without contact state recognition have recently attracted scientific interest. However, the existing RL methods still rely on numerous explorations and a long training time, which cannot be directly applied to real-world tasks. This article takes inspiration from the manner in which human beings can learn assembly skills with a few trials, which relies on the variable time-scale predictions (VTSPs) of the environment and the optimized assembly action control strategy. Our proposed fuzzy logic-driven variable time-scale prediction-based reinforcement learning (FLDVTSP-RL) can be implemented in two steps. First, the assembly environment is predicted by the VTSP defined as general value functions (GVFs). Second, assembly action control is realized in an impedance action space with a baseline defined by the impedance parameter mapped from the predicted environment by the fuzzy logic system (FLS). Finally, a dual peg-in-hole assembly experiment is conducted; compared with deep Q-learning (DQN), FLDVTSP-DQN can decrease the assembly time about 44%; compared with deep deterministic policy gradient (DDPG), FLDVTSP-DDPG can decrease the assembly time about 24%.**

***Index Terms*—Fuzzy logic system (FLS), multiple peg-in-hole, prediction learning, reinforcement learning (RL), robotic assembly.**

Manuscript received July 7, 2020; accepted September 7, 2020. Date of publication September 30, 2020; date of current version January 6, 2022. This article was recommended for publication by Associate Editor V. Stojanovic and Editor K. Saitou upon evaluation of the reviewers' comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFE0206200; in part by the National Natural Science Foundation of China under Grant U1613205, Grant 51675291, and Grant 51935010; in part by the Beijing Municipal Natural Science Foundation under Grant L192001; in part by the Funding for Basic Scientific Research Program under Grant JCKY2018205B029; and in part by the State Key Laboratory of China under Grant SKL2020C15. *(Corresponding author: Jing Xu.)*

The authors are with the State Key Laboratory of Tribology, Beijing Key Laboratory of Precision/Ultra-Precision Manufacturing Equipment Control, Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China (e-mail: jingxu@tsinghua.edu.cn).

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

Digital Object Identifier 10.1109/TASE.2020.3024725

I. Introduction

IN RECENT years, industrial robots have become increasingly essential in many manufacturing processes, e.g., robotic assembly [1]. Multiple peg-in-hole assembly is an essential foundation in many industrial manufacturing areas from large-scale aviation components assembly to small-scale mold casting manufacturing and microscale layered assembly [2], [3].

A. Peg-in-Hole Assembly

Peg-in-hole assembly tasks are generally solved through search and insertion of two separate phases [4]. For the insertion phase, contact-model-based methods have been extensively explored, where the contact state is recognized from the assembly force, and the assembly action control is realized based on recognition results [5]. For the search phase, some search paths, e.g., a spiral path [6], [7], were often designed

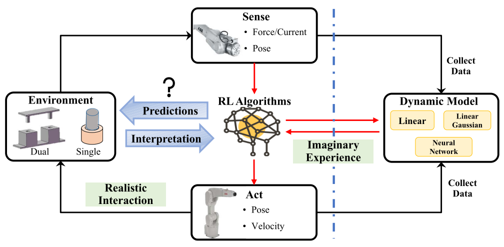


Fig. 1. RL for robotic peg-in-hole assembly.

to search the initial position automatically. Compared with active searching, some previous studies measured the initial position of holes manually based on a laser tracker [8] and vision-based approaches [9]. Although current measurements could solve the larger variability and uncertainty, the expensive device is necessary and the environmental noise may affect the precision. In addition, vision-based approaches are impractical for small uncertainties from connector and reinstallation [9] and may introduce large contact force [1]. Similar to human sensing and touching, force-based approaches are more practical to compensate for such small environmental noise [10].

The contact-model-based methods have been successful in single peg-in-hole assembly, where the contact state can be recognized exactly based on the force signals; and the corresponding controllers can be derived [10], [11]. However, multiple peg-in-hole has more complicated contact model with multiple contact points [5], and the contact state cannot be recognized exactly from the six-degree force sensor [8]; for instance, there are 26 possible contact states for the strictly parallel dual peg-in-hole assembly [12]. Without the accurate contact state recognition, the contact-model-based methods, e.g., impedance controller, require substantial efforts to tune impedance parameters [13]. In addition, the contact-model-based methods are sensitive to environmental noise and difficult to be generalized to new assembly tasks [14]. Recently, reinforcement learning (RL) has addressed the contact-rich assembly tasks without contact state recognition, e.g., gear-wheel assembly [15]. The greatest advantage of RL is the capability of learning the assembly skill (environment prediction and action control) similar to human beings, which is acquired through interactions with environments [16]. Most importantly, the learned assembly skills through RL algorithms can solve the search and insertion phase in the same scheme, which can be generalized for unseen multiple peg-in-hole assembly tasks, e.g., different clearance.

B. Related Work

RL has been used in the robotic peg-in-hole assembly tasks where the assembly action is directly obtained according to observations without contact state recognition [15], [17]–[19]. For instance, as shown in Fig. 1, the assembly actions (e.g., pose and velocity) are determined directly from the sensor signals (e.g., force and pose). Q-learning and deep Q-learning (DQN) as discrete action control RL algorithms were trained to select the reference force for the variable impedance controller on the single peg-in-hole assembly tasks [18], [20]. However, the real-world robotic assembly tasks require continuous motion; therefore, RL is expected to learn the assembly skills in a high-dimensional continuous action space. For this purpose, the deep deterministic policy gradient (DDPG) has been extensively studied to learn the deterministic continuous assembly skills through an actor network [21].

DQN and DDPG as most popular off-policy RL have shown better data efficiency performance than on-policy RL [22] via reusing the previous experiences. However, the existing RL algorithms still depend on numerous explorations, resulting in long training time and less data efficiency. To increase data efficiency in real-world assembly tasks, one way is to integrate RL with the conventional controllers, e.g., variable impedance controller-based RL [17], [21] and residual RL [14]. Another alternative way is to learn the imaginary experience from an imaginary environment fitted by a dynamic model of the environment, which can reduce the useless exploration in the realistic environment [23]. As shown in Fig. 1, the dynamic model can predict the environment in the next states given a selected action, leading to reduced realistic exploration.

There are mainly two ways to utilize the dynamic model. Dyna-like RL algorithms [24] learn the assembly skill from both the realistic and imaginary assembly environments of the dynamic model in the same scheme. On the other hand, guided policy search (GPS)-like RL algorithms learn the parameterized policies in an imaginary assembly environment from a dynamic model. Afterward, the trajectory is optimized and the dynamic model is also improved in the realistic assembly environment. GPS has shown great success in many contact-rich manipulation tasks [25]. Iterative linear-quadratic-Gaussian (ILQG) [15] as a common trajectory optimization algorithm is often explored to learn the optimal control strategy for robotic assembly tasks. However, the performance of the abovementioned algorithms largely depends on the accuracy of dynamic model. The high exploitation–exploration ratio to utilize the dynamic model easily results in producing local optima [23]. The dynamic model is often fitted by statistical techniques, e.g., linear function regression [16], linear Gaussian model [25], and neural networks. However, the statistical dynamic model is usually a one-step prediction, which can only approximate the environment in a local region. In addition, the statistical dynamic model is sensitive to the hyperparameters and environmental noise, especially the non-Gaussian noise that often appears in measurement [26]–[28]. Therefore, it is quite difficult to fit a statistical dynamic model for an accurate environment prediction (e.g., force and pose information) with the noise-laden and inconsistent sensor signals [23].

In contrast, human beings have the ability of multistep (also called variable time-scale) prediction that allows estimating the environment in a large region, resulting in the "optimal" action with less exploration. In this article, as shown in Fig. 1, we aim to propose a variable time-scale prediction (VTSP) instead of the one-step statistical dynamic model to reduce the unnecessary exploration and develop a fuzzy logic-driven variable time-scale prediction-based RL (FLDVTSP-RL) to obtain the assembly action control from the predicted environment and improve the data efficiency. To learn VTSPs, general value functions (GVFs) that expand the definition of value function through incorporating expert experience were proposed to represent the assembly environment predictions [29]. GVFs can be defined with different time scales to obtain more effective information than that achieved in one-step prediction [30]; currently, determining how to use the learned predictions to improve the action control is seldom studied. Fuzzy logic systems (FLSs) were explored to solve the robotic peg-in-hole assembly problems in which the assembly expert experience was formulated as logic rules and constraints [31]. Specifically, in this article, the predicted environment is mapped to the impedance parameters by a designed FLS; the mapped impedance parameters of the reformulated impedance action space are utilized to derive the action baseline, which can improve the data efficiency of typical RL algorithms largely.

