Imitation Learning for High Precision Peg-in-Hole Tasks

Sagar Gubbi[*], Shishir Kolathaya[*], Bharadwaj Amrutur
Robert Bosch Center for Cyber-Physical Systems, Indian Institute of Science, Bengaluru,
India
email: {sagar,shishirk,amrutur} @iisc.ac.in

Abstract— **Industrial robot manipulators are not able to match the precision and speed with which humans are able to execute contact rich tasks even to this day. Therefore, as a means to overcome this gap, we demonstrate generative methods for imitating a peg-in-hole insertion task in a 6-DOF robot manipulator. In particular, generative adversarial imitation learning (GAIL) is used to successfully achieve this task with a 6 μm peg-hole clearance on the Yaskawa GP8 industrial robot. Experimental results show that the policy successfully learns within 20 episodes from a handful of human expert demonstrations on the robot (i.e., < 10 tele-operated robot demonstrations). The insertion time improves from > 20 seconds (which also includes failed insertions) to < 15 seconds, thereby validating the effectiveness of this approach.**

Keywords-imitation learning; generative adversarial networks; peg-in-hole insertion

I. INTRODUCTION

High precision assembly tasks refer to the class of tasks where the position accuracy required far exceeds that of the robot manipulators. For example, the Yaskawa GP8 robot (shown in Fig. 1) has accuracy levels of $20 - 100$ μm, whereas some of the peg-in-hole insertion tasks require accuracy levels of ≤ 10 μm. More importantly, these types of insertion tasks are contact-rich (as shown by Fig. 1), thereby requiring limits on the maximum allowable forces exerted by the peg. Therefore, executing these types of tasks remain a challenge for traditional control schemes even to this day.

Classical methods used for the peg-in-hole task involve the application of a transimpedance controller rather than a direct position or velocity control of the joint angles [1, Chapter 8.8]. A transimpedance controller realizes active compliant behavior of the end-effector, thereby enabling peg-in-hole insertions despite the lack of positioning accuracy of the manipulators. It is important to note that the forces normal to the surface must be controlled to prevent any form of damage to the peg or the hole. However, even if the stiffness and damping parameters of the impedance controller are well tuned, the insertion times are high (> 20 seconds) for tight clearances (≤ 10 μm). This is in contrast to insertion times taken by human experts, which are < 5 seconds. Hence, the question that we would like to answer is, can we mimic this human expert policy in the robot via one of the well known imitation learning methods available in literature?

It is worth noting that modeling the contact is difficult, but the policy employed by human experts to insert the peg
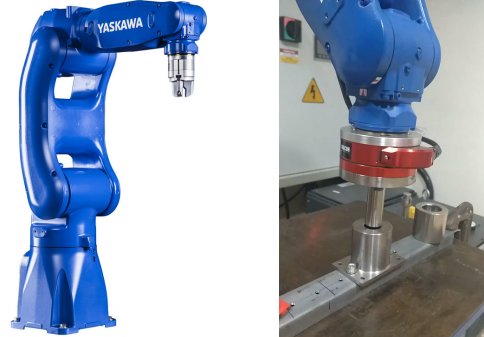
Figure 1. The Yaskawa GP8 robot is shown on the left, and the peg being inserted by this robot into a hole with a gap of 6 μm is shown on the right.

is seemingly simple. In imitation learning, a policy network is trained to copy the expert and take the same actions that the expert would have taken given the same observations [2], [3]. It is a robust skill acquisition method that has been successfully applied to autonomous driving [4] and complex manipulation tasks [5]. Therefore, the goal of the paper is to mimic an expert policy via imitation learning for the peg-in-hole task.

Imitation learning requires expert data. In some domains, such as autonomous driving, the control interface makes this straight forward to procure. The observations from the camera and other sensors can be recorded along with the driver's actions on the steering wheel, accelerator, and brake pedals. Expert data can also be collected for complex manipulation tasks by teleoperating a robot with a VR controller [2]. For high precision assembly tasks, it is much harder to collect expert data. Although humans can reliably place a peg in a hole, it is difficult to capture the forces applied and the feedback forces received. Hence, we chose to gather useful expert data for the peg-in-hole task by teleoperation. Since the amount of expert data is limited in this context, the question we address in this work is: Can imitation learning succeed in copying an expert with a small set of demonstrations?

