A Learning Approach to Robot-Agnostic Force-Guided High Precision Assembly

Jieliang Luo* and Hui Li*

Abstract— In this work we propose a learning approach to high-precision robotic assembly problems. We focus on the contact-rich phase, where the assembly pieces are in close contact with each other. Unlike many learning-based approaches that heavily rely on vision or spatial tracking, our approach takes force/torque in task space as the only observation. Our training environment is *robotless*, as the end-effector is not attached to any specific robot. Trained policies can then be applied to different robotic arms without re-training. This approach can greatly reduce complexity to perform contact-rich robotic assembly in the real world, especially in unstructured settings such as in architectural construction. To achieve it, we have developed a new distributed RL agent, named Recurrent Distributed DDPG (RD2), which extends Ape-X DDPG[1] with recurrency and makes two structural improvements on prioritized experience replay[2]. Our results show that RD2 is able to solve two fundamental high-precision assembly tasks, lap-joint and peg-in-hole, and outperforms two state-of-the-art algorithms, Ape-X DDPG and PPO with LSTM. We have successfully evaluated our robot-agnostic policies on three robotic arms, Kuka KR60, Franka Panda, and UR10, in simulation. The video presenting our experiments is available at `https://sites.google.com/view/rd2-rl`

I. INTRODUCTION

Robotic systems for automated assembly have been widely used in manufacturing, where the environment can be carefully and precisely controlled, but they are still in infancy in architectural construction. A main reason is that current robotic systems are not adaptive to the diversity of the real world, especially in unstructured settings. RL-based robotic systems [3], [4], [5] are a promising direction given their adaptability to uncertainties. Current successful RL examples heavily rely on vision or spatial tracking to perform complex control tasks [6], [7], [8], [9]. However, it is unrealistic to expect motion capture or other tracking systems at a construction site, as they are hard to install, calibrate, and scale. A vision system is more portable, but in the contact-rich phase of assembly, it often fails to help due to occlusion or poor lighting conditions. Another limitation of the current RL-based robotic systems is that the policies are robot-specific and cannot readily generalize to other robotic platforms. This limits the efficiency and scalability of RL-based robotic systems, especially in construction, where often a collection of different robotic platforms are used.

In this paper, we present a learning approach to solving high-precision robotic assembly tasks of varying complexity. We are interested in the contact-rich phase of assembly,

because when the assembly pieces are in close contact with each other, force/torque (F/T) measurements become the most revealing observation. The research question we want to answer is: in this contact-rich phase, is F/T alone sufficient for learning a robot control policy? To do this, we develop a recurrent distributed deep RL agent called Recurrent Distributed DDPG (RD2) that extends Ape-X DDPG and solves the force-guided robotic assembly problems in the continuous action space. Specifically, we add recurrency in the neural networks to learn a memory-based representation in order to compensate for partial observability of the process, i.e. the lack of pose observations. Additionally, we create a dynamic scheme to pre-process episode transitions before sending them to the reply buffer by allowing overlap of the last two sequences in each episode to be variable, which maintains important information of the final transitions and avoids crossing episode boundaries. To overcome the training instability of the DDPG family [10], [1], [11], [12], we calculate priorities for the sequences in the replay buffer as well as priorities of the transitions in each sequence. The former priorities are used for sequence sampling and the latter are for bias annealing. We use *robotless* environments for training. During the contact-rich phase of assembly, our tasks can be easily confined within the robot workspace that is collision free and singularity free, and robot motion can be considered quasi-static. This ensures the policies can successfully transfer to various robotic arms without re-training.

We evaluate RD2 on two fundamental assembly tasks, lap-joint and peg-in-hole, with tight tolerance and varying complexity, and compare the performance of RD2 to Ape-X DDPG and PPO [13] with LSTM (LSTM-PPO). The results show that RD2 outperforms the other two algorithms across all the tasks and presents a stable performance as the difficulty of the tasks increases, whereas Ape-X DDPG and LSTM-PPO fail on most of the tasks. We also show that the trained robotless policies adapt well to different robotic arms with different initial states and different physical noise injected in F/T measurements and friction parameters in simulation.

The main contribution of this work is a robot-agnostic learning approach to solving contact-rich high-precision robotic assembly tasks with F/T as the only observation. Trained polices can be deployed on different robotic arms without re-training. We believe this work is an important step towards deploying robots on unstructured construction sites, where different robots can share trained policies and adapt to misalignment with minimal sensing setup.