C. Contributions

The contributions of this article are twofold. From the practical aspect, a practical RL approach is applied to learn the compliant assembly skills for multiple peg-in-hole assembly, in a manner that integrates the search and insertion phases together. First, the RL algorithm without contact state recognition is robust to environmental noises; therefore, the tuning of accurate parameters and pose measurement employed in the contact-model-based methods is not necessary. Second, to reduce training times of current popular RL algorithms in real-world tasks, an impedance action space is reformulated.

From the theoretical aspect, first, instead of the previous statistical dynamic model, the VTSPs defined as GVFs are proposed to predict the force sensor signal (assembly environment) for the subsequent assembly control. The defined GVFs can be learned by the off-policy temporary difference (TD) method efficiently. Second, a data-efficient RL framework with the reformulated impedance action space is proposed, where the baseline of action is determined by the impedance parameters mapped from the predicted environment by designed FLS.

The rest of this article is organized as follows. Section II defines the VTSPs for assembly environment in the multiple peg-in-hole assembly tasks and then learns the VTSPs by TD. Section III reformulates an impedance action space for assembly action control, whose baseline is determined by the impedance parameter mapped from the predicted environment by FLS. Section IV verifies the proposed prediction learning and assembly action control. The conclusions are drawn in Section V.

II. Variable Time-Scale Prediction Learning

As shown in Fig. 1, the environment predictions (e.g., assembly force for the potential subsequent assembly action) can reduce unnecessary exploration of RL algorithm. This section focuses on how to predict the assembly environment by the proposed VTSPs. First, we state the multiple peg-in-hole assembly problems. Second, four kinds of VTSPs are defined as GVFs. Finally, the defined GVFs are learned by off-policy TD.

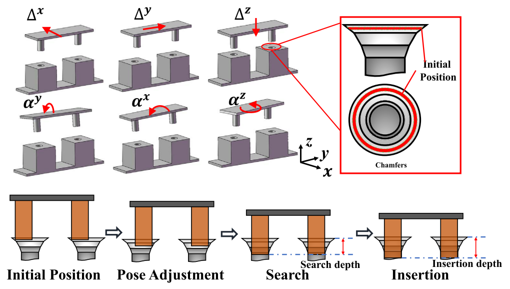


Fig. 2. Multiple peg-in-hole assembly.

A. Problem Statement

For robotic multiple peg-in-hole assembly, the holes are often fixed on the experimental table in the work space of the robot, and an external force sensor is mounted between the end-effector and pegs to measure assembly force. To simplify the presentation, the assembly force is equivalent to the output of the force sensor ignoring gravity. In search phase, the robot places the pegs to measure the pose (position/orientation) of holes and places the pegs within the initial range, as shown in Fig. 2. If the peg reaches the desired depth with the appropriate assembly force, the robot can start the insertion phase. The state $\mathbf{s}_t$ that represents the pegs during assembly expressed as

$$\mathbf{s}_t = \left[P_t^x, P_t^y, P_t^z, O_t^x, O_t^y, O_t^z\right] \in \mathbb{R}^6 \tag{1}$$

where P_t^x, P_t^y, and P_t^z and O_t^x, O_t^y, and O_t^z denote the position and orientation of pegs, respectively, and x, y, and z denote the axes of the robot base coordinate system.

The assembly control command $\mathbf{u}_t$ that adjusts the pose of the pegs in Cartesian space is shown in Fig. 2, which is expressed as

$$\mathbf{u}_t = \left[\Delta_t^x, \Delta_t^y, \Delta_t^z, \alpha_t^x, \alpha_t^y, \alpha_t^z\right] \in \mathbb{R}^6 \tag{2}$$

where Δ and α represent the translation and rotation along the corresponding axes, respectively.

B. Variable Time-Scale Prediction Definition

In RL community [16], the state value function $v_\pi(\mathbf{s}_t)$ denotes the prediction to estimate the expected cumulative reward received from environments, which is expressed as

$$v_\pi(\mathbf{s}_t) \doteq \mathbb{E}_\pi[G_t|\mathbf{s}_t] \tag{3}$$

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \tag{4}$$

where subscript π denotes the policy, the function to select the action $\mathbf{a}_t$ according to state $\mathbf{s}_t$ at time step t. $\mathbb{E}$ denotes the expectation of return over the policy π. G_t denotes the return following time t, and it is the cumulative reward of one assembly episode starting from time t to terminal time T. r_k denotes the reward to evaluate the selected action given the state $\mathbf{s}_t$, and γ is the discount function.

The state value function $v_\pi(\mathbf{s}_t)$ in (3) can calculate the prediction values only given the state $\mathbf{s}_t$ without executing the

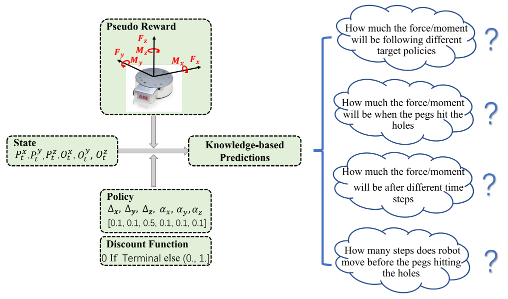

Pseudo Reward
F_z
M_z
F_y
M_y
M_x
F_x
State
$P_t^x, P_t^y, P_t^z, O_t^x, O_t^y, O_t^z$
Knowledge-based Predictions
Policy
$\Delta_x, \Delta_y, \Delta_z, \alpha_x, \alpha_y, \alpha_z$
[0.1, 0.1, 0.5, 0.1, 0.1, 0.1]
Discount Function
0 If Terminal else (0., 1.]
How much the force/moment will be following different target policies ?
How much the force/moment will be when the pegs hit the holes ?
How much the force/moment will be after different time steps ?
How many steps does robot move before the pegs hitting the holes ?

Fig. 3. Examples of VTSPs.

selected action of the whole episode. Similarly, the defined prediction $v_\pi^G(\mathbf{s}_t)$ is extended with the pseudoreward C, the policy π, and the general discount function $\gamma(\mathbf{s}_t)$, which is expressed as

$$v_\pi^G(\mathbf{s}_t) \doteq \mathbb{E}_\pi\left[\sum_{k=t}^{T} C_{t+1}\prod_{i=t+1}^{k}\gamma(\mathbf{s}_i)|\mathbf{s}_t, C_t, \gamma\right] \tag{5}$$

where the defined prediction $v_\pi^G(\mathbf{s}_t)$ basically can answer the question, e.g., "How much of the pseudoreward C_t can the robot receive from the environment following the given policy π?" To predict the subsequent assembly force, the pseudoreward C_t is defined according to the assembly force from the force sensor as

$$C_t = \Upsilon\left(F_t^x, F_t^y, F_t^z, M_t^x, M_t^y, M_t^z, \gamma(\mathbf{s}_t)\right) \tag{6}$$

where F and M are the force and moment of the force sensor, respectively.

The existing dynamic model can only predict the assembly force in the next step; however, the predictions based on GVFs can obtain VTSP with high-level features combining the peg-in-hole assembly experience. As shown in Fig. 3, four kinds of predictions (in addition to the assembly force) about the environment with the following potential assembly actions are defined through designing the policy, pseudoreward, and discount function in (5). The details of the defined VTSPs are described as follows.

1) When the pseudoreward is defined according to the assembly force as in (6), the predictions can be learned to answer the question "How much will the force/moments be when following different target policies?"
2) When the pseudoreward is defined according to the designed discount function $\gamma(\mathbf{s}_t)$ to predict the appearance of an event, e.g., "the pegs hit the holes," the predictions can be learned to answer the question "How much will the force/moments be when the pegs hit the holes?"
3) When the discount function $\gamma(\mathbf{s}_t)$ is extended to define the terminal function, the VTSPs can be defined. For instance, the assembly process will terminate when $\gamma(\mathbf{s}_t)$ is equal to zero; therefore, $1-\gamma(\mathbf{s}_t)$ represents the probability that the assembly process will terminate at the state $\mathbf{s}_t$. $\gamma(\mathbf{s}_t) = 1 - 1/T$ determines the time scale T of the predictions.
4) When the pseudoreward is defined without the assembly force, the predictions can be learned to predict "How many time steps will the robot move before the pegs hit the holes?"

C. Variable Time-Scale Prediction Learning

The defined predictions $v_\pi^G(\cdot)$ in (5) can be approximated by a linear function as $V_\pi(\cdot|\mathbf{w})$ parameterized by $\mathbf{w}$, which generally are implemented through data collecting and predictions learning. For prediction learning, the policy denotes the mapping from state to the probabilities of selected possible assembly motion directions (three translations and three rotations, see Fig. 2). Specifically, the policy π_t to define predictions is named target policy; a policy named behavior policy μ_t is utilized to generate assembly action and interact with the environment. If the predictions are learned via the data generated by the given target policy, it requires the on-policy learning [29]. For the on-policy scheme, the TD (λ) approach [16] can update the parameters $\mathbf{w}$ of V_π step by step through minimizing the TD error, which is expressed as

$$\begin{aligned}\delta_t &= C_{t+1} + \gamma_{t+1}V_\pi(\varphi(\mathbf{s}_{t+1})|\mathbf{w}_t) - V_\pi(\varphi(\mathbf{s}_t)|\mathbf{w}_t)\\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha\delta_t\mathbf{z}_t\\ \mathbf{z}_t &= \gamma_t\lambda\mathbf{z}_{t-1} + \nabla_{\mathbf{w}}V_\pi(\varphi(\mathbf{s}_t)|\mathbf{w}_t)\end{aligned} \tag{7}$$

where δ_t is the difference between the approximated value $V_\pi(\varphi(\mathbf{s}_t))$ of state $\mathbf{s}_t$ and an accurate approximation $C_{t+1} + \gamma_{t+1}V_\pi(\varphi(\mathbf{s}_{t+1}))$. $\varphi(\cdot)$ denotes the feature function of state, and $\alpha \in \mathbb{R}^+$ is the learning rate. $\mathbf{z}_t$ is the eligibility trace as a short-term memory to store the TD error, and $\lambda \in [0, 1]$ controls the length of the eligibility trace.

To learn different predictions from the single source data in parallel, we implement an efficient prediction learning scheme called off-policy [29]. For instance, based on the off-policy learning, the predictions defined by the dangerous target policies can be learned from the data generated by a safe behavior policy or the human demonstration, which is essential for real-world assembly tasks. For off-policy scheme, the parameters $\mathbf{w}$ are updated via the off-policy TD (λ) through combining with a correction term ρ_t [29] as

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha\delta_t\mathbf{z}_t\\ \mathbf{z}_t &= \rho_t(\gamma_t\lambda\mathbf{z}_{t_1} + \nabla_{\mathbf{w}}V_\pi(\varphi(\mathbf{s}_t)|\mathbf{w}_t))\end{aligned} \tag{8}$$

where the correction term ρ_t is calculated by the importance sampling ratio $\rho_t = ((\pi(\mathbf{a}_t|\mathbf{s}_t))/(\mu_(\mathbf{a}_t|\mathbf{s}_t)))$. In summary, the off-policy scheme is more general, the on-policy scheme is a special case of off-policy scheme, and the implementation of VTSP learning is summarized in Algorithm 1. All predictions with different target policies can be learned from the collected data from the same behavior policy.

III. Variable Time-Scale Prediction-Based RL for Assembly Control

This section focuses on how to utilize the learned VTSPs for assembly action control. First, the preliminaries of RL algorithms for assembly action control are introduced and the impedance action space is reformulated. Second, FLS

Algorithm 1 VTSP Learning

```
1: Initialize replay buffer $\mathcal{D}$; number of predictions $M$ and
   weight of predictions $\varpi$
2: Collect data
3: for $t = 0, \ldots, N$ do
4:   Initialize state $\mathbf{s}_0$; $done = 0$
5:   while $done = 0$ do
6:     Observe state $\mathbf{s}_t$
7:     Get action $\mathbf{a}_t = \mu(\mathbf{s}_t)$ and execute assembly control
       command $\mathbf{u}_t \leftarrow \S(\mathbf{u}_t^e, \mathbf{a}_t | \mathbf{K}_p, \mathbf{K}_d)$ (12)
8:     Observe next state $\mathbf{s}_{t+1}$
9:     Get pseudoreward $C_t^1, C_t^2, C_t^3, \ldots$; discount rate
       $\gamma^1, \gamma^2, \ldots$
10:    Store $(\mathbf{s}_t, \mathbf{a}_t, [C_t^1, C_t^2, C_t^3, \ldots], [\gamma^1, \gamma^2, \ldots], \mathbf{s}_{t+1})$ in $\mathcal{D}$
11:    $done = 1$ if search phase finish else $done = 0$
12:  end while
13: end for
14: Learn predictions
15: for $i = 0, \ldots, M$ do
16:   Initialize $V_\pi^i$ with weight $\mathbf{w}_0^i$, target policy $\pi^i$, feature $\varphi$,
      discount rate $\gamma^i$ and pseudoreward $C^i$
17:   for $t = 0, \ldots, T$ do
18:     Sample $(\mathbf{s}_t, \mathbf{a}_t, C_t^i, \mathbf{s}_{t+1})$ and calculate $\gamma_t^i$
19:     $\delta_t^i = C_{t+1}^i + \gamma_t^i V_\pi^i(\varphi(\mathbf{s}_{t+1})|\mathbf{w}_t^i) - V_\pi^i(\varphi(\mathbf{s}_t)|\mathbf{w}_t^i)$
20:     $\rho_t^i = \frac{\pi^i(\mathbf{a}_t|\mathbf{s}_t)}{\mu(\mathbf{a}_t|\mathbf{s}_t)}$, $\mathbf{w}_t^i$ updated by TD($\lambda$) as (7)
21:   end for
22: end for
23: Output: weights of predictions $\varpi = [\mathbf{w}^1, \mathbf{w}^2, \mathbf{w}^3, \ldots.]$
```

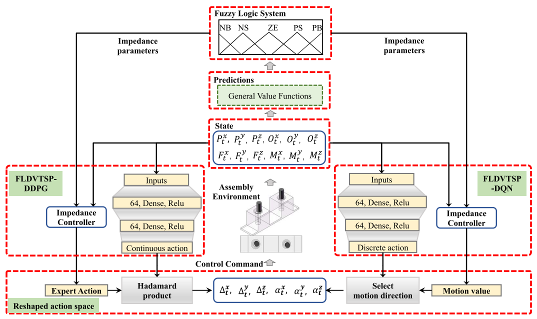


Fig. 4. Schematic of FLDVTSP-RL algorithms (FLDVTSP-DQN and FLDVTSP-DDPG).

is used to map the predicted environment to the impedance parameter in an impedance action space as the baseline for RL policy learning. Finally, as shown in Fig. 4, based on the two most popular RL algorithms (DQN [18] and DDPG [19]), FLDVTSP-DQN and FLDVTSP-DDPG are implemented and analyzed theoretically.

A. Preliminary and Impedance Action Space Reformulation

For assembly action control based on RL algorithms, the assembly policies are trained to select the assembly action $\mathbf{a}_t$ through maximizing the following objective as:

$$\max_{\mathbf{a}_t} \mathbb{E}_{\tau \sim \mu}\left[\sum_{t=0}^{\mathrm{T}} \mathcal{R}_t(\mathbf{s}_t, \mathbf{a}_t)\right] \tag{9}$$

where $\tau = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_2, \ldots, \mathbf{s}_\mathrm{T}, \mathbf{a}_\mathrm{T}\}$ represents the assembly trajectory induced by the behavior policy $\mu(\mathbf{a}_t|\mathbf{s}_t)$. In addition to the position and orientation, the state $\mathbf{s}_t \in \mathbb{R}^{12}$ for assembly action control also consists of the force and moment, which is expressed as

$$\mathbf{s}_t = [\underbrace{P_t^x, P_t^y, P_t^z, O_t^x, O_t^y, O_t^z}_{\text{position and orientation}}, \underbrace{F_t^x, F_t^y, F_t^z, M_t^x, M_t^y, M_t^z}_{\text{force and moment}}]. \tag{10}$$

Reward $\mathcal{R}_t$ is designed to award the high assembly speed and penalize the assembly action resulting in a large force, which is expressed as

$$\mathcal{R}_t = \begin{cases} -0.1, & \text{done} = \text{False and safe} = \text{True} \\ 1 - \dfrac{k_T}{k_{\max}}, & \text{done} = \text{True and safe} = \text{True} \\ -1, & \text{safe} = \text{False} \end{cases} \tag{11}$$

where k_T represents the final number of assembly steps and $k_{\max}$ denotes the maximum assembly steps within one episode. If the assembly task can be completed within $k_{\max}$ steps, the robot will receive a positive reward [0, 1). At each time step t, if the assembly force or moment exceeds the setting of the safe threshold, the robot will receive a large negative reward -1, and otherwise, a small negative reward -0.1 (note that this value needs to be well designed to balance the assembly speed and assembly force).

A natural action space $\mathbf{a}_t \in \mathcal{A}$ for RL policy is the Cartesian space as shown in Fig. 2, where only the pose of pegs is controlled, leading to a large assembly force and assembly parts damage. For real-world assembly tasks, to reduce the training time and avoid dangerous exploration via considering the effects of assembly force, as shown in Fig. 4, the action space for FLDVTSP-DDPG and FLDVTSP-DQN is reformulated into the impedance action space $\mathbf{a}_t \in \mathcal{A}^c$, which is expressed as

$$\begin{aligned} \mathbf{u}_t &\leftarrow \S\left(\mathbf{u}_t^e, \mathbf{a}_t \middle| \mathbf{K}_p, \mathbf{K}_d\right) \\ \mathbf{u}_t^e &= \mathbf{K}_p \circ \mathbf{e}_t + \mathbf{K}_d \circ \dot{\mathbf{e}}_t \\ \mathbf{e}_t &= \mathcal{F}_d - \mathcal{F}_t \end{aligned} \tag{12}$$

where $\mathcal{F}_t = [F_t^x, F_t^y, F_t^z, M_t^x, M_t^y, M_t^z]$ and $\mathcal{F}_d = [F_d^x, F_d^y, F_d^z, M_d^x, M_d^y, M_d^z]$ denote the measured assembly force and reference assembly force corresponding to the axes of the robot base coordinate system in Fig. 2, respectively; $\mathbf{e}_t$ denotes the difference between the reference assembly force and the measured assembly force, and $\dot{\mathbf{e}}_t$ is the differential of $\mathbf{e}_t$. Basically, given the reference assembly force $\mathcal{F}_d$, an expert assembly command $\mathbf{u}_t^e$ is derived based on the impedance controller, which can enable robot to achieve a safe compliant interaction. The action derived from the RL policy is applied to revise the derived expert assembly commands. The performance of the proposed impedance action space largely depends on the impedance parameters $\mathbf{K}_p$ and $\mathbf{K}_d$. Therefore, the data efficiency will be largely affected by the selected impedance parameters, and an FLS can be designed to obtain the impedance parameters of the impedance action space.

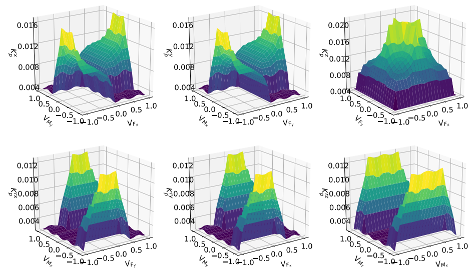


Fig. 5. Surface view of input–output for six fuzzy inferences.

TABLE I
DEFINITION OF FLS

Inputs	Outputs
V_{F_x}, V_{M_y}	$K_p^x \in [0,\ 0.015]$
V_{F_y}, V_{M_x}	$K_p^y \in [0,\ 0.015]$
V_{F_x}, V_{F_y}	$K_p^z \in [0,\ 0.02]$
V_{M_x}, V_{F_y}	$K_p^{rx} \in [0,\ 0.015]$
V_{M_y}, V_{F_x}	$K_p^{ry} \in [0,\ 0.015]$
V_{M_z}, V_{M_x}	$K_p^{rz} \in [0,\ 0.015]$

B. FLS Design

In practice, an FLS is designed to only determine the proportional parameters $\mathbf{K}_p = [K_p^x, K_p^y, K_p^z, K_p^{rx}, K_p^{ry}, K_p^{rz}]$ in the impedance action space (12), and $\mathbf{K}_d$ is directly derived from $\mathbf{K}_p$. The input of the FLS is the learned VTSPs $\widehat{\mathcal{F}}_t = V_\pi(\mathbf{s}_t|\varpi)$ via Algorithm 1. As shown in Fig. 5, the fuzzy rules of the FLS are employed to map the learned predictions about the subsequent assembly force to six proportional parameters $\mathbf{K}_p$ of the impedance action space. Basically, the optimized impedance controller based on the learned predictions provides the high-level goals, which can reduce the useless exploration and provide the stable target implicitly. As shown in Table I, FLS consists of six fuzzy inferences corresponding to six proportional parameters in the impedance action space; and every fuzzy inference consists of two kinds of predictions corresponding to two force sensor signals as input and one proportional parameter corresponding to an element of the assembly action as output. Specifically, the designed FLS is implemented in the following three steps.

1) The input and output of each fuzzy inference are partitioned into various fuzzy sets through the triangular membership function in the fuzzification process. The input of each fuzzy interference is divided into five fuzzy sets, and each fuzzy set is associated with linguistic terms, e.g. "NB," "NS," "ZE," "PS," and "PB." The output is divided into three fuzzy sets with respect to "NS," "ZE," and "PS."
2) The fuzzy rules are plotted in Fig. 5. Each fuzzy inference includes 25 If-Then rules in the following form: $R^{(i)}$: if $V_{F_x}^{(i)}$ is "NB" and $V_{M_y}^{(i)}$ is "NB," then $K_p^{x(i)}$ is "NS." As shown in Table I, all inputs of the fuzzy inference system are normalized in $[-1, 1]$, and the range of the output generally needs to be predefined according to the real-world assembly settings.
3) The weighted average defuzzification approach [32] is utilized to calculate the output of each proportional parameter, e.g., $K_p^x = \Theta^{\mathrm{T}}\Phi$, where $\Theta^{\mathrm{T}} = [K_p^{x(1)}, K_p^{x(2)}, \dots, K_p^{x(25)}]$ is the output of the ith fuzzy rule and $\Phi = [\mu_M(K_p^{x(1)}), \dots, \mu_M(K_p^{x(25)})]/(\sum_{i=1}^{25} \mu_M(K_p^{x(i)}))$ is vector to calculate the corresponding weight of each fuzzy rule by the triangular membership function $\mu_M(\cdot)$.

C. FLDVTSP-DQN

DQN is generally used for the discrete assembly action control in the peg-in-hole assembly tasks [18]. FLDVTSP-DQN is designed to improve the data efficiency of typical DQN, and the discrete impedance action space $\mathcal{A}^c$ is defined as

$$\begin{aligned}\mathcal{A}^c = [&[F_d^x,\ 0,\ F_d^z,\ 0,\ 0,\ 0]\\ &[0,\ F_d^y,\ F_d^z,\ 0,\ 0,\ 0]\\ &[0,\ 0,\ F_d^z,\ 0,\ 0,\ 0]\\ &[0,\ 0,\ F_d^z,\ M_d^x,\ 0,\ 0]\\ &[0,\ 0,\ F_d^z,\ 0,\ M_d^y,\ 0]\\ &[0,\ 0,\ F_d^z,\ 0,\ 0,\ M_d^z]]\end{aligned} \tag{13}$$

where the output action $\mathbf{a}_t$ from FLDVTSP-DQN is a one-shot vector to select the motion direction (except the translation along z-axis, see Fig. 2). The assembly control command $\mathbf{u}_t \leftarrow \S(\mathbf{u}_t^e, \mathbf{a}_t|\mathbf{K}_p, \mathbf{K}_d)$ corresponding to the selected motion direction can be calculated as $\mathbf{a}_t \circ \mathbf{u}_t^e$, where the proportional parameters $\mathbf{K_p}$ are obtained based on the predicted environment about the subsequent assembly force by the designed FLS. Compared to typical DQN [18], the proposed FLDVTSP-DQN can perform impedance-based assembly behaviors. Basically, the reformulated action space can provide the baseline for assembly policy learning, which can reduce useless exploration and make robot only learn the residual skills besides the baseline. In addition, the reformulated action space will be optimized according to the predicted environments by GVFs.

For multiple peg-in-hole assembly, it is quite difficult to recognize the contact force exactly from the six-degree force sensor due to the complicated coupling relations. The "optimal" motion direction can not only reduce the contact force along this direction but also eliminate the relevant contact force along another direction. At each sample step, the "optimal" motion direction is selected according to the action-value network as

$$\mathbf{a}_t = \max_{\mathbf{a}}\ Q^*(\mathbf{s}_t, \mathbf{a}|\theta) \tag{14}$$

where Q^* means that motion along this direction has the largest probability to complete the assembly with higher reward. The action-value network parameterized by θ is trained through minimizing the loss as

$$\mathcal{L} = \frac{1}{|B|}\sum_m (y_m - Q(\mathbf{s}_m, \mathbf{a}_m|\theta))^2 \tag{15}$$

where $y_m = \mathcal{R}_{m+1} + \gamma \max_{\mathbf{a}} Q(\mathbf{s}_{m+1}, \mathbf{a}|\widetilde{\theta})$ is calculated by a target action-value network with parameters $\widetilde{\theta}$. We applied the same training techniques *experience replay*, *target network*, and *normalization* as in [18] and [33]. In addition, to improve the stability of FLDVTSP-RL algorithms, at each training step, a minibatch of transitions B sampled from the replay buffer uniformly $\mathcal{D}$ is used to update the action-value network. The details of training FLDVTSP-DQN are summarized in Algorithm 2.

Algorithm 2 FLDVTSP-RL

```
1: Input learned predictions $\varpi = [\mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^6]$
2: Initialize network and target network; replay buffer $\mathcal{D}$
3: for each episode do
4:   Initialize state of pegs $\mathbf{s}_0$ randomly in limited range
5:   for each sample step do
6:     Select action $\mathbf{a}_t$
7:     Select parameters $\mathbf{K_p} = \mathbf{FLS}(\mathbf{s}_t, \varpi)$
8:     Calculate command $\mathbf{u}_t \leftarrow \S(\mathbf{u}_t^e, \mathbf{a}_t | \mathbf{K}_p, \mathbf{K}_d)$
9:     Execute $\mathbf{u}_t$, observe reward $\mathcal{R}_{t+1}$ and next state $\mathbf{s}_{t+1}$
10:    Store $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, r_t, \mathbf{s}_{t+1})$ in $\mathcal{D}$
11:    for each training step do
12:      Sample a minibatch of $B$ transitions uniformly
         $(\mathbf{s}_m, \mathbf{a}_m, r_m, \mathbf{s}_{m+1})$ from $\mathcal{D}$
13:      Update parameters of all networks
14:      Update target network
15:    end for
16:  end for
17: end for
```

D. FLDVTSP-DDPG

DDPG is a commonly used RL algorithm for continuous assembly action control in peg-in-hole assembly tasks [13], [21]. To improve the data efficiency of DDPG, FLDVTSP-DDPG is designed with a reformulated impedance action space $\mathbf{a}_t \in \mathcal{A}^c$, and the assembly control command can be derived as

$$\mathbf{u}_t = \text{clip}\{\mathbf{a}_t, -b, +b\} \circ \mathbf{u}_t^e + \mathbf{u}_t^e \tag{16}$$

where $\mathbf{u}_t^e$ denotes the expert action derived from the impedance controller as in (12) with proportional parameters $\mathbf{K_p}$ mapped from the prediction environment about the subsequent assembly force by the designed FLS. Compared with the FLDVTSP-DQN, FLDVTSP-DDPG also needs to learn an actor network parameterized by θ^μ, as shown in Fig. 4; at each sample step t, the action is calculated from the actor network $\mathbf{a}_t = \mu(\mathbf{s}_t|\theta^\mu) + \mathcal{N}_t$, $\mathbf{a}_t \in [-1, 1]^6$, $\mathcal{N}_t$ is the exploration noise, and b is utilized to regulate the confidence of the actor network. Likewise, our proposed FLDVTSP-DDPG with the reformulated action space as in (16) can learn the impedance-based assembly behaviors, which can reduce the early-stage random learning and also improve the stability performance without unpredicted exploration.

At each training step, a minibatch of transitions B are sampled from the replay buffer uniformly $\mathcal{D}$. First, the action-value network is updated through minimizing the error similar to that in (15); however, the target value is calculated as

$$y_m = \mathcal{R}_{m+1} + \gamma\, Q(\mathbf{s}_{m+1}, \mu(\mathbf{s}_{m+1}|\widetilde{\theta^\mu})|\widetilde{\theta^Q}) \tag{17}$$

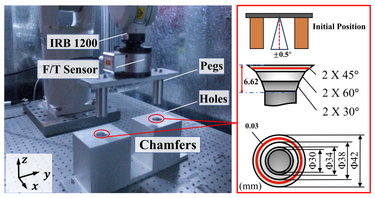


Fig. 6. Robotic dual peg-in-hole assembly platform.

where $\mu(\mathbf{s}_{m+1}|\widetilde{\theta^\mu})$ is calculated via the target actor network. Afterward, the actor network is updated based on the deterministic policy gradient, which is expressed as

$$\nabla_{\theta^\mu} J = \frac{1}{|B|}\sum_m \nabla_a Q(\mathbf{s}_m, \mu(\mathbf{s}_m)|\theta^Q)\nabla_{\theta^\mu}\mu(\mathbf{s}_m|\theta^\mu) \tag{18}$$

where the training settings are the same as those in [34]. The details of training FLDVTSP-DDPG can also refer to Algorithm 2.

IV. Experiments

In this section, first, a dual peg-in-hole assembly experimental system is set up, as shown in Fig. 6. Second, to verify the effectiveness of VTSPs learned by GVFs, three kinds of GVFs are evaluated. To verify the efficiency of the FLDVTSP-RL algorithms (including FLDVTSP-DQN and FLDVTSP-DDPG), two experiments are conducted. Finally, to evaluate the generalization performance, the new assembly tasks with the tight clearance are applied to evaluate the performance of learned assembly policies.

A. Experiment Setup

An ABB six-degree F/T sensor and the dual pegs are mounted on the end-effector of the robot, the dual holes are fixed on the experimental table, and the definition of the robot base coordinate system is shown in Fig. 6. In this article, all the pegs and holes were both machined from Aluminum 6061. The diameter of the pegs and holes is 29.96 and 30.00 mm, respectively. The distance between the two pegs' axes and the two holes' axes is 200.0 mm. Furthermore, to simulate a complicated assembly environment, three different size chamfers are designed, as shown in Fig. 6. The search phase aims to reach the desired depth along the z-axis of the robot base coordinate system within the safe threshold [80 N, 5 Nm] and then start the insertion phase.

B. VTSP Learning

As shown in Algorithm 1, the predictions are learned via two stages: data collecting and VTSP learning.

1) Data Collecting: The behavior policy μ_t is denoted as

$$\mu_t(\cdot|\cdot) = [0.1,\ 0.1,\ 0.5,\ 0.1,\ 0.1,\ 0.1] \tag{19}$$

which means that at each time step t, the robot has the probability of 0.5 to move down along the z-axis and has equal probability 0.1 to move along the remaining directions. Specifically, the motion magnitude is calculated via impedance

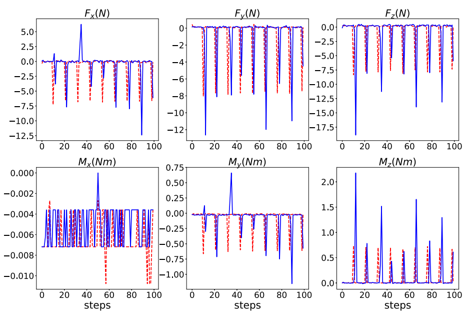


Fig. 7. F/T sensor signals. Red lines: fixed initial pose. Blue lines: random initial pose.

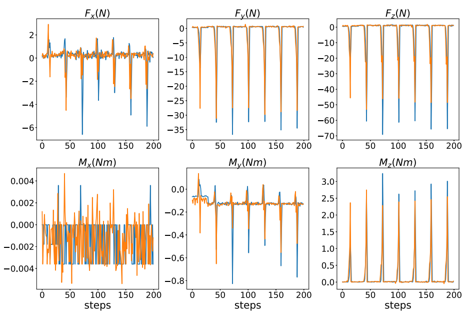


Fig. 8. Predictions of how much assembly force will be when pegs hit holes. Sky-blue lines: true values. Orange lines: learned results.

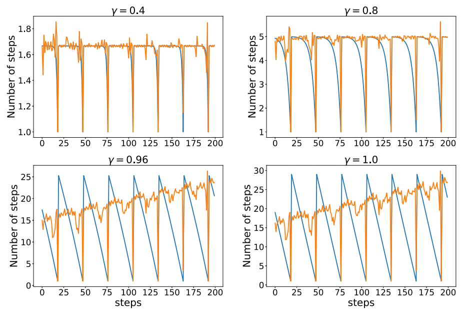


Fig. 9. Different time-scale predictions. Sky-blue lines: true values for predictions. Orange lines: learned results.

controller as shown in (12). The behavior policy μ_t is sampled for 1000 time steps to execute and collect data. To simulate the environmental noise, as shown in Fig. 6, the initial position of the pegs is randomly set to be uniform within $[-0.03, 0.03]$ mm, and the initial orientation of the pegs is randomly set to be uniform within $[-0.5°, 0.5°]$. In addition, the pegs will be reset when it has reached the desired depth or the F/T sensor signals exceed the set safe threshold. The assembly force measured from the F/T sensor during interactions is plotted in Fig. 7. The assembly tasks will become complicated with the random initial pose, shown as the blue lines, compared to a fixed initial pose, shown as the red lines.

2) Learning Predictions: Six linear functions are updated to approximate the predictions of the assembly force signals. The features of the state of the linear function are derived by the tile coding [29], [30]. For every dimension of the state, ten tilings are used and every dimension is hashed to a vector of 1024 dimensions. To eliminate the impact of the different physical units, the assembly force measured from the F/T sensor is normalized in $[-1, 1]$. Three kinds of predictions are evaluated as follows. For a clear description, we only plot the last 200 time steps for the following results.

First, the predictions defined via the pseudoreward C_t^i are utilized to approximate how much the six force/moments will be when the pegs hit the holes. The pseudoreward C_t^i is denoted as

$$C_t^i = \left(1 - \gamma_t^i\right)\mathcal{F}_t^i \tag{20}$$

where $\mathcal{F}_t^i$ denotes the assembly force of the i th prediction at time step t and γ_t^i will be 0 when the pegs have hit the holes; otherwise, it is 1. The true values for the predictions are calculated by (3). The learned six different predictions to approximate the assembly force can predict how much the assembly force will be when the pegs hit the holes. As shown in Fig. 8, the learned predictions that are shown as orange lines are accurate enough compared with the true value shown as the sky-blue line.

Second, we verify the relationship between the time scale of predictions and discount rate γ. In addition to the assembly force $\mathcal{F}_t$, we design the handcrafted pseudoreward following the same behavior policy in (19) with a different discount rate γ, corresponding to the different time scales of predictions. For instance, the predictions are utilized to predict the period of an appearance of event $\aleph$, e.g., when the pegs hit the holes, which is expressed as

$$C_t^i = \begin{cases} 1, & \aleph \text{ is done} \\ 0, & \text{else} \end{cases} \tag{21}$$

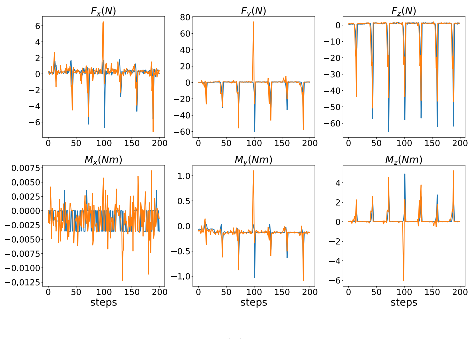
when $\gamma = 0.96$, the true value of learned predictions is equal to the time scale 25. As shown in Fig. 9, likewise, we plot four different time-scale predictions with a different discount rate γ such that the predictions with a smaller discount rate γ are easier to learn, and the predictions with a larger discount rate γ are not as accurate as the small one.

Third, to demonstrate that the predictions with two different target policies can be learned through the off-policy scheme (see Algorithm 1), two different target policies are denoted as

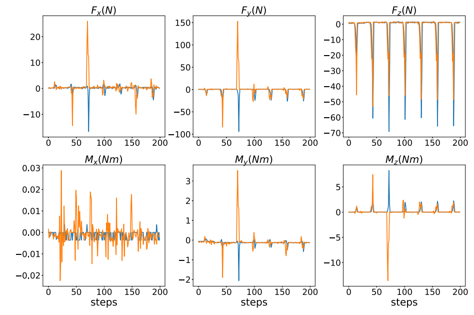
$$\pi_t^1(\cdot|\cdot) = [0.3,\ 0.1,\ 0.3,\ 0.1,\ 0.1,\ 0.1] \tag{22}$$
$$\pi_t^2(\cdot|\cdot) = [0.05,\ 0.4,\ 0.4,\ 0.05,\ 0.05,\ 0.05] \tag{23}$$

which have the different probabilities of selecting six motion directions. Likewise, the predictions are utilized to predict the subsequent assembly, if the robots follow the actions generated by the target policy π_t^1 [it has a greater probability to translate along the x-axis compared to the behavior policy in (19)] (see Fig. 8). However, these predictions can be learned using the collected data from the behavior policy in (19) instead of executing actions from the corresponding target policies.

The predictions about the assembly force corresponding to the target policy in (22) and (23) can be effectively learned via the off-policy scheme in Algorithm 1 (see Fig. 10). The learned predictions are accurate at most of the time steps compared with the true data shown in Fig. 10. More importantly,



(a)



(b)

Fig. 10. Predictions with different target policies. Sky-blue lines: true values for predictions. Orange lines: learned results. (a) Target policy π_t^1. (b) Target policy π_t^2.

via the off-policy scheme, the different target policies can be utilized to strengthen the probability of one motion direction, which will be utilized to obtain the accurate prediction about how much the assembly force will be if the pegs move along this direction. In addition, the target policies can also be utilized to execute the dangerous actions that might result in a large contact force. Therefore, via the off-policy scheme, more useful information about the assembly environment can be learned without the dangerous action execution. However, Fig. 10(b) shows that the error of the learned predictions will become larger when the target policy in (23) is far from the behavior policy. It means that the importance ratio ρ_t cannot compensate for the gap between these two policies.

C. Control Performance and Analysis

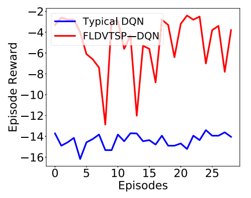
1) FLDVTSP-DQN Versus DQN: FLDVTSP-DQN and typical DQN [18] have the same network architecture with two fully connected layers (see Fig. 4). Both FLDVTSP-DQN and typical DQN are trained for 100 episodes with the same hyperparameters settings as shown in Table II and the same reward signal as in (11). To eliminate the impact of different physical units, every dimension of the state in (10) given to the FLDVTSP-DQN and the typical DQN is normalized within $[-1, 1]$. In addition, the network training techniques, e.g., the target network and experience replay are used to make the training stable and efficient. The FLS of the FLDVTSP-DQN shown in Algorithm 2 is designed according to Section II. All GVFs to predict the subsequent assembly force are also normalized within $[-1,\ 1]$ and input to the FLS. The range of proportional parameters from the FLS output for impedance action space is shown in Table I.

TABLE II
HYPERPARAMETERS OF DQN

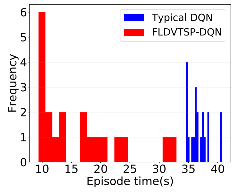
Parameters	Value
minibatch size	64
learning rate	1e-3
target net update rate	1e-3
buffer size	1e5

TABLE III
HYPERPARAMETERS OF DDPG

Parameters	Value
minibatch size	64
actor learning rate	1e-4
critic learning rate	1e-3
target net update	1e-3
buffer size	1e5

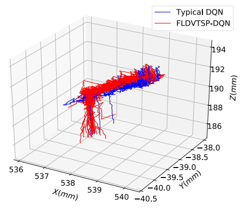


(a)

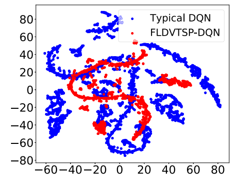


(b)

Fig. 11. FLDVTSP-DQN and typical DQN. (a) Episode reward. (b) Episode assembly time.



(a)



(b)

Fig. 12. FLDVTSP-DQN and typical DQN. (a) Assembly trajectories. (b) Action space (dimensionality is reduced by t-SNE for visualization).

To evaluate the robustness against environmental noise, the learned assembly policies of FLDVTSP-DQN and typical DQN are applied to perform the same assembly tasks for 30 times. Fig. 11(a) shows that FLDVTSP-DQN can obtain more rewards within one episode and Fig. 11(b) shows that FLDVTSP-DQN decreases the assembly time about 44% compared with typical DQN. The 30 times assembly trajectories are plotted in Fig. 12(a), FLDVTSP-DQN can achieve an average deeper insertion depth compared with typical DQN. The season is that the learned predictions can reduce the unnecessary exploration during assembly [see Fig. 12(b)], FLDVTSP-DQN only needs to explore the narrow action space compared with typical DQN. Obviously, the proposed impedance action space enables the robots to explore better policies efficiently. Furthermore, the assembly force of both FLDVTSP-DQN and typical DQN for 30 times assembly is shown in Fig. 13. Both FLDVTSP-DQN and typical DQN can complete the assembly within the safety threshold, but FLDVTSP-DQN can achieve deeper insertion depth with fewer steps.

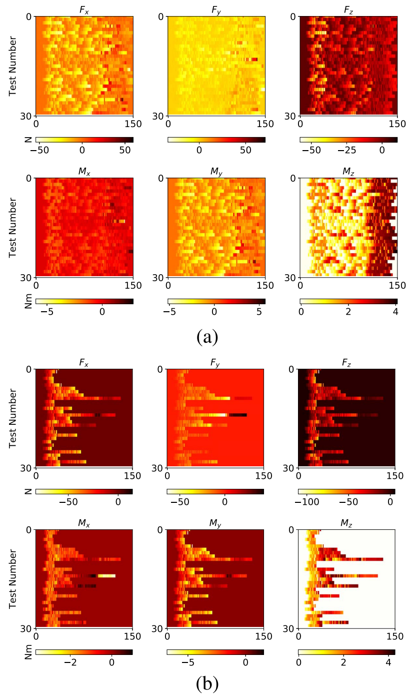


Fig. 13. Comparison of force. (a) Typical DQN. (b) FLDVTSP-DQN.

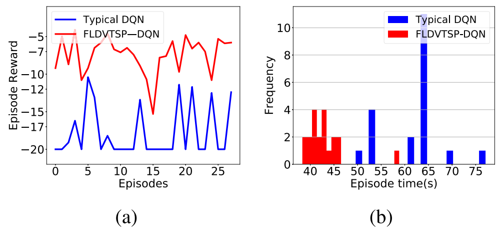


Fig. 14. Generalization performance of FLDVTSP-DQN and typical DQN. (a) Episode reward. (b) Episode assembly time.

To test the generalization performance in new assembly tasks (diameters of peg and hole are 29.98 and 30.00 mm, respectively, H7h7 tolerance), the learned assembly policies by FLDVTSP-DQN and typical DQN are implemented for 30 times. As shown in Fig. 14, obviously, typical DQN sometimes cannot complete the assembly tasks within 200 time steps and cannot achieve a 100% success rate. However, FLDVTSP-DQN can complete the assembly tasks with less time successfully and achieve the 100% success rate.

2) FLDVTSP-DDPG Versus DDPG: FLDVTSP-DDPG and typical DDPG are set the same network architecture as shown in Fig. 4, and and the action threshold b in (16) is set as 0.2. Likewise, for fair comparison, FLDVTSP-DDPG and typical DDPG are trained for 100 episodes with the same hyperparameters settings as in [13] and [21] (see Table III). Likewise, to evaluate the robustness against the environmental noise, the learned assembly policies of FLDVTSP-DDPG and typical DDPG are applied to perform the same assembly tasks for 30 times. As shown in Fig. 15(a), FLDVTSP-DDPG can receive more reward within one episode than typical DDPG and Fig. 15(b) shows that FLDVTSP-DDPG decreases the assembly time about 24% compared with typical DDPG. The assembly trajectories of FLDVTSP-DDPG and typical

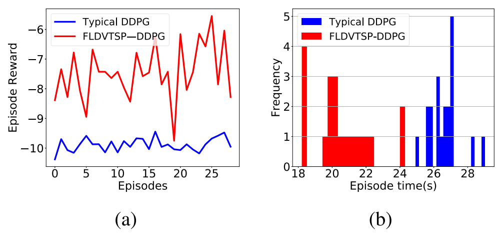


Fig. 15. FLDVTSP-DDPG and typical DDPG. (a) Episode reward. (b) Episode assembly time.

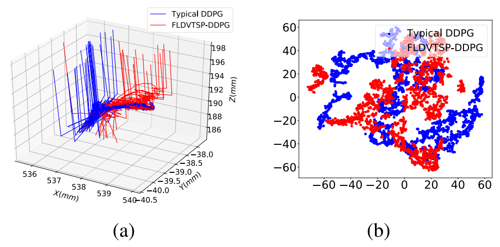


Fig. 16. FLDVTSP-DDPG and typical DDPG. (a) Assembly trajectories. (b) Action space (dimensionality is reduced by t-SNE for visualization).

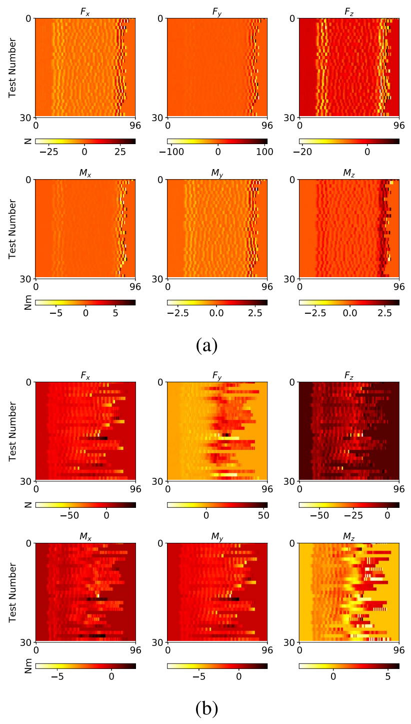


Fig. 17. Comparison of force. (a) Typical DDPG. (b) FLDVTSP-DDPG.

DDPG are plotted in Fig. 16(a); clearly, FLDVTSP-DDPG can complete the search phase and achieve the deeper insertion depth in the insertion phase, as the red lines. As shown in Fig. 17, the assembly force of both FLDVTSP-DDPG and DDPG is within the set safety threshold; however, FLDVTSP-DDPG can complete the assembly tasks with fewer steps and smaller assembly force. Likewise, as shown in Fig. 16(b), FLDVTSP-DDPG only needs to explore the narrow action space without unnecessary exploration; obviously, the reformulated impedance action space enables FLDVTSP-DDPG to learn more efficiently.

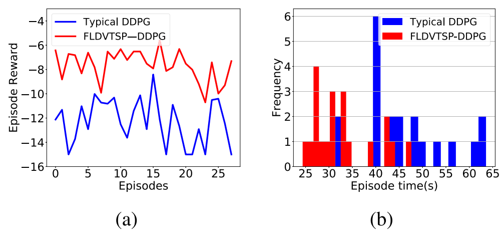


Fig. 18. Generalization performance of FLDVTSP-DDPG and typical DDPG. (a) Episode reward. (b) Episode assembly time.

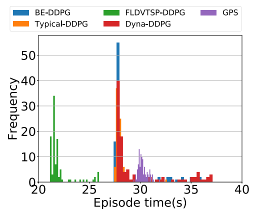


Fig. 19. Performance of assembly time compared with model-based RL.

Likewise, to test the generalization performance in new assembly tasks (diameters of peg and hole are 29.98 and 30.00 mm, respectively, H7h7 tolerance), the learned assembly policies by FLDVTSP-DDPG and typical DDPG are implemented for 30 times. As shown in Fig. 18, it is more difficult to complete the tight clearance assembly. Although both FLDVTSP-DDPG and typical DDPG can achieve a 100% success rate within 200 time steps, obviously, compared with typical DDPG, FLDVTSP-DDPG can receive more rewards and decrease the assembly time about 24.7%.

3) FLDVTSP-DDPG Versus Model-Based RL: To demonstrate the effectiveness of our proposed FLDVTSP-DDPG compared with previous model-based RL, four RL approaches assisted by the fitted dynamic model are utilized to perform the peg-in-hole assembly tasks, including typical DDPG, Dyna-DDPG [24], BE-DDPG (DDPG with better exploration based on dynamic model), and GPS [15]. The transition samples interacting with environment are collected and stored in the replay buffer $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$. Afterward, the data are sampled from $\mathcal{D}$ to fit the dynamic model.

We apply three commonly used statistical techniques to fit the dynamic model via the collected data in replay buffer $\mathcal{D}$: linear approximation, nonlinear approximation with a neural network, and time-varying linear-Gaussian approximation. Therefore, for a fair comparison, all aforementioned model-based RL algorithms with the linear Gaussian based dynamic model are trained with the same settings as typical DDPG and FLDVTSP-DDPG. All the trained assembly policies are evaluated for 100 times. As shown in Fig. 19, FLDVTSP-DDPG can complete one episode of assembly with less time, leading to the highest data efficiency. The dynamic model with bias may mislead the assembly action learning, e.g., GPS and BE-DDPG; obviously, the learned predictions can enable the FLDVTSP-DDPG to avoid the trap during training.

V. Conclusion

This article proposes the FLDVTSP-RL for the multiple peg-in-hole assembly as solutions that are more data efficient than the existing RL algorithms. First, GVFs are used to learn the VTSPs of the subsequent assembly. The impedance action space for FLDVTSP-RL is reformulated, in which the baseline is calculated based on the impedance parameters mapped from the predicted environment by the designed FLS. The results of a dual peg-in-hole assembly experiment demonstrate that the predictions of the subsequent assembly force can be learned through GVFs effectively, and our proposed FLDVTSP-RL algorithms can perform the assembly tasks more efficiently than the typical RL algorithms. Note that parameters tuning according to specific tasks may improve the final performance. Our idea can also combine with other RL methods, e.g., NAF [35] and SAC. In addition, it is easy to apply the proposed FLDVTSP-RL to solve other similar contact-rich problems [9], e.g., power plugs [19] and USB connectors [36] referring to our implementation.

Some limitations of our proposed FLDVTSP-RL algorithms can be solved in the future. Currently, the FLS to map the learned predictions by GVFs is designed based on prior knowledge. Further work could focus on updating the FLS following the nature-inspired parameter tuning approaches [37]–[39]. For instance, the way to map the learned predictions can be formulated as an optimization problem under the constraints of the learned predictions [39], and then, impedance parameters are selected quickly and easily combined with other RL algorithms or other learning-based approaches [40]. The safety and efficiency of applying RL methods in real-world manufacturing scenarios still need to be improved, e.g., the idea to constrain the output through the optimal input design, which is an interesting way to improve the stability and practicality of the current RL [26]–[28].

References

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

[2] J. Mici, J. W. Ko, J. West, J. Jaquith, and H. Lipson, "Parallel electrostatic grippers for layered assembly," *Additive Manuf.*, vol. 27, pp. 451–460, May 2019.

[3] N. Gershenfeld, M. Carney, B. Jenett, S. Calisch, and S. Wilson, "Macrofabrication with digital materials: Robotic assembly," *Architectural Design*, vol. 85, no. 5, pp. 122–127, Sep. 2015.

[4] H. Chen, J. Li, W. Wan, Z. Huang, and K. Harada, "Integrating combined task and motion planning with compliant control," *Int. J. Intell. Robot. Appl.*, vol. 4, pp. 149–163, Jun. 2020.

[5] Y. Fei and X. Zhao, "Contact and jamming analysis for three dimensional dual peg-in-hole mechanism," *Mechanism Mach. Theory*, vol. 39, no. 5, pp. 477–499, May 2004.

[6] K. Van Wyk, M. Culleton, J. Falco, and K. Kelly, "Comparative peg-in-hole testing of a force-based manipulation controlled robotic hand," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 542–549, Apr. 2018.

[7] H. Park, J. Park, D.-H. Lee, J.-H. Park, M.-H. Baeg, and J.-H. Bae, "Compliance-based robotic peg-in-hole assembly strategy without force feedback," *IEEE Trans. Ind. Electron.*, vol. 64, no. 8, pp. 6299–6309, Aug. 2017.

[8] K. Zhang, J. Xu, H. Chen, J. Zhao, and K. Chen, "Jamming analysis and force control for flexible dual peg-in-hole assembly," *IEEE Trans. Ind. Electron.*, vol. 66, no. 3, pp. 1930–1939, Mar. 2019.

[9] J. C. Triyonoputro, W. Wan, and K. Harada, "Quickly inserting pegs into uncertain holes using multi-view images and deep network trained on synthetic data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 5792–5799.

[10] I. F. Jasim, P. W. Plapper, and H. Voos, "Position identification in force-guided robotic peg-in-hole assembly tasks," *Procedia CIRP*, vol. 23, pp. 217–222, Jan. 2014.

[11] T. Tang, H.-C. Lin, Y. Zhao, Y. Fan, W. Chen, and M. Tomizuka, "Teach industrial robots peg-hole-insertion by human demonstration," in *Proc. IEEE Int. Conf. Adv. Intell. Mechtron. (AIM)*, Jul. 2016, pp. 488–494.
[12] K. Sathirakul and R. H. Sturges, "Jamming conditions for multiple peg-in-hole assemblies," *Robotica*, vol. 16, no. 3, pp. 329–345, May 1998.
[13] Y. Fan, J. Luo, and M. Tomizuka, "A learning framework for high precision industrial assembly," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 811–817.
[14] T. Johannink *et al.*, "Residual reinforcement learning for robot control," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6023–6029.
[15] J. Luo *et al.*, "Reinforcement learning on variable impedance controller for high-precision robotic assembly," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3080–3087.
[16] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.
[17] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from CAD," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–9.
[18] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 819–825.
[19] F. Li, Q. Jiang, S. Zhang, M. Wei, and R. Song, "Robot skill acquisition in assembly process using deep reinforcement learning," *Neurocomputing*, vol. 345, pp. 92–102, Jun. 2019.
[20] V. Gullapalli, R. A. Grupen, and A. G. Barto, "Learning reactive admittance control," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1992, pp. 1475–1480.
[21] J. Xu, Z. Hou, W. Wang, B. Xu, K. Zhang, and K. Chen, "Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 3, pp. 1658–1667, Mar. 2019.
[22] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[23] D. Xing, F. Liu, S. Liu, and D. Xu, "Efficient insertion of partially flexible objects in precision assembly," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 706–715, Apr. 2019.
[24] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proc. Mach. Learn.* Amsterdam, The Netherlands: Elsevier, 1990, pp. 216–224.
[25] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 156–163.
[26] V. Stojanovic and V. Filipovic, "Adaptive input design for identification of output error model with constrained output," *Circuits, Syst., Signal Process.*, vol. 33, no. 1, pp. 97–113, Jan. 2014.
[27] V. Stojanovic and N. Nedic, "Robust identification of OE model with constrained output using optimal input design," *J. Franklin Inst.*, vol. 353, no. 2, pp. 576–593, Jan. 2016.
[28] V. Filipovic, N. Nedic, and V. Stojanovic, "Robust identification of pneumatic servo actuators in the real situations," *Forschung im Ingenieurwesen*, vol. 75, no. 4, pp. 183–196, Dec. 2011.
[29] R. S. Sutton *et al.*, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *Proc. 10th Int. Conf. Auto. Agents Multiagent Syst.*, vol. 2, 2011, pp. 761–768.
[30] J. Modayil, A. White, and R. S. Sutton, "Multi-timescale nexting in a reinforcement learning robot," *Adapt. Behav.*, vol. 22, no. 2, pp. 146–160, Apr. 2014.
[31] L. M. Brignone and M. Howarth, "A geometrically validated approach to autonomous robotic assembly," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 2, Sep. 2002, pp. 1626–1631.
[32] A. Morim, E. S. Fortes, P. Reis, C. Cosenza, F. Doria, and A. Gonçalves, "Think fuzzy system: Developing new pricing strategy methods for consumer goods using fuzzy logic," *Int. J. Fuzzy Log. Syst.*, vol. 7, no. 1, pp. 1–15, 2017.
[33] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[34] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2016, pp. 1–14.
[35] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2829–2838.
[36] G. Schoettler *et al.*, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural reward signals," in *Proc. ICML Workshop*, 2019, pp. 1–10.
[37] D. Pršić, N. Nedić, and V. Stojanović, "A nature inspired optimal control of pneumatic-driven parallel robot platform," *Proc. Inst. Mech. Eng. C, J. Mech. Eng. Sci.*, vol. 231, no. 1, pp. 59–71, Jan. 2017.
[38] V. Stojanovic and N. Nedic, "A nature inspired parameter tuning approach to cascade control for hydraulically driven parallel robot platform," *J. Optim. Theory Appl.*, vol. 168, no. 1, pp. 332–347, Jan. 2016.
[39] V. Stojanovic, N. Nedic, D. Prsic, L. Dubonjic, and V. Djordjevic, "Application of cuckoo search algorithm to constrained control problem of a parallel robot platform," *Int. J. Adv. Manuf. Technol.*, vol. 87, nos. 9–12, pp. 2497–2507, Dec. 2016.
[40] A. A. Bhat and V. Mohan, "Goal-directed reasoning and cooperation in robots in shared workspaces: An internal simulation based neural framework," *Cognit. Comput.*, vol. 10, no. 4, pp. 558–576, Aug. 2018.

Zhimin Hou (Graduate Student Member, IEEE) received the B.E. degree in mechanical engineering from Tongji University, Shanghai, China, in 2016, and the master's degree in mechanical engineering from Tsinghua University, Beijing, China, in 2019.

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

Zhihu Li received the B.E. degree in mechanical engineering from Beihang University, Beijing, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Mechanical Engineering, Tsinghua University, Beijing.

His research interests include metal additive manufacturing, reinforcement learning, and machine learning.

Chenwei Hsu received the B.E. degree in mechanical engineering from Chung Yuan Christian University, Taoyuan, China, in 2018. He is currently pursuing the master's degree with the Department of Mechanical Engineering, Tsinghua University, Beijing, China.

His research interests include 3-D measurement and intelligent robotics.

Kuangen Zhang received the B.E. degree from Tsinghua University, Beijing, China, in 2016. He is currently pursuing the joint Ph.D. degree with The University of British Columbia (UBC), Vancouver, BC, Canada, and the Southern University of Science and Technology (SUSTech), Shenzhen, China.

His research interests include machine learning, computer vision, sensor fusion, and predictive control.

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

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