To address this issue, we first build a teleoperation system that uses a space mouse (see Fig. 5) to control a Yaskawa GP8 robot. The feedback forces received by the MotoFit force sensor are plotted on a display, and the expert can exert forces on the peg using the space mouse. The feedback forces and the applied forces are recorded as the expert performs the peg-in-hole task. The recorded data is used to train a neural network that takes the force feedback and the current position and produces the force to be applied. We specifically
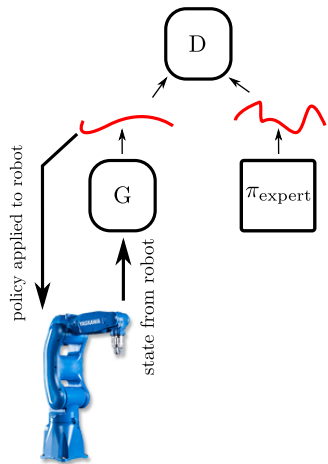
Figure 2. Figure showing the GAIL framework used for the peg-in-hole task. The generator is denoted by G, and the discriminator is denoted by D, and the expert policy is denoted by π_{expert}.

use generative adversarial imitation learning (GAIL) [3] for learning the policy, which are known to be sample efficient. Pictorial representation of GAIL is provided in Fig. 2.

Our contributions are:

- We show that a neural network can be trained via generative methods to copy the actions taken by a human expert.
- We find that only a handful of expert trajectories (less than ten) are sufficient for the peg-in-hole task to achieve a high success rate.

The remainder of this paper is structured as follows. In Section II, we discuss related work. Following this in Section III, we describe the proposed architecture. Subsequently, we present experimental results in Section IV and conclude this paper with a short discussion in Section V.

II. RELATED WORK

Deep reinforcement learning (D-RL) has been used to learn controllers for a variety of tasks ranging from walking robots [6], [7], [8] to manipulating objects with an arm [9], [10], [11], [12], [13]. Hence reinforcement learning, indeed, offers a way to realize peg-in-hole tasks via random explorations, thereby eliminating the need to hand craft an effective control/policy without using any form of expert data. However, the remarkable success observed in D-RL cannot be translated directly to realize high precision assembly tasks. One of the factors being the hard limit on the number of allowable training iterations in hardware. As the tasks become more complex, the training becomes more expensive. Sample efficiency of the learning algorithm is critical to directly deploy on real robots [14]. For example, in [15], the task of inserting the peg inside the hole defines a reward function and uses Q learning to arrive at a policy that, given the measured feedback forces, outputs the forces to be applied. The minimum number of episodes required for learning a stable policy for a peg-hole clearance of 10 μm

was 100. Therefore, with a view toward reducing the number of training episodes, the goal of this paper is to use imitation learning to copy the policy of human experts. We find that imitation is significantly more sample efficient; uses < 20 episodes to fully learn the policy for pegs with tighter clearances i.e., 6 μm.

[16] proposed to use an analytical model of the contact to develop better policies for the peg-in-hole insertion tasks. The reactionary forces when a peg is pushed against the hole at an arbitrary angle are recorded, and the model is used to solve for the actual orientation of the peg. Once the orientation is known, the peg is pushed in at the appropriate angle. In our work, we assume the peg is aligned with the hole within 2 degrees and obtain a policy that can push the peg inside the hole and react appropriately if the peg is stuck mid way through the hole. We do not attempt to model the contact, so our method is model-free.

Visual imitation learning was proposed in [17], [2], [5], [18] to copy an expert performing complex manipulation tasks using a virtual reality controller. The tasks accomplished include inserting blocks into shape sorting cubes. Although our work is superficially similar, there are several differences. First, we do not use vision because we assume that the peg is close to the hole at the starting position. Second, our objective requires significantly higher precision because the gap between the peg and the hole is only 6 μm. This necessitates the use of impedance control using a force sensor rather than position/velocity control.

III. PROPOSED ARCHITECTURE

In imitation learning, behavior cloning [2] and generative adversarial imitation learning (GAIL) [3] are two approaches to train a policy network. In behavior cloning, once the dataset is collected, it is used to train a policy network through supervised learning, and the robot does no further exploration during learning. In GAIL the reward function is inferred from expert data, and the inferred reward function is used to train the controller. GAIL has the potential to enable the robot to generalize better while using limited training data by training the policy network with reinforcement learning [3]. This is similar to inverse reinforcement learning where a reward function is extracted from the expert data. However, in GAIL, the reward function is never explicitly recovered from the expert data. Instead, a discriminator network (Fig. 3) is trained to distinguish between expert trajectories and trajectories generated by the generator policy network (see Fig. 2). The trained discriminator is then used as a reward function to train the generator network (for example, with PPO [19]) so as to confuse the discriminator. Thus, training the generator and discriminator proceed alternately in GAIL, and the generator improves over time. Neural network models for both the generator and discriminator are shown in Figs. 4, 3. See [3] for more details for the detailed description for the control algorithm for GAIL implemented in the robot.

After successful training, the generator network becomes our policy network. The inputs to the policy network are:
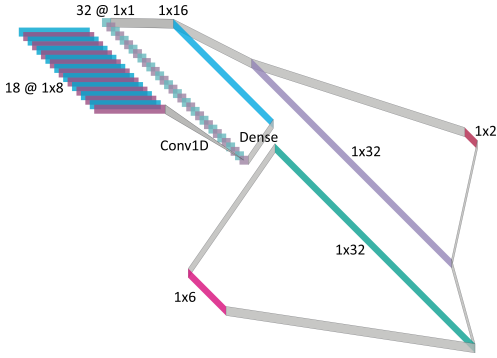
- x, y, z positions of the peg: p_x, p_y, p_z.

Figure 3. The discriminator network that takes (state, action) pairs as input and classifies if the pair comes from expert data or from the generator (policy) network.
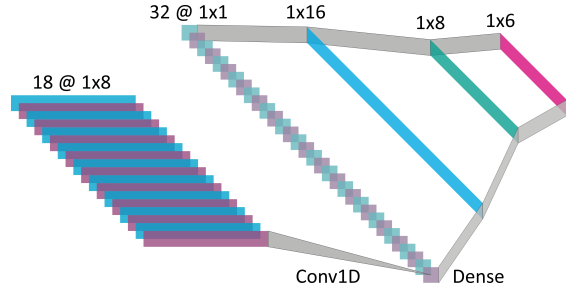

Figure 4. The policy network (generator) that takes the current state (current force feedback and position) and outputs the actuator command (target forces) that are fed as input to the transimpedance controller.
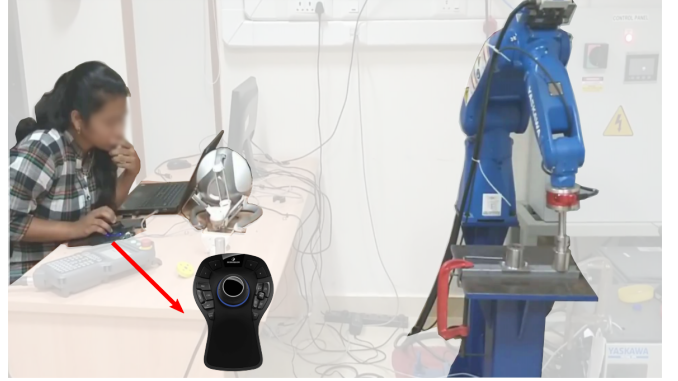
- Roll, pitch and yaw positions of the peg: p_{r_x}, p_{r_y}, p_{r_z}.

The output of the policy network determines if the peg should apply a downward force, apply wiggle in every time step. Therefore, the action (A_p) is a discrete variable, i.e., wiggle[1] with a downward force, wiggle without a downward force, apply a downward force without wiggle, and apply a downward force with wiggle. Hence, there are four actions. We will number the actions from 0 to 3 with each action described as follows:

- 0: Downward force with wiggle
- 1: Downward force without wiggle
- 2: Wiggle without downward force
- 3: No wiggle and no downward force

The policy network is shown in Fig. 4. The first layer is the LayerNorm normalization layer and is followed by a 1D convolutional layer along the time axis which accepts the state input from the last few time steps. This is followed by a few densely connected layers to produce the discretized target force outputs. There are two notable aspects to this network:

- At the very beginning, the first layer is a normalization layer with large epsilon. We explain below why this is necessary.
- Even though stateful elements such as LSTM are not used, the network is recurrent because the previous outputs are included as inputs. This is needed to produce output waveforms that look like, for example, a square wave.

The layer normalization is defined to be the following:

$$LN(x_i) = \frac{x_i - mean(x_i, x_{i-1}, \ldots, x_{i-T+1})}{\epsilon + std(x_i, x_{i-1}, \ldots, x_{i-T+1})}, \quad (1)$$

where the $mean$ and std are over the time dimension. The subscripts i, $i-1$, ... are the time indices for states, with T being the sample size. Subtraction of the mean is necessary because of the large range of values the input might take. For example, the position p_z is fed in mm and can vary from -170 mm to $+300$ mm depending on the height of the

[1]Here wiggle means random forces being applied along roll, pitch and yaw directions.

table, whereas the change in p_z during insertion may only be 20 mm (i.e., the height of the hole). We also use a large $\epsilon = 0.1$ in order to suppress noise when one of the inputs is mostly static and unchanging over time.

IV. RESULTS

With the network architecture described, we now discuss the main results. The experiments were performed on a Yaskawa GP8 robot with the MotoFit 6-axis force sensor. We wrote a program for the YRC1000 controller using the MotoPlus API in C that communicates with a PC via UDP. The average round trip latency between the PC and the YRC1000 controller was less than 1 ms. The neural network was implemented using Keras on a Linux PC. The sampling rate for collecting the sensor data and to update the target force was 100 Hz. We assume that the position of the hole relative to the base of the robot is known and the peg is already positioned to be just above the hole (with an error of $+/-1$ mm, and with a randomly chosen roll and pitch angle (i.e., between -2 and $+2$ degree with respect to the hole). Therefore, a camera is not necessary. However, even if the peg is precisely placed above the hole, plain insertion i.e., applying a downward force on the peg, is not guaranteed to be successful every time. This is due to the fact that the position accuracy of the robot is > 20 μm. The transimpedance controller serves as the low level controller that receives actuator commands from the policy network.

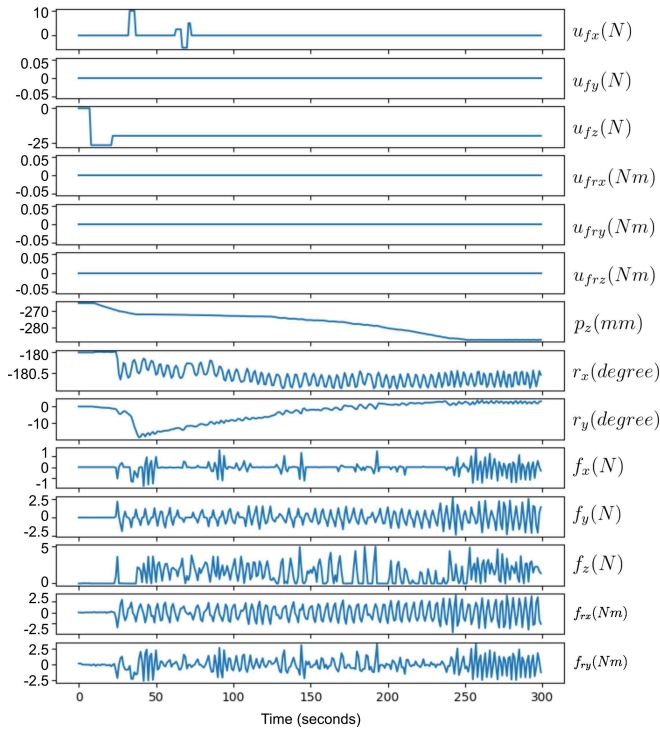Figure 5. The space mouse used by the human expert to guide the peg into the hole.

Figure 6. A sample episode where the expert guides the peg into the hole using the space mouse. The plots are described as follows (from top to bottom): force along x direction (uf_x), force along y direction (uf_y), force along z direction (uf_z), moment about x direction (uf_{rx}), moment about y direction (uf_{ry}), moment about z direction (uf_{rz}), vertical position of peg p_z, roll of peg r_x, pitch of peg r_y, force measured along x, force measured along y, force measured along z, moment measured about x, and moment measured about y.

A. Data collection

Collecting expert data proved to be a challenge. One option was to detach the peg from the robot while leaving it connected to the force sensor and to have the expert insert it in the hole. The problem with this approach is that it is difficult to measure what forces the expert is applying on the peg. Instead we chose to display the force sensor readings along with the position of the peg on a screen and to have the expert apply a force on the peg using a space mouse (Fig. 5). The lack of haptic feedback is a major drawback of this approach. Nevertheless, the experts were successful in inserting the peg.

Fig. 5 shows the expert operating the space mouse to guide the peg into the hole. A constant downward force is applied during insertion. The expert can apply forces in the X and Y direction using the space mouse. Although a torque in R_z can be applied, none of the experts did so. A sample episode where the expert successfully inserts the peg is shown in Fig. 6. We collected 8 such episodes for training.

B. GAIL Training

Roughly 500 samples from the MotoFit algorithm were used for Generator. Discriminators are trained with cross-entropy loss, and the generators are trained with the PPO loss

[19]. After every episode equal number of samples are chosen both from the generated and expert data, and the discriminator is trained for about 100 − 500 iterations. Similarly, after every episode, the discriminator is evaluated for the samples and used as reward for the generator. With these rewards, the generator is trained via PPO. It is worth noting that the frequency of update for the generator is much higher than that of the discriminator. This is due to the fact that the discriminator tends to learn faster than the generator. The above training framework is run for about 20 episodes in the robot, and the results are obtained. Fig. 7 shows the trajectories of vertical position, roll and pitch angles of the peg during some of the training episodes. It can be verified that the "wiggle" mode with a downward force is turned on whenever the peg is stuck.

Figs. 8 show the generator reward and the discriminator loss as a function of the episodes. It is worth noting that the generator and discriminator are playing a game, and there is no termination condition. Both the losses must stabilize to a constant value (or oscillates), which are nonzero. Fig. 9 shows the comparison between the untrained and trained networks. As shown by Fig. 9, the action values applied before training are arbitrary, and those applied after training are more meaningful. It can be verified that the insertion time is also improved (from Fig. 8) after about eight episodes. A video demonstrating the training of peg-in-hole is provided in this link: https://youtu.be/ztyt94rsG5s.

V. CONCLUSION

We showed that high precision assembly tasks such as peg-in-hole insertions with small clearances can be trained using imitation learning. Imitation learning is more sample efficient t han r einforcement l earning a nd d oes n ot require reward shaping. We used generative methods for imitating the expert policy. However, it is worth noting that gathering expert data for such tasks still remains a challenge. Future work will involve using better data collection techniques and performing imitation learning on a wider array of tasks.

ACKNOWLEDGMENT

REFERENCES

[1] A. Ghosal, *Robotics: fundamental concepts and analysis*. Oxford university press, 2006.

[2] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[3] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

[4] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.
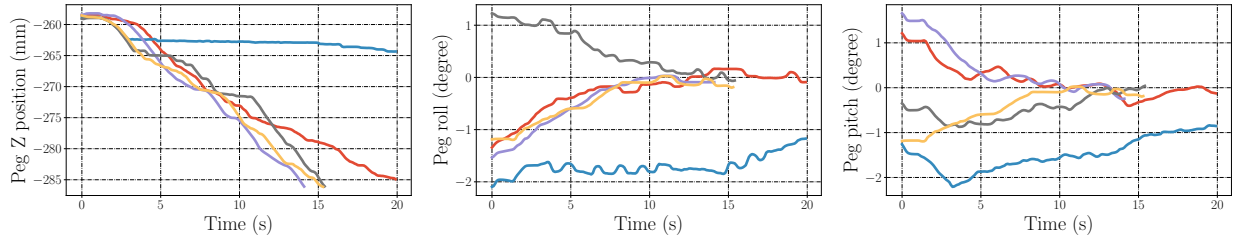
Figure 7. Position of the peg p_z, roll of the peg p_{rx}, and pitch angle of the peg p_{ry} are given here for 5 episodes (randomly chosen from the 20). It can be verified that the initial pose varies from −2 to 2 degrees.
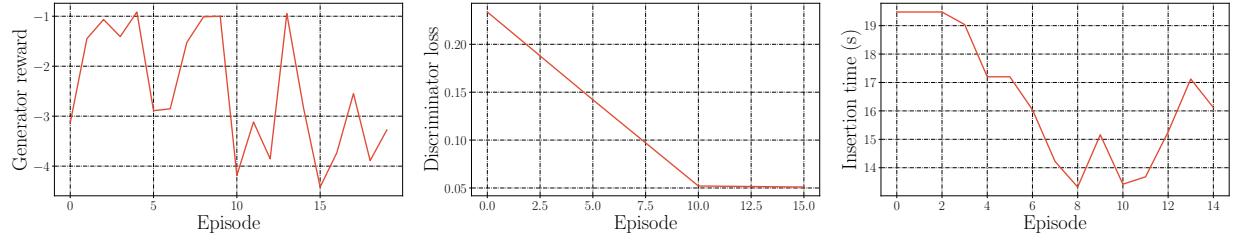

Figure 8. Figures showing the generator reward and the discriminator loss and the mean insertion time (averaged over 5 consecutive episodes) for twenty episodes. The generator reward oscillates, while the discriminator loss stabilizes after approximately ten episodes. More importantly, the mean insertion time is decreasing over time.
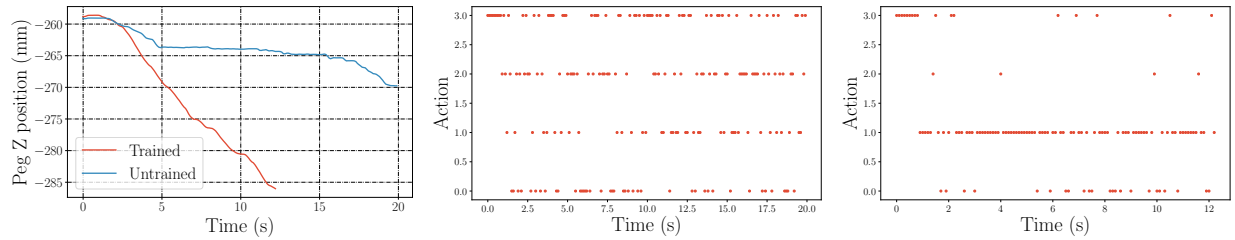

Figure 9. Left figure is showing the comparison between the results with the untrained and trained networks. Actions from an untrained (middle) and a trained (right) network are also shown. The actions from the untrained network are arbitrary, while the trained network yields a more meaningful policy.

[5] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," *arXiv preprint arXiv:1802.01557*, 2018.

[6] D. Dholakiya, S. Bhattacharya, A. Gunalan, A. Singla, S. Bhatnagar, B. Amrutur, A. Ghosal, and S. Kolathaya, "Design, development and experimental realization of a quadrupedal research platform: Stoch," in *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, April 2019, pp. 229–234.

[7] S. Bhattacharya, A. Singla, Abhimanyu, D. Dholakiya, S. Bhatnagar, B. Amrutur, A. Ghosal, and S. Kolathaya, "Learning active spine behaviors for dynamic and efficient locomotion in quadruped robots," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, Oct 2019, pp. 1–6.

[8] S. Kolathaya, A. Joglekar, S. Shetty, D. Dholakiya, Abhimanyu, A. Sagi, S. Bhattacharya, A. Singla, S. Bhatnagar, A. Ghosal, and B. Amrutur, "Trajectory based deep policy search for quadrupedal walking," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, Oct 2019, pp. 1–6.

[9] 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.

[10] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

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

[12] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *arXiv preprint arXiv:1704.03073*, 2017.

[13] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016.

[14] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[15] 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.

[16] T. Tang, H.-C. Lin, Y. Zhao, W. Chen, and M. Tomizuka, "Autonomous alignment of peg and hole by force/torque measurement for robotic assembly," in *2016 IEEE international conference on automation science and engineering (CASE)*. IEEE, 2016, pp. 162–167.

[17] S. Gubbi and B. Amrutur, "One-shot object localization using learnt visual cues via siamese networks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.

[18] T. Yu, P. Abbeel, S. Levine, and C. Finn, "One-shot hierarchical imitation learning of compound visuomotor tasks," *arXiv preprint arXiv:1810.11043*, 2018.

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