*The authors contributed equally and are with Autodesk Research, San Francisco, United States. Emails: `rodger.luo@autodesk.com`, `hui.xylo.li@autodesk.com`
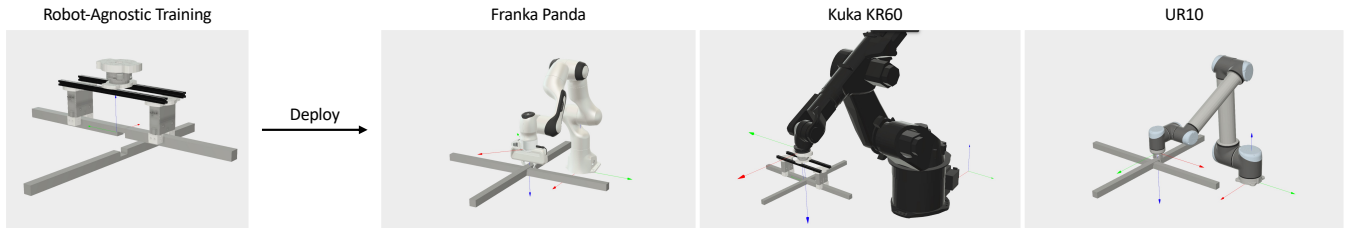
Fig. 1: We develop a learning approach to solving robotic assembly tasks in the contact-rich phase with F/T measurements as the only observation. The training environment in simulation is robot-agnostic so that the polices can be deployed on various robotic arms.

The remainder of this paper is structured as follows. Problem statement and related work are stated in Section II, followed by a detailed explanation of our proposed approach in Section III. Experimental setup, results, and evaluation are presented in Section IV. Section V concludes the paper and discusses future work.

II. PROBLEM STATEMENT AND RELATED WORK

A. Problem Statement

We model the problem we solve in this paper as a Partially Observable Markov Decision Process (POMDP), which is described by a set of states S, a set of actions A, a set of conditional probabilities $p(s_{t+1}|s_t, a_t)$ for the state transition $s_t \rightarrow s_{t+1}$, a reward function $R: S \times A \rightarrow \mathbb{R}$, a set of observations Ω, a set of conditional observation probabilities $p(o_t|s_t)$, and a discount factor $\gamma \in [0, 1]$.

In principle, the agent makes decisions based on the history of observations and actions $h_t = (o_1, a_1, o_2, a_2..., o_t, a_t)$ and the goal of the agent is to learn an optimal policy π_θ in order to maximize the expected discounted rewards:

$$\max_{\pi_\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t) \right], \qquad (1)$$

where trajectory $\tau = (s_1, o_1, a_1, s_2, o_2, a_2, ..., s_T, o_T, a_T)$, θ is the parameterization of policy π, and $\pi_\theta(\tau) = p(s_1)p(o_1|s_1)\pi_\theta(a_1|h_1)\prod_2^T p(s_t|s_{t-1}, a_{t-1})p(o_t|s_t)\pi_\theta(a_t|h_t)$.

For many POMDP problems, it is not practical to condition on the entire history of the observations [12]. In this paper, we tackle this challenge by investigating recurrent neural networks with distributed model-free RL and we focus on the continuous action domain.

B. Distributed Reinforcement Learning

Distributed reinforcement learning can greatly improve sample efficiency of model-free RL by decoupling learning and data collection.

Ape-X [1] disconnects exploration from learning by having multiple actors interacting with their own environment and sending the collected transitions into one of the distributed replay buffers. A learner asynchronously samples a batch of transitions from a randomly picked buffer. Ape-X has both DQN [14] and DDPG [11] variants to support discrete and continuous action spaces, respectively. Built upon Ape-X, D4PG [10] introduced a distributional critic update and incorporated N-step returns and prioritization of experience replay to achieve a more stable learning signal.

Unlike the actors and the learner in Ape-X that randomly feed and samples transitions from replay buffers, IM-PALA [15] asks each actor to send the collected transitions via a first-in-first-out queue to the learner and to update its policy weights from the learner before the next episode. In addition, it introduces V-trace, a general off-policy learning algorithm that corrects a *policy-lag* between the learner and the actors as the actors' policies are usually several updates behind the learner's. Both Ape-X and IMPALA have demonstrated strong performance in the Atari-57 and DMLab-30 benchmarks, but they have not been examined on partially observable robotic assembly tasks.

C. RL for Robotic Assembly

RL has been studied actively in the area of robotic assembly as it can reduce human involvement and increase robustness to uncertainty. Dynamic Experience Replay [8] uses experience replay samples not only from human demonstrations but also successful transitions generated by RL agents during training and therefore improves training efficiency. [16] proposed a framework to combine DDPG [11] and GPS [17] to take advantage of both model-free and model-based RL [18] to solve high-precision Lego insertion tasks. [19] uses self-supervision to learn a multimodal representation of visual and haptic feedback to improve sample efficiency. Both [5] and [9] focus on training a variable compliance controller to solve peg-in-hole tasks. [5] introduces incremental displacement and force in the observation space and visual Cartesian stiffness in the action space to improve the efficiency of the trained policies. [9] applies domain transfer-learning techniques to improve training efficiency and domain randomization to increase the adaptability of the learned policies. [20] solves the problem of timber joint assembly in architectural construction. These methods all require pose information, directly or indirectly, as observations. In our work, we focus on the contact-rich phase and only use measurements from the wrist-mounted F/T sensor as observations. This allows the RL system to solve assembly tasks where vision or pose tracking systems are unavailable, e.g. inside a small confined space.

D. RL under Partial Observability

Partial observability is a well known challenge in robotics, resulting from occlusions, unpredictable dynamics, or noisy
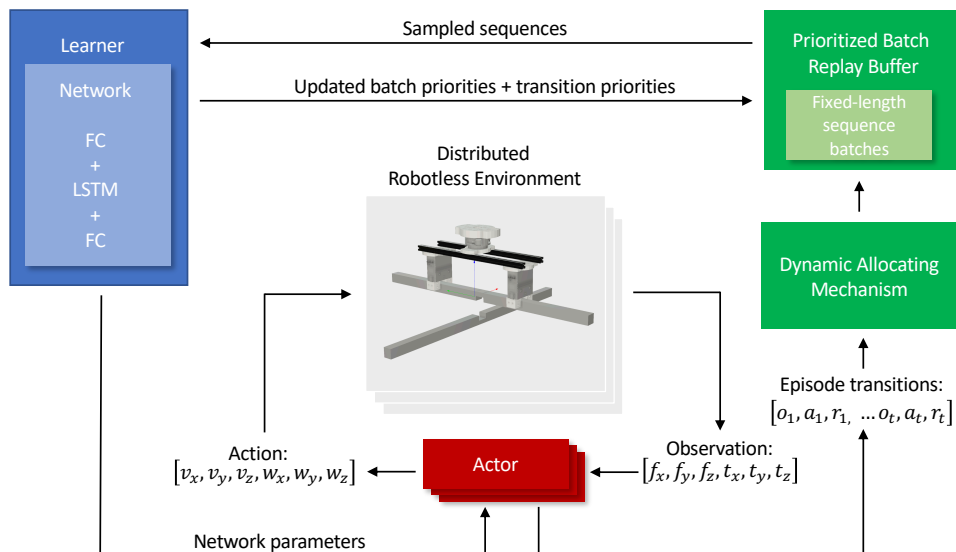
Fig. 2: The architecture of RD2: it has multiple actors, each with its own instance of environment, to collect episodes of (o, a, r). Since the length of each episode is uncertain, we design a dynamic allocating mechanism to break one episode of transitions $(o_1, a_1, r_1, ..., o_t, a_t, r_t)$ into a group of fixed-length sequences of transitions, saved in a prioritized batch replay buffer. A learner samples the sequences based on their priorities, updates its networks, and sends the priorities of the sequences and the priorities of each individual transition back to the buffer. The actors periodically update their networks' parameters from the learner.

sensors. [21] proposed RDQN, which replaces the first post-convolutional fully-connected layer with an LSTM layer in DQN. This modification allows the agent to only see a single frame at each timestep, but is capable of replicating DQN's performance on standard Atari games. Built upon Ape-X DQN, [22] extended RDQN to R2D2, where LSTM-based agents learn from distributed prioritized experience replay. The result shows that the R2D2 agent can unprecedentedly exceed human-level performance in 52 of the 57 Atari games. Because the action space of the two algorithms is discrete, they cannot address continuous control problems in robotics.

On the robotic side, [23] used LSTM as an additional hidden layer in PPO to train a five-fingered humanoid hand to manipulate a block. The memory-augmented method was a key factor in successfully transferring the policy trained in randomized simulations to a real robotic hand, suggesting that the use of memory could help the policy to adapt to a new environment. [24] used a Q-learning based method with two LSTM layers for Q-function approximation to solve low-tolerance peg-in-hole tasks. Although memory-augmented policies have proven to improve training results in continuous control problems, neither method investigated in partially observable tasks with minimal set of observations.

III. METHOD

In this section, we introduce the setup of the RL training environment, explain the details of the RD2 agent, and describe our method to transfer robot-agnostic polices to robotic arms.

A. Setup

For both training and deployment, we use an internal simulator with the Bullet [25] physics engine.

Observation: The observation space is 6-dimensional, being the F/T measurements $(f_x, f_y, f_z, \tau_x, \tau_y, \tau_z)$ from the sensor, which is mounted at the robot end-effector.

Action: The action space is continuous and 6-dimensional, being the desired Cartesian-space linear velocity (v_x, v_y, v_z) and angular velocity (w_x, w_y, w_z) at the center of the assembly piece under control.

Reward: We use a simple linear reward function based on the distance between the goal pose and the current pose of the moving joint member. Additionally we use a large positive reward (+100) if the current pose is within a small threshold of the goal pose:

$$r = \begin{cases} -|g-x|, & |g-x| > \varepsilon \\ -|g-x| + R, & |g-x| \le \varepsilon \end{cases}$$

where x is the current pose of the joint member, g is the goal pose, ε is the distance threshold, and R is the large positive reward. We use the negative distance as our reward function to discourage the behavior of loitering around the goal because the negative distance also contains time penalty.

Note that the reward function is only used during training in simulation, where distance is easy to acquire, and is not used during rollouts. Hence, no vision or spatial tracking system is needed when deploying the policies on real robots.

Termination: An episode is terminated when the distance between the goal pose and the pose of the joint member is within a pre-defined threshold or when a pre-defined number of timesteps are reached.

B. The Recurrent Distributed DDPG Agent

We propose Recurrent Distributed DDPG (RD2) to solve the partially observable assembly tasks. Built upon Ape-X DDPG, RD2 adds an LSTM layer between the first

fully-connected layer and the second fully-connected layer in both the actor and the critic networks as well as their target networks. No convolutional layers are added since we explore in the low-dimensional observation space. The details of the network architecture is provided in Table I.

Layer	Input Size	Output Size
Fully-connected (Relu)	[#Observations]	[256]
LSTM (Relu)	[256]	[256]
Fully-connected (Q Network)	[256]	[1]
Fully-connected (Tanh) (π Network)	[256]	[#Actions]

TABLE I: Specification of the Q network and π network used in RD2 for all the experiments. It also applies to their target networks.

In the experience replay buffer, we store fixed-length sequences of transitions. Each sequence contains (m, $m=2k$, where $k \in \mathbb{Z}^+$) transitions, each of the form ($observation, action, reward$). Adjacent sequences overlap by $m/2$ timesteps and the batches of sequences never cross the episode boundary. As the length of each episode in the assembly tasks varies, which may result in some sequences containing transitions from two episodes if we naively segment transitions into fixed-length sequences, we introduce a dynamic mechanism to allow the last overlap in each episode to be a variable between $[m/2, m-1]$. Specifically, the last overlap is calculated as:

$$O = \begin{cases} m - T \bmod (m/2), & T \bmod (m/2) \neq 0 \\ m/2, & T \bmod (m/2) = 0 \end{cases}$$

where O is the number of transitions in the last overlap and T is the total timesteps in each episode. This mechanism prevents losing or compromising any transitions at the end of each episode, which usually contain crucial information for training.

Similar to R2D2, we sample the sequences in the replay buffer based on their priorities, formulated as:

$$p = \eta \max(\delta) + (1-\eta)\bar{\delta}$$

where δ is a list of absolute n-step TD-errors in one sequence. We set η to 0.9 to avoid compressing the range of priorities and limiting the ability to pick out useful experience.

In addition, as discussed in [2], prioritized replay introduces bias because it changes the distribution of the stochastic updates in an uncontrolled fashion, and therefore changes the solution that the estimates converge to. For each transition in a sequence, we correct the bias using importance-sampling weights:

$$w_i = (N \times P(i))^{-\beta}$$

where N is the size of the replay buffer and we set β to 0.4. We normalize the weight of each transition before sending the sequences for backpropagation through time (BPTT) [26] by $1/max_i w_i$. On the implementation level, we initialize two sum-tree data structures such that one keeps the priorities of

the sequences and the other one keeps the priorities of the transitions. We observe that this step is crucial to stabilize the training process for our tasks. The details of the RD2 architecture is shown in Fig.2.

We use a zero start state in LSTM to initialize the network at the beginning of the sampled sequence and train the RD2 agent with Population Based Training (PBT) [27] on an AWS p3.16xlarge instance. Every training session includes 8 concurrent trials, each of which contains one single GPU-based learner and 8 actors. We make the size of the batches, the length of the sequences, and n-step as mutable hyper-parameters to PBT. Each trial evaluates in every 5 iterations to determine whether to keep the current training or copy from a better trial. If a copy happens, the mutable hyper-parameters are perturbed by a factor of 1.2 or 0.8 or have 25% probability to be re-sampled from the original distribution. The details of the hyper-parameters fine-tuned in PBT are provided in Appendix.

C. Robotless-to-Robot Transfer

As we focus on the contact-rich phase of assembly, our tasks can be confined within the robot workspace that is collision free and singularity free. Moreover, robot motion in our tasks is slow enough to be considered quasi-static, and hence, the robot dynamics and inertial forces can be ignored.

In order to transfer policies trained in the robotless environment to the deployment environment of a specific robotic arm, we consider both the observation and the action spaces. For the observation space, we apply a coordinate transformation to the F/T measurements, using the force-torque twist matrix. If we denote F/T from its coordinates in the robotless environment (frame $\mathscr{F}_b$) as $^bh = (^bf, ^b\tau)$, and denote F/T from its coordinates in the end-effector of a robotic arm (frame $\mathscr{F}_a$) as $^ah = (^af, ^a\tau)$, then the transformation is shown in Eq. 2.

$$^ah = \begin{bmatrix} ^aR_b & 0_3 \\ [^at_b]^aR_b & ^aR_b \end{bmatrix} {}^bh \tag{2}$$

Where ^aR_b and ^at_b are the rotation matrix and the translation vector respectively from frame $\mathscr{F}_a$ to frame $\mathscr{F}_b$.

The action space is defined as the Cartesian-space velocity at the center of the assembly piece under control, which is identical across different robotic arm setups. Hence, there is no transformation needed for actions.

IV. EXPERIMENTS

In this section, we answer the following questions: (1) How does RD2 compare to the state-of-the-art RL algorithms on robotic assembly tasks with the force/torque as the only observation? (2) How does the difficulty of the tasks affect the performance of RD2 and its comparative algorithms? (3) How do the robot-agnostic policies perform on various robotic arms and with different initial poses as well as with noise injected in F/T and friction?

To answer these questions, we create two customized robotless assembly environments, **lap-joint** and **peg-in-hole**,
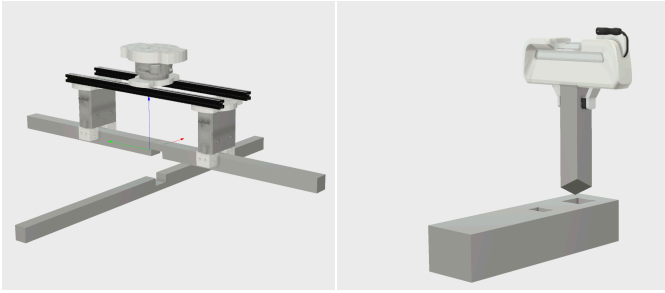
Fig. 3: Visualizations of the two robotless assembly environments: the lap-joint environment on the left has 2mm assembly tolerance and the peg-in-hole environment on the right has 0mm tolerance.

to evaluate the performance of RD2 in comparison to SOTA RL algorithms in the continuous action domain. To access the adaptability of the trained robot-agnositc polices, we rollout the policies on three robotic arms, Franka Panda, Kuka KR60, and UR10, with varying initial offsets and physical noise injected in simulation. We use an internal simulator that simulates the available robots in our lab and contains a variety of embedded drivers of the physical hardware [1]. The video presenting our experiments is available at https://sites.google.com/view/rd2-rl

A. Robotless Assembly Environments

In the training environment in simulation, both the lap-joint and peg-in-hole tasks are performed without robots. For each environment, we only include objects contributing to F/T readings. Specifically, the lap-joint environment consists of a model of a customized gripper, of a sensor, and of a pair of joint members, shown in Fig. 3(left); the peg-in-hole environment consists of a model of the Franka Panda gripper, of a sensor, of a tampered peg and a hole, shown in Fig. 3(right). Every single dynamic object is assigned an estimated inertial property (mass and centre of mass) and applied friction. The F/T sensors are gravity compensated. The tolerance of the lap-joint task is 2mm and the tolerance of the peg-in-hole task is 0mm.

B. RD2 Performance

We design 5 tasks for each environment with varying complexity, by setting the gripper at different initial poses, so the joint member or the peg on the gripper has an angular offset or a linear offset from its default pose. In general, the larger the offset the more difficult it is to train. We compare RD2 to (1) Ape-X DDPG, which has the similar architecture of RD2 but without memory augmentation and the replay buffer improvements in RD2, (2) LSTM-PPO, which performs on-policy training with memory augmentation. [2] We plot the average reward reached by the agent against the number of timesteps of training for each algorithm, as shown in Fig.4

[1]The internal simulator is built for easy deployment of sim-trained policies on physical robots, which unfortunately due to COVID-19 we have not been able to access.
[2]Same as RD2, the two algorithms also trained with PBT with 8 concurrent trials, each of which contains one single GPU-based learner and 8 actors.

for the lap-joint tasks and Fig.5 for the peg-in-hole tasks. Note that positive reward in the plots indicates successful assemblies and the higher the positive reward, the higher the success rate there is for the assembly tasks.

Both figures show that RD2 outperforms the other two algorithms across all the tasks. As the difficulty increases in each environment, RD2 keeps a stable performance while the performance of the other two algorithms drop significantly. In general, the off-policy algorithms (RD2 and Ape-X DDPG) perform better than the on-policy algorithm (LSTM-PPO) on our tasks, which suggests the importance of sample efficiency for partially observable environments. However, the declined performance of Ape-X DDPG on harder tasks in both environments indicates the necessity of our improvements on RD2, especially in real-world settings where small misalignments are inevitable.

C. Evaluation of Adaptation

In this section, we evaluate how well the robotless policies transfer to different robotic arms and how they generalize to different initial pose offsets and different physical noises. Specifically, we take into account the following factors: linear and angular offset, and Gaussian noise in F/T measurements and in friction.

We use the lap-joint task environment for evaluating different initial pose offsets. Table II shows the result of evaluating the same robotless policy on three different robotic arms, with 0 offset and with position and orientation offsets. We show the success rate over 10 runs. The trained robotless policy transfers well to different robots and it generalizes well to different initial position and orientation offsets for the Panda and UR10 robots. In future work we will investigate in the implementation of our internal simulator why the policy does not generalize as well on the Kuka robot.

Robots	0 offset	3mm offset along XYZ	5 deg offset around Z
Franka Panda	100%	100%	100%
UR10	100%	90%	90%
Kuka KR60	100%	40%	30%

TABLE II: We show the success rate of 10 rollouts of the same robotless policy of the lap-joint task on three different robotic arms: Panda, UR10 and Kuka. The same policy is also tested with varying initial position and orientation offsets

We use the peg-in-hole task environment for evaluating physical noises. Table III shows the result of evaluating the same robotless policy on three different robotic arms, with no noise and with Gaussian noise in F/T measurements and friction. We show the success rate over 10 runs. The policy generalizes well when Gaussian noise of 0 mean and 20% variance added to F/T measurements and to friction.

We further evaluate the policies on assembly pieces that are placed at different poses as opposed to the training pose, as shown in Fig.6. As it is common to have the same assembly at various poses during construction, it is important for a trained policy to be able to generalize to different poses.
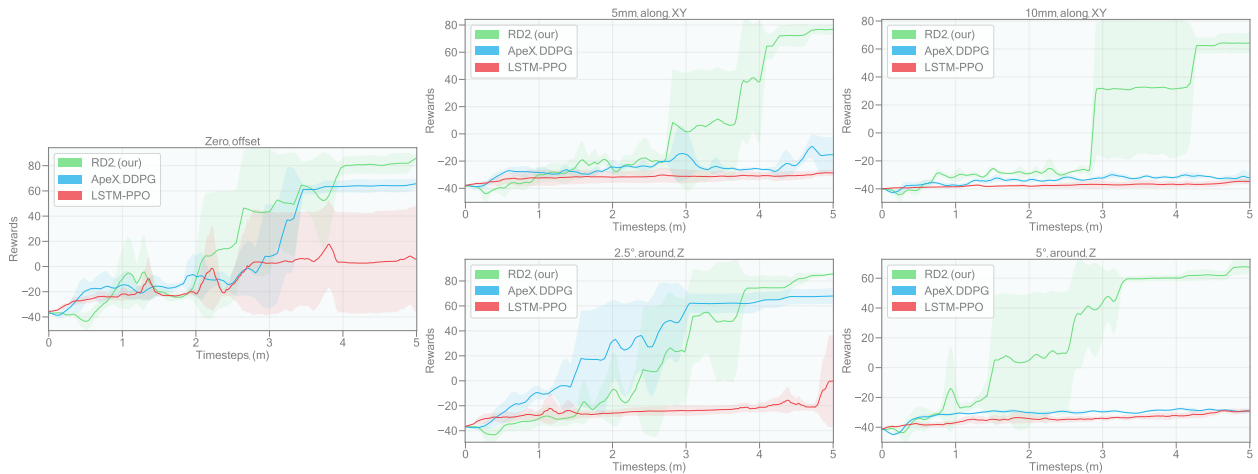
Fig. 4: Rewards comparison for the **lap-joint** tasks. The difficulty of the tasks increases from left to right. The lines visualize the average of the best model performance across time for three PBT runs with different random seeds and the shaded areas show the 95% confidence bound.
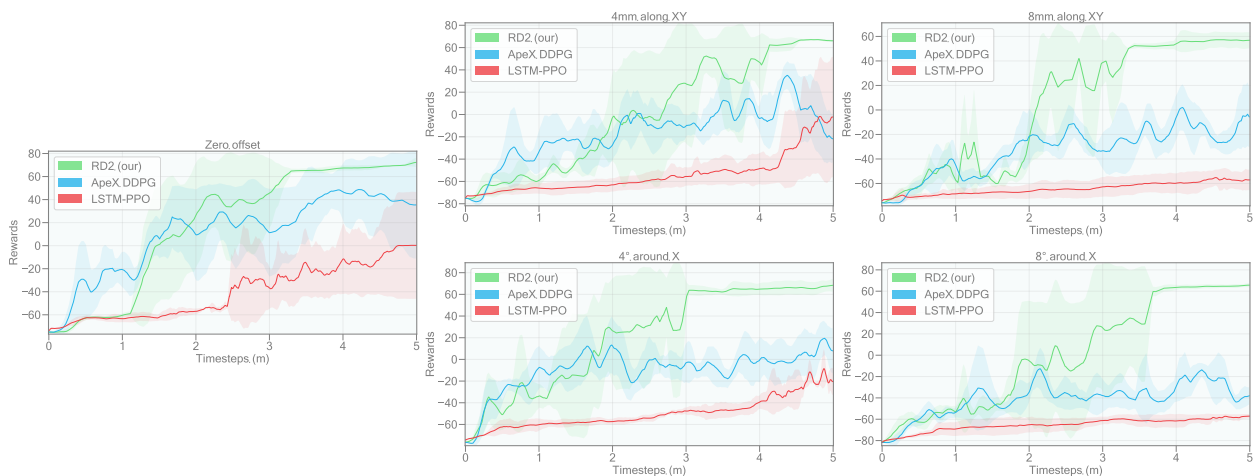

Fig. 5: Rewards comparison for the **peg-in-hole** tasks. The difficulty of the tasks increases from left to right. The lines visualize the average of the best model performance across time for three PBT runs with different random seeds and the shaded areas show the 95% confidence bound.

Robots	0 noise	20% F/T noise	20% friction noise
Franka Panda	100%	80%	100%
UR10	100%	80%	90%
Kuka KR60	100%	90%	100%

TABLE III: We show the success rate of 10 rollouts of the same robotless policy of the peg-in-hole task on three different robotic arms: Panda, UR10 and Kuka. The policy is tested with Gaussian noise of 0 mean and 20% variance added to F/T, as well as with Gaussian noise of 0 mean and 20% variance added to friction.

Fig. 6: Visualizations of evaluating the trained policies on assembly pieces at different poses from the training pose.

For each task, when we transform the actions to be in the frame of the target during training, our trained policy can successfully transfer to tasks with varying target poses.

V. CONCLUSION AND FUTURE WORK

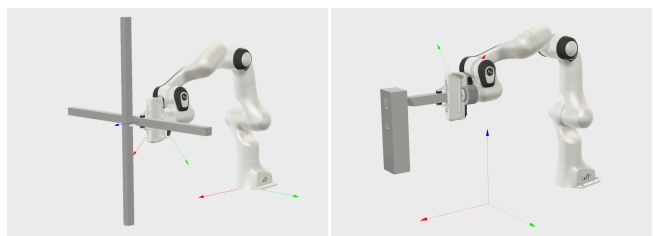This paper presented a learning approach to solving high-precision robotic assembly tasks with the focus on the contact-rich phase. We use F/T measurements as the only observation. The approach allows different robots to share one trained policy and to adapt to misalignment without requiring external position tracking or vision systems, which improves the capability and scalability of RL-based robotic systems in unstructured settings, such as architectural construction. To achieve it, RD2 learns a memory-based representation

to compensate for partial observability. Training takes place in robotless environments and trained polices can transfer to various robotic arms without re-training. Our results show that RD2 achieves the best performance on all assembly tasks in comparison to two baselines, Ape-X DDPG and LSTM-PPO. Furthermore, RD2 demonstrates its strength on harder tasks that cannot be solved by either baseline algorithm. We also show that our trained policies transfer well to different robotic arms, and can adapt to various initial pose misalignment as well as noise injected in F/T measurements and friction parameters.

Future work includes comparing our approach to augmenting F/T measurements with vision as observations. When hardware access becomes possible, we will deploy the trained polices on different physical robots to further assess the approach.

Although the reward function is solely used during training in simulation, where distance is easy to acquire, and not used during rollouts, we suspect using distance in the reward function may be detrimental to the adaptability of trained policies. We will investigate reward learning in the future.

APPENDIX

A. The hyper-parameters fine-tuned in RD2 using PBT

Fine-tuned Hyperparameter	Fine-tuned range
Number of batches	$(20, 120)^1$
Sequence length	$[16, 32, 64, 128]^2$
Target network update frequency	[25000, 50000, 75000, 100000]
N-step	(3, 8)
Minimal iteration time (s)	[30, 40, 50, 60]

[1] Sampled from the given range, [2] Sampled from the given list

TABLE IV: The hyper-parameters fine-tuned in RD2 using PBT

ACKNOWLEDGMENT

REFERENCES

[1] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," *arXiv preprint arXiv:1803.00933*, 2018.

[2] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[3] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real-world robotic reinforcement learning," *arXiv preprint arXiv:2004.12570*, 2020.

[4] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. Mc-Grew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[5] T. Ren, Y. Dong, D. Wu, and K. Chen, "Learning-based variable compliance control for robotic assembly," *Journal of Mechanisms and Robotics*, vol. 10, no. 6, p. 061008, 2018.

[6] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," *arXiv preprint arXiv:1906.05841*, 2019.

[7] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[8] J. Luo and H. Li, "Dynamic experience replay," *arXiv preprint arXiv:2003.02372*, 2020.

[9] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," *Applied Sciences*, vol. 10, no. 19, p. 6923, 2020.

[10] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," in *6th International Conference on Learning Representations*, 2018.

[11] 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," in *6th International Conference on Learning Representations*, 2016.

[12] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," *arXiv preprint arXiv:1512.04455*, 2015.

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

[14] 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 518, pages 529–533*, 2015.

[15] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *arXiv preprint arXiv:1802.01561*, 2018.

[16] Y. Fan, J. Luo, and M. Tomizuka, "A learning framework for high precision industrial assembly," *arXiv preprint arXiv:1809.08548v3*, 2018.

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

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[19] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *2019 International Conference on Robotics and Automation (ICRA).* IEEE, 2019, pp. 8943–8950.

[20] A. Apolinarska*, M. Pacher*, H. Li*, N. Cote, R. Pastrana, F. Gramazio, and M. Kohler, "Robotic assembly of timber joints using reinforcement learning," *Automation in Construction Journal, https://doi.org/10.1016/j.autcon.2021.103569*, vol. 125, 2021.

[21] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 AAAI Fall Symposium Series*, 2015.

[22] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement learning," in *International conference on learning representations*, 2018.

[23] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1808.00177

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

[25] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository*, 2016.

[26] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[27] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan *et al.*, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.