A Learning Framework for High Precision Industrial Assembly

Yongxiang Fan[1], Jieliang Luo[2], Masayoshi Tomizuka[1]

Abstract— **Automatic assembly has broad applications in industries. Traditional assembly tasks utilize predefined trajectories or tuned force control parameters, which make the automatic assembly time-consuming, difficult to generalize, and not robust to uncertainties. In this paper, we propose a learning framework for high precision industrial assembly. The framework combines both the supervised learning and the reinforcement learning. The supervised learning utilizes trajectory optimization to provide the initial guidance to the policy, while the reinforcement learning utilizes actor-critic algorithm to establish the evaluation system even the supervisor is not accurate. The proposed learning framework is more efficient compared with the reinforcement learning and achieves better stability performance than the supervised learning. The effectiveness of the method is verified by both the simulation and experiment. Experimental videos are available at [1].**

I. Introduction

Automatic precision assembly is important for industrial manipulators to improve the efficiency and reduce the cost. Most of the current assembly tasks rely on dedicated manual tuning to provide trajectories for specific tasks, which requires intensive labors and is not robust to uncertainties. To reduce the human involvement and increase the robustness to uncertainties, more researches are focusing on learning the assembly skills.

There are three types of learning in Psychology [2]: classical conditioning, observational learning and operant conditioning. The second and third types correspond to supervised learning and reinforcement learning, respectively. The supervised learning is ideal when the training data is sufficient. Practically, collecting data is inefficient under various uncertainties of the environment. A Gaussian mixture model (GMM) is trained in [3] from human demonstration to learn a peg hole insertion skill. The peg hole insertion task is simplified by constraining the policy into planar motion and the trained policy is not adaptable to different environments.

The reinforcement learning (RL) learns a sequence of optimal actions by exploring the environment to maximize the expected reward. Different types of RL methods include the direct policy gradient such as REINFORCE [4], Q-learning based methods such as DQN [5], as well as the actor-critic framework such as DDPG [6] or PPO [7]. These methods are called model-free RL since the dynamics model is not used during exploration. Despite lack of dynamics, the model-free RL has been successfully applied to assembly tasks [8], [9]. The model-free RL requires considerable data to explore the state/action space and reconstruct the transitions of the environment. Consequently, it is less data-efficient and time-efficient.

Model-based RL is proposed to increase the data efficiency [10], [11]. It fits dynamics models and applies optimal control such as iLQR/iLQG [12] to compute the optimal trajectories. The exploration is conducted by adding random noise to the actions during the optimization. Then the optimized trajectories are used to train a neural network policy in a supervised manner. Compared with model-free RL, the model-based RL has larger exploit-exploration ratio, thus explores narrower space and converges faster than the model-free RL. The performance of the model-based RL depends on the behavior of the optimal controller (i.e. supervisor), which in turn is effected by the accuracy of the local dynamics model. For the rigid robot dynamics with force/torque as states, the dynamics model is less smooth[1], which makes the dynamics fitting not effective. Consequently, the model-based RL cannot converge consistently. In practice, people usually use soft robotics model (Baxter, PR2) [11] with position/velocity states by ignoring the force/torque feedback.

This paper proposes a learning framework to train a more natural assembly policy by incorporating both the force/torque and the positional feedback signals. The proposed framework combines the model-based RL with the model-free actor-critic to learn the manipulation skills for precision assembly tasks. The model-based RL computes for the optimal trajectories with both positional and force/torque feedback. The performance of the controller might be affected by the smoothness of the local fitted dynamics model. To avoid the problem of inconsistency or tedious parameter tuning of optimal controller, a critic network is introduced to learn the correct critic value (Q-value). Instead of training the policy network by pure supervision, we train an actor network by combining the supervised learning with the policy gradient. To accelerate the training efficiency of the critic network, the Q-value from the optimal control is employed to train the critic network.

The contribution of this work are as follows. First, the optimal controller is able to constrain the exploration space in safe region compared with the random exploration at the first iterations of actor-critic methods. Secondly, the optimal controller is more data-efficient when exploring in a narrower space and solving for optimal trajectory mathematically. Thirdly, the combined critic network is able to address the potential inconsistency and instability of the optimal controller caused by the rigid robotics system and force/torque

Yongxiang Fan and Masayoshi Tomizuka are with University of California, Berkeley, Berkeley, CA 94720, USA yongxiang_fan, tomizuka@berkeley.edu

Jieliang Luo is with University of California, Santa Barbara, Santa Barbara, CA 93106, USA jieliang@ucsb.edu

[1]The dynamics change dramatically as the trajectory slightly changes.

feedback, and build up a ground truth critic for the policy network.

The remainder of this paper is described as follows. The related work is stated in Section II, followed by a detailed explanation of the proposed learning framework in Section III. Simulation and experiment results are presented in Section IV. Section V concludes the paper and proposes future works.

II. Related Work

The objective of an assembly task is to learn an optimal policy $\pi_\theta(a_t|o_t)$ to choose an action a_t based on the current observation o_t in order to maximize an expected reward:

$$\min_{\pi_\theta} E_{\tau\sim\pi_\theta}(l(\tau)), \tag{1}$$

where θ is the parameterization of the policy, $\tau = \{s_0, a_0, s_1, a_1, ..., s_T, a_T\}$ is the trajectory, $\pi_\theta(\tau) = p(s_0)\prod_1^T p(s_t|s_{t-1}, a_{t-1})\pi_\theta(a_t|s_t)$, and l is the loss of the trajectory τ.

Equation (1) can be solved by optimization once a global dynamics model $p(x_t|x_{t-1}, u_{t-1})$ is explicitly modeled. For a contact-rich complex manipulation task, the global dynamics model is extremely difficult to obtain. Therefore, the assembly task either avoids using dynamics [9] or fits the a linear dynamics model [3], [10], [11].

On one hand, the RL without dynamics requires excessively data to explore the space and locate to the optimal policy due to the potential high-dimensionality of the action space. On the other hand, the performance of the [10], [11] can be downgraded once the robotic system is rigid or the force/torque feedback is included in the optimal controller.

We propose a learning framework that combines the actor-critic framework and optimal control for efficient high-accuracy assembly. The optimal controller is adapted from the model-based RL [10], while the actor-critic framework is modified from the DDPG algorithm. These two algorithms will be briefly introduced below.

A. Deep Deterministic Policy Gradient (DDPG)

The DDPG algorithm collects sample data (s_j, a_j, s_{j+1}, r_j) from the replay buffer R and trains a critic network Q_ϕ and actor network u_θ parameterized by ϕ and θ. More specifically, the critic network is updated by:

$$\begin{aligned}\phi &\leftarrow \operatorname*{argmin}_\phi \frac{1}{N_{dd}}\sum_{j=1}^{N_{dd}}\left(y_j - Q_\phi(s_j, a_j)\right)^2, \\ y_j &= r_j + \gamma Q_{\hat{\phi}}(s_{j+1}, u_{\hat{\theta}}(s_{j+1})),\end{aligned} \tag{2}$$

where N_{dd} is the batch size for DDPG, $\hat{\phi}, \hat{\theta}$ are parameters of the target critic network and target actor network, and γ is the discount for future reward.

The policy network is updated by:

$$\theta \leftarrow \operatorname*{argmax}_\theta \frac{1}{N_{dd}}\sum_{j=1}^{N_{dd}} Q_{\hat{\phi}}(s_j, u_\theta(s_j)), \tag{3}$$

where θ is the parameters for the policy network to be optimized. Policy gradient is applied to update the parameters of the actor network:

$$\theta \leftarrow \theta + \alpha\frac{1}{N_{dd}}\sum_{j=1}^{N_{dd}}\nabla_a\hat{Q}(s,a)|_{s=s_j,a=a_j}\nabla_\theta u_\theta(s)|_{s=s_j}, \tag{4}$$

where the α is the learning rate of the actor network.

The target networks are updated by

$$\begin{aligned}\hat{\phi} &\leftarrow \delta\phi + (1-\delta)\hat{\phi}, \\ \hat{\theta} &\leftarrow \delta\theta + (1-\delta)\hat{\theta},\end{aligned} \tag{5}$$

where δ is the target update rate and is set to be small value ($\delta \approx 0.01$).

B. Guided Policy Search (GPS)

With the involvement of guiding distribution $p(\tau)$,Problem (1) can be rewritten as

$$\min_{\pi_\theta, p} E_p(l(\tau)), \quad s.t.\ \ p(\tau) = \pi_\theta(\tau). \tag{6}$$

GPS solves (6) by alternatively minimizing the augmented Lagrangian with respect to primal variables p, π_θ and updating the Lagrangian multipliers λ. The augmented Lagrangian for θ and p optimization are:

$$\begin{aligned}L_p(p,\theta) =& E_p(l(\tau)) + \lambda\left(\pi_\theta(\tau) - p(\tau)\right) + \\ & \nu D_{KL}\left(p(\tau)\|\pi_\theta(\tau)\right), \\ L_\theta(p,\theta) =& E_p(l(\tau)) + \lambda\left(\pi_\theta(\tau) - p(\tau)\right) + \\ & \nu D_{KL}\left(\pi_\theta(\tau)\|p(\tau)\right),\end{aligned} \tag{7}$$

where λ is the Lagrangian multiplier, ν is the penalty parameter for the violation of the equality constraint, and D_{KL} represents the KL-divergence. The optimization of primal variable p is called trajectory optimization. It optimizes the guiding distribution p with learned local dynamics. To assure the accuracy of dynamics fitting, the optimization is constrained within the trust region ϵ:

$$\min_p L_p(p,\theta), \quad s.t.\ \ D_{KL}(p(\tau)\|\hat{p}(\tau)) \le \epsilon, \tag{8}$$

where $\hat{p}$ is the guiding distribution of the previous iteration. The Lagrangian of (8) is:

$$\mathcal{L}(p) = L_p(p,\theta) + \eta(D_{KL}(p(\tau)\|\hat{p}(\tau)) - \epsilon), \tag{9}$$

where η is the Lagrangian multiplier for the constraint optimization. With the Gaussian assumption of the dynamics, (9) is solved by iLQG. To avoid large derivation from the fitted dynamics, η is adapted by comparing the predicted KL-divergence with the actual one.

The optimization of the policy parameters θ can be written as a supervised learning problem. With the Gaussian policy $\pi_\theta(a_t|o_t) = \mathcal{N}(u_\theta(o_t), \Sigma_t^\pi)$, we can rewrite $L_\theta(p,\theta)$ in (7) as:

$$\begin{aligned}L_\theta(\theta,p) = \frac{1}{2N_b}\sum_{i,t=1}^{N_b,T} E_{p_i(s_t,o_t)}[\mathrm{tr}\left(C_{ti}^{-1}\Sigma_t^\pi\right) - \log|\Sigma_t^\pi| + \\ \left(u_\theta(o_t) - u_{ti}^p(s_t)\right)^T C_{ti}^{-1}\left(u_\theta(o_t) - u_{ti}^p(s_t)\right) + 2\lambda_t^T u_\theta(o_t)],\end{aligned} \tag{10}$$

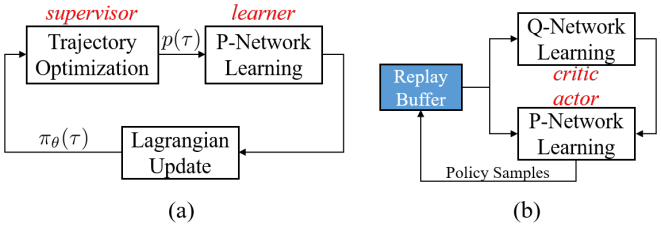


Fig. 1: (a) Guided Policy Search (GPS) and (b) Deep deterministic policy gradient (DDPG).

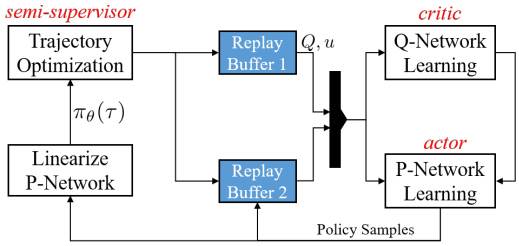


Fig. 2: Illustration of the proposed learning framework (guided-DDPG). Trajectory optimization provides initial guidance to both actor and critic nets to avoid excessive exploration. The actor-critic nets gradually establish the evaluation system, instead of relying on pure supervised learning.

where $p_i(u_t|s_t) \sim \mathcal{N}(u^p_{ti}(s_t), C_{ti})$ is the guiding distribution. Equation (10) contains the decoupled form of the variance optimization and policy optimization. Refer [11] for more details.

C. Comparison of GPS and DDPG

GPS decouples RL into a trajectory optimization (*supervisor*) and a supervised policy network learning (*learner*), as shown in Fig. 1(a). The performance of the learner relies on the quality of the supervisor. By fitting the dynamics from sampling data and computing the supervisor with the optimal control, GPS is more efficient than the DDPG and many other model-free RL algorithms. However, the performance of the learner would be compromised if the system has high stiffness and has force/torque feedback as states due to the less smooth dynamics and smaller trust region.

In comparison, DDPG uses rollout samples to jointly train the Q-network (*critic*) and policy network (*actor*), as shown in Fig. 1(b). The critic gradually builds up the Q-value from physical rollouts, and the Q-value is applied to train the actor network based on policy gradient. The actor-critic framework provides more stable policy in the tasks with non-smooth dynamics. These tasks are common in high precision industrial assembly where the system has higher stiffness and contains force/torque feedback in the states. Despite the reliable performance, the actor-critic framework is less data efficient due to the intensive exploration, which is usually unnecessary since assembly tasks only requires exploration in narrow trajectory space.

III. Proposed Approach

Precision industrial assembly usually has large system stiffness in order to achieve precise tracking performance and reduce the vibration. With large stiffness, small clearance and force/torque feedback, both the model-free RL and model-based method cannot accomplish the task efficiently and stably. In this paper, we propose a learning framework that combines the actor-critic with the model-based RL for high precision industrial assembly. The framework is named as guided-deep deterministic policy gradient (guided-DDPG). Guided-DDPG behaves more efficient than the actor-critic and more stable/reliable than the model-based RL.

Figure 2 illustrates the proposed guided-DDPG algorithm. Due to the discontinuity of the fitted dynamics in rigid precise systems, the trajectory optimization can have inconsistent behavior or requires dedicated parameter tuning. Therefore, a pure supervised learning from trajectory optimization cannot fulfill the task consistently. The actor-critic is incorporated to the framework to address this issue. The trajectory optimization serves as a *semi-supervisor* to train the actor-critic to establish the initial critic and constrain the network in narrow task space. The involvement of the supervision will be reduced as the training progresses and the critic network becomes more accurate, since the actor-critic exhibits superior performance than the semi-supervisor.

To be more specific, the trajectory optimization (semi-supervisor) has the following form:

$$\min_p E_p(l(\tau)), \quad s.t. \;\; D_{KL}(p(\tau)\|\hat{p}_\theta(\tau)) \le \epsilon, \tag{11}$$

where $\hat{p}_\theta$ is set as the trajectory distribution generated by actor policy at the first sub-iteration, and is set as the previous trajectory distribution $\hat{p}$ for the successive $N_{trajopt} - 1$ sub-iterations. Equation (11) is optimized by the dual:

$$\max_\eta \{\min_p E_p(l(\tau)) + \eta(D_{KL}(p(\tau)\|\hat{p}_\theta(\tau)) - \epsilon)\}. \tag{12}$$

The optimization of p is solved by LQG with fixed η and dynamics, and the optimization of η is done heuristically: decrease η if $D_{KL}(p(\tau)\|\hat{p}_\theta(\tau)) < \epsilon$, otherwise increase η. The trust region ϵ varies based on the expected improvement and actual one. ϵ would be reduced once the actual improvement is far smaller from the expected one, thus the network focuses on penalizing the KL divergence from $\hat{p}_\theta(\tau)$.

We collect the trajectory after $N_{trajopt}$ sub-iterations to replay buffer R_1 for supervised training of actor-critic nets, and feed all the sample data during $N_{trajopt}$ executions to replay buffer R_2. With the supervision from R_1, the critic is trained by:

$$\begin{aligned}\phi \leftarrow \operatorname*{argmin}_\phi \frac{1}{N_{dd}} \sum_{j=1}^{N_{dd}} (y_j - Q_\phi(s_j, a_j))^2 + \\ w_{to} \frac{1}{N_{to}} \sum_{i=1}^{N_{to}} \|Q_\phi(s_i, a_i) - Q_i^{to}\|^2\end{aligned} \tag{13}$$

where w_{to}, N_{to} are the weight and batch size of the semi-supervisor, y_j has the same form as (2). (s_i, a_i, Q_i^{to}) is the supervision data from R_1, and (s_j, a_j, r_j, s_{j+1}) is the sample data from R_2.

Algorithm 1 Guided-DDPG

1: **input:** $EP, N_{ddpg}, N_{inc}, N_{trajopt}, N_{roll} = 0, R_{1/2} \leftarrow \Phi$
2: **init:** $Q_\phi(s,a), u_\theta(s)$, set target nets $\hat{\phi} \leftarrow \phi, \hat{\theta} \leftarrow \theta$
3: **for** $epoch = 0 : EP$ **do**
4: $\quad p_{prev} \leftarrow u_\theta$
5: $\quad$ **for** $it = 0 : N_{trajopt}$ **do**
6: $\quad\quad \mathcal{S} \leftarrow sample\_data(p_{prev})$, $R_2 \leftarrow R_2 \cup \mathcal{S}$
7: $\quad\quad f_{dy} \leftarrow fit\_dynamics(\mathcal{S})$
8: $\quad\quad \hat{p}_\theta \leftarrow linearize\_policy(p_{prev}, \mathcal{S})$
9: $\quad\quad p \leftarrow update\_trajectory(f_{dy}, \hat{p}_\theta), p_{prev} \leftarrow p$
10: $\quad$ **end for**
11: $\quad S \leftarrow sample\_data(p)$, $R_1 \leftarrow R_1 \cup \mathcal{S}, R_2 \leftarrow R_2 \cup \mathcal{S}$
12: $\quad$ **for** $it = 0 : N_{ddpg}$ **do**
13: $\quad\quad \mathcal{N}_{ex} \leftarrow exploration\_noise()$
14: $\quad\quad s_0 \leftarrow observe\_state(), w_{to} = \frac{c}{c+N_{roll}++}$
15: $\quad\quad$ **for** $t = 0 : T$ **do**
16: $\quad\quad\quad a_t = u_\theta(s_t) + \mathcal{N}_{ex}(t)$, observe s_{t+1}, r_t
17: $\quad\quad\quad R_2 \leftarrow R_2 \cup (s_t, a_t, s_{t+1}, r_t)$
18: $\quad\quad\quad$ sample N_{to}, N_{dd} transitions from R_1, R_2
19: $\quad\quad\quad$ update critic and actor nets by (13) and (14)
20: $\quad\quad\quad$ update target nets by (5)
21: $\quad\quad$ **end for**
22: $\quad$ **end for**
23: $\quad N_{ddpg} \leftarrow N_{ddpg} + N_{inc}$
24: **end for**

The actor is trained by:

$$\theta \leftarrow \underset{\theta}{\operatorname{argmax}} \frac{1}{N_{dd}} \sum_{j=1}^{N_{dd}} Q_{\hat{\phi}}(s_j, u_\theta(s_j)) + w_{to} \frac{1}{N_{to}} \sum_{i=1}^{N_{to}} \|u_\theta(s_i) - a_i\|^2 \tag{14}$$

The supervision weight w_{to} decays as the number of training rollouts N_{roll} increases. We use $w_{to} = \frac{c}{N_{roll}+c}$, where c is a constant to control the decay speed.

The guided-DDPG algorithm is summarized in Alg. 1. The critic and actor are initialized in Line 2. Guided-DDPG runs for EP epochs in total. In each epoch, semi-supervisor is first executed to update the trajectories for supervision. With the high stiffness, small clearance and the force/torque feedback, the fitted dynamics (Line 7) is discontinuous and has small trust region. Therefore, the trajectories generated from semi-supervisor might be sub-optimal. Nevertheless, they are sufficient to guide the initial training of the actor-critic. The actor-critic is trained in Line (12 - 22) following the standard procedure of DDPG with the modified objective function (Line (19)). The supervision weight w_{to} is decreased as the training progresses due to the superior performance of the actor-critic than the semi-supervisor.

IV. Simulations and Experiments

This section presents both the simulation and experimental results of the guided-DDPG to verify the effectiveness of the proposed learning framework. The videos are available at [1].

To compare the performance of the guided-DDPG with other state-of-the-art RL algorithms, we built up a simulation model using the Mujoco physics engine [13]. The host computer we used was a desktop with 32GB RAM, 4.0GHz CPU and GTX 1070 GPU. A 6-axis UR5 robot model from universal robotics was used to perform the tasks. Two different assembly tasks were simulated, the first one was the Lego brick insertion, and the second one was the U-shape joint assembly, as shown in Fig. 3.

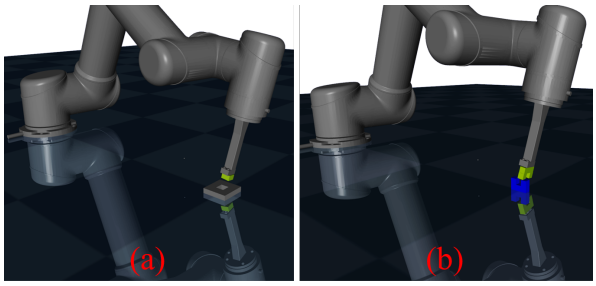


Fig. 3: Two simulation tasks for algorithm evaluation. (a) Lego brick insertion, (b) U-shape joint assembly.

A. Parameter Lists

The number of the maximum epoch is set to $EP = 100$, initial number of rollouts for DDPG and trajectory optimization were $N_{ddpg} = 21$ and $N_{trajopt} = 3$, respectively. To ensure less visit of trajectory optimization as the training progresses, we increased the number of rollouts by $N_{inc} = 15$ for each DDPG iteration. The sizes of the replay buffer R_1, R_2 were 2000 and 1E6, respectively. The soft update rate $\gamma = 0.001$ in (5). The batch size for trajectory optimization N_{to} and DDPG N_{dd} were both 64. The algorithm used a cost function $l(s,a) = 0.0001\|a\|_2 + \|FK(s) - p_{tgt}(s)\|_2$, where FK represents the forward kinematics and p_{tgt} is the target end-effector points.

B. Simulation Results

The simulation results on U-shape joint assembly and Lego brick insertion are shown by Fig. 4. Both simulations were trained with assembly clearance as 0.1 mm. Guided-DDPG takes poses and force/torque measurements of the end-effector as the states, and generates joint torques as action to drive the robot. The U-shape joint has more complicated surface than the Lego brick, and a successful assembly requires matching the shapes twice, as shown in Fig. 4 (Top). Despite the difficulties, the proposed algorithm was able to train the policy within 1000 rollouts. We also visualized the adaptability of the trained policy on the Lego brick insertion task, as shown in Fig. 4 (Bottom). The policy was trained with a brick of size 2×2 and clearance 0.1 mm and tested with a brick of size 4×2 and clearance 1 µm. Moreover, the brick position had an unknown offset (1.4 mm) to the network. The proposed network was able to address these uncertainties and successfully inserted the brick to a tighter hole with uncertain position.

1) Comparison of Different Supervision Methods: The proposed learning framework guides both the critic and actor. To illustrate the necessity of the proposed guidance, we

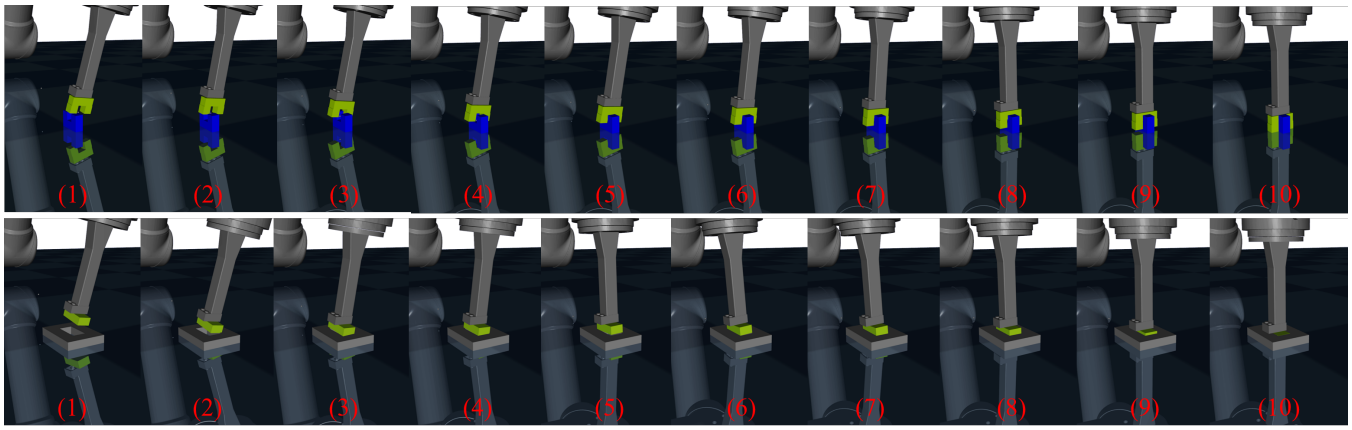


Fig. 4: Simulation animations of the proposed guided-DDPG on (Top) U-shape joint assembly and (Bottom) Lego brick insertion. The guided-DDPG was trained on 2×2 Lego and tested on 4×2 one. Snapshots are taken from left to right.

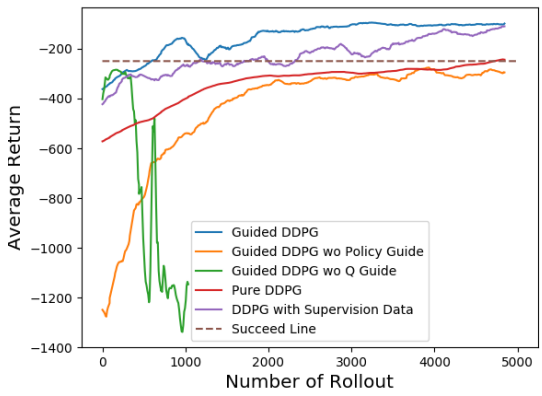


Fig. 5: Comparison of different supervisions with Lego brick insertion task. The supervision methods with performance in descending order: guided-DDPG (proposed), DDPG with supervised data in Replay buffer, pure DDPG, guided-DDPG w/o policy guidance, and guided-DDPG w/o critic guidance.

compared the results of guided-DDPG with several other supervision methods, including the guided-DDPG with partial guidance, pure-DDPG with supervision data to replay buffer (no supervision on objective function) and the pure-DDPG. The result was shown in Fig. 5. The proposed guided-DDPG achieved the best performance. The partial guidance without critic (Fig. 5 Green) was able to guide the actor and realized safe exploration at the beginning. However, the actor network behaved worse as the involvement of the semi-supervisor reduced and the weight of the critic increased, since the critic is trained purely by the contaminated target actor (2). In contrast, the partial guidance without actor (Fig. 5 Orange) had poorly behaved actor since the actor was trained purely by the policy gradient using the contaminated critic (3). The pure-DDPG with supervision data (Fig. 5 Purple) achieved better performance than pure-DDPG, since the trajectories obtained from semi-supervisor were better behaved than the initial rollouts of DDPG. This kind of supervision is similar with the human demonstration in [8].

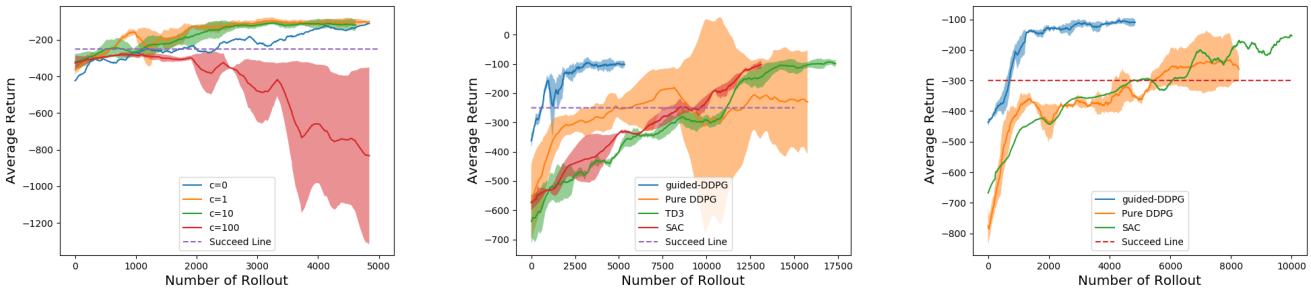
TABLE I: Comparison between DDPG and guided-DDPG

items	DDPG	Guided-DDPG
time (min)	83	**37.3**
data (rollouts)	7000	**1500**

2) Effects of the Supervision Weight w_{to}: The supervision weight w_{to} balances the model-based supervision and model-free policy gradient in actor/critic updates, as shown in (14) and (13). The results of different weights on Lego brick insertion are shown in Fig. 6 (a). With $c = 1$, the supervision weight is $w_{to} = \frac{1}{1+N_{roll}}$. The weights starts with 1 and decays to 0.001 as $N_{roll} = 1000$, while $c = 100$ makes w_{to} decay to 0.1 as $N_{roll} = 1000$. Slower decay provides excessive guidance by the semi-supervisor and contaminates the original policy gradient and makes the DDPG unstable. Empirically, $c = 1 \sim 10$ achieves comparable results.

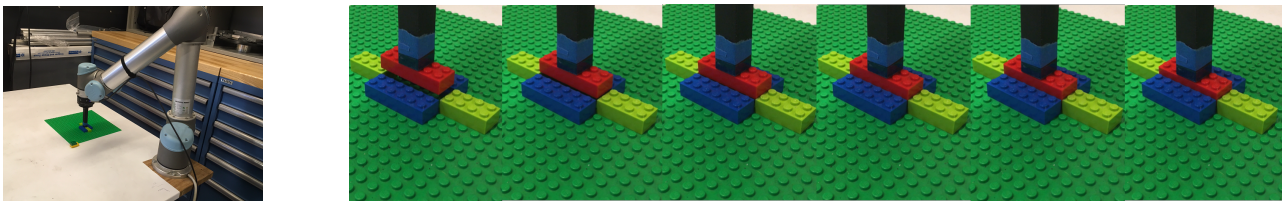
3) Comparison of Different Algorithms: The proposed learning framework was compared with other state-of-the-art algorithms, including the pure-DDPG, twin delayed deep deterministic policy gradients (TD3) [14] and the soft actor-critic (SAC) [15]. Default parameters were used for TD3, as shown in [16]. As for SAC, we used the default parameters in [16] with tuned reward scale as 10. The comparison result on Lego brick insertion task is shown in Fig. 6 (b). The proposed guided-DDPG passed the success threshold (shaded purple line) at the 800 rollouts and consistently succeeded the task after 2000 rollouts. In comparison, the pure DDPG passed the success threshold at the 5000 rollouts and collapsed around 10000 rollouts. The performance of pure DDPG was inconsistent in seven different trials. TD3 and SAC had the similar efficiency with pure DDPG. The comparison of the algorithms on U-shape joint assembly is shown in Fig. 6 (c). Similar with Lego brick insertion, the guided-DDPG achieved more stable and efficient learning. The time efficiency and data-efficiency of the DDPG and guided-DDPG are compared in Table I.

4) Adaptability of the Learned Policy: The adaptability of the learned policy is discussed in this section. Three different types of uncertainties were considered. The first type was the unknown hole position. The learned policy was able to



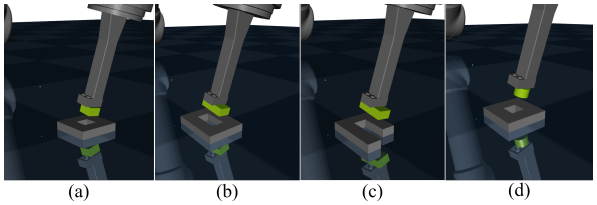
(a) **(b)** **(c)**

Fig. 6: (a) Illustration of the supervision weights on Lego brick insertion task. (b) Comparison of the algorithms for Lego brick insertion task. (c) Comparison of the algorithms for U-shape joint assembly task.



(a) **(b)**

Fig. 7: (a) Experimental setup, and (b) experimental results for Lego brick insertion.



(a) (b) (c) (d)

Fig. 8: Different shapes of the bricks and holes for adaptability test. (a) 2×2 brick used in training, (b) 4×2 brick, (c) 4×2 brick with incomplete hole, and (d) cylinder brick.

successfully insert the brick when moving the hole to an uncalibrated position (maximum offset is 5 mm, hole has width of 16 mm). The second type of uncertainty was the shapes of peg/hole. We found that the learned policy is robust to different shapes shown in Fig. 8. The third type was the different clearance. The policy was trained with clearance 0.1 mm and tested successfully on insertion tasks with clearance 10 μm, 1 μ and 0. The simulation videos are available at [1].

C. Experimental Results

Experimental results are presented in this section. The Lego brick was attached to a 3D printed stick at the end-effector of the Universal robot (UR5). A Robotiq FT 300 force torque sensor was used to collect the force/torque signal at the wrist. The experimental setup is shown in Fig. 7(a). The policy took the estimated hole position and the force/torque reading as inputs, and generated transitional velocities for the end-effector. The velocity was tracked by a low-level tracking controller. The clearance of the Lego brick is less than 0.2 mm. The target position of the hole had 0.5 mm uncertainty, yet the policy was able to successfully locate the hole and insert the brick, as shown in Fig. 7(b). It took 2 hours for pure-DDPG to find a policy in the exploration space bounded within 1 mm around the hole, and took 1.5 hours for guided-DDPG to find a policy in a larger exploration space bounded within 3 mm around the hole. The experimental videos are shown in [1].

V. Conclusions and Future Works

This paper proposed a learning framework for high precision assembly task. The framework contains a trajectory optimization and an actor-critic structure. The trajectory optimization was served as a semi-supervisor to provide initial guidance to actor-critic, and the critic network established the ground-truth quality of the policy by learning from both the semi-supervisor and exploring with policy gradient. The actor network learned from both the supervision of the semi-supervisor and the policy gradient of the critic. The involvement of critic network successfully addressed the stability issue of the trajectory optimization caused by the high-stiffness and the force/torque feedback. The proposed learning framework constrained the exploration in a safe narrow space, improved the consistency and reliability of the model-based RL, and reduced the data requirements to train a policy. Simulation and experimental results verified the effectiveness of the proposed learning framework.

In the future, the authors would evaluate the algorithm on more realistic industrial applications such as connector insertion, furniture assembly and tight peg-in-hole tasks.

Acknowledgment

The authors would like to thank Dr. Yotto Koga and AI Lab in Autodesk Inc. for the help on experiments.

REFERENCES

[1] Experimental Videos for A Learning Framework for High Precision Assembly Task, http://me.berkeley.edu/%7Eyongxiangfan/ICRA2019/guidedddpg.html.

[2] J. W. Kalat, *Introduction to psychology*. Nelson Education, 2016.

[3] T. Tang, H.-C. Lin, Y. Zhao, Y. Fan, W. Chen, and M. Tomizuka, "Teach industrial robots peg-hole-insertion by human demonstration," in *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on*. IEEE, 2016, pp. 488–494.

[4] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[8] M. Vecerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *CoRR, abs/1707.08817*, 2017.

[9] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 819–825.

[10] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[12] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4906–4913.

[13] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.

[14] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[16] V. Pong, "rlkit: reinforcement learning framework and algorithms implemented in pytorch." https://github.com/vitchyr/rlkit.git, 2018.