2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards

Gerrit Schoettler*[1], Ashvin Nair*[2], Jianlan Luo[2], Shikhar Bahl[2],
Juan Aparicio Ojea[1], Eugen Solowjow[1], Sergey Levine[2]

***Abstract*— Connector insertion and many other tasks commonly found in modern manufacturing settings involve complex contact dynamics and friction. Since it is difficult to capture related physical effects with first-order modeling, traditional control methods often result in brittle and inaccurate controllers, which have to be manually tuned. Reinforcement learning (RL) methods have been demonstrated to be capable of learning controllers in such environments from autonomous interaction with the environment, but running RL algorithms in the real world poses sample efficiency and safety challenges. Moreover, in practical real-world settings, we cannot assume access to perfect state information or dense reward signals. In this paper, we consider a variety of difficult industrial insertion tasks with visual inputs and different natural reward specifications, namely sparse rewards and goal images. We show that methods that combine RL with prior information, such as classical controllers or demonstrations, can solve these tasks from a reasonable amount of real-world interaction.**

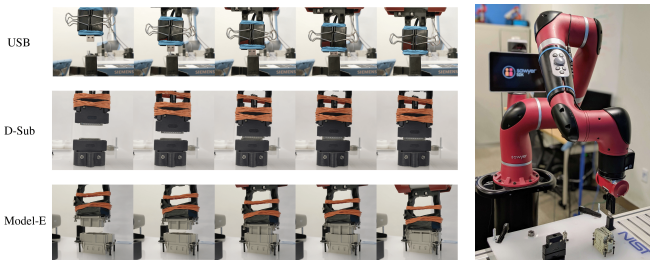

Fig. 1: We train policies directly in the real world to solve connector insertion tasks from raw pixel input and without access to ground-truth state information for reward functions. Left: a rollout from a learned policy that successfully completes the insertion task for each connector is shown. Right: a full view of the robot setup. Videos are available at `https://industrial-insertion-rl.github.io`.

I. Introduction

Many industrial tasks on the edge of automation require a degree of adaptability that is difficult to achieve with conventional robotic automation techniques. While standard control methods, such as PID controllers, are heavily employed to automate many tasks in the context of positioning, tasks that require significant adaptability or tight visual perception-control loops are often beyond the capabilities of such methods, and therefore are typically performed manually. Standard control methods can struggle in presence of complex dynamical phenomena that are hard to model analytically, such as complex contacts. Reinforcement learning (RL) offers a different solution, relying on trial and error learning instead of accurate modeling to construct an effective controller. RL with expressive function approximation, i.e. deep RL, has further shown to automatically handle high dimensional inputs such as images [1].

However, deep RL has thus far not seen wide adoption in the automation community due to several practical obstacles. Sample efficiency is one obstacle: tasks must be completed without excessive interaction time or wear and tear on the robot. Progress in recent years on developing better RL algorithms has led to significantly better sample efficiency, even in dynamically complicated tasks [2], [3], but remains a challenge for deploying RL in real-world robotics contexts. Another major, often underappreciated, obstacle is goal specification: while prior work in RL assumes a reward signal to optimize, it is often carefully shaped to allow the system to learn [4], [5], [6]. Obtaining such dense reward signals that provide a reward value at almost every visited state can be a significant challenge, as one must additionally build a perception system that allows computing dense rewards on state representations. Shaping a reward function so that an agent can learn from it is also a manual process that requires considerable manual effort. An ideal RL system would learn from rewards that are natural and easy to specify. How can we enable robots to autonomously perform complex tasks without significant engineering effort to design perception and reward systems?

We first consider an end-to-end approach that learns a policy from images, where the images serve as both the state representation and the goal specification. Using goal images is not fully general, but can successfully represent tasks when the task is to reach a final desired state [7]. Specifying goals via goal images is convenient, and makes it possible to specify goals with minimal manual effort. Using images as the state representation also allows a robot to learn behaviors that utilize direct visual feedback, which provides some robustness to sensor and actuator noise.

Secondly, we consider learning from simple and sparse reward signals. Sparse rewards can often be obtained conveniently, for instance from human-provided labels or simple instrumentation. In many electronic assembly tasks, which we consider here, we can directly detect whether the electronics are functional, and use that signal as a reward. Learn-

* First two authors contributed equally, [1] Siemens Corporation, [2] University of California, Berkeley. Correspondence: `anair17@berkeley.edu`

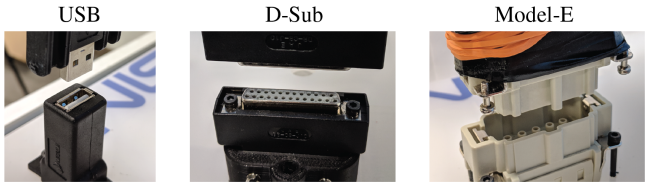

Fig. 2: A close-up view of the three connector insertion tasks shows the contacts and low tolerances the agent must navigate to solve these tasks. These tasks require sub-millimeter precision without visual feedback.

ing from sparse rewards poses a challenge, as exploration with sparse reward signals is difficult, but by using sufficient prior information about the task, one can overcome this challenge. To handle this challenge, we extend the residual RL approach [8], [9], which learns a parametric policy on top of a fixed, hand-specified controller, to the setting of vision-based manipulation.

In our experiments, we show that we can successfully complete real-world assembly tasks, such as inserting USB connectors, using RL from images with reward signals that are convenient for users to specify. We can learn from only a sparse reward based on the electrical connection for a USB connector plug, and we demonstrate learning insertion skills with rewards based only on goal images. These reward signals require no extra engineering and are easy to specify for many tasks. Beyond showing the feasibility of RL for solving these tasks, we evaluate multiple RL algorithms across three tasks and study their robustness to imprecise positioning and noise.

II. Related Work

Learning has been applied previously in a variety of robotics contexts. Different forms of learning have enabled autonomous driving [10], biped locomotion [11], block stacking [12], grasping [13], and navigation [14], [15]. Among these methods, many involve reinforcement learning, where an agent learns to perform a task by maximizing a reward signal. Reinforcement learning algorithms have been developed and applied to teach robots to perform tasks such as balancing a robot [16], playing ping-pong [17] and baseball [18]. The use of large function approximators, such as neural networks, in RL has further broadened the generality of RL [1]. Such techniques, called "deep" RL, have further allowed robots to be trained directly in the real world to perform fine-grained manipulation tasks from vision [19], open doors [20], play hockey [21], stack Lego blocks [22], use dexterous hands [23], and grasp objects [24]. In this work we further explore solving real-world robotics tasks using RL.

Many RL algorithms introduce prior information about the specific task to be solved. One common method is reward shaping [4], but reward shaping can become arbitrarily difficult as the complexity of the task increases. Other methods incorporate a trajectory planner [25] but for complex assembly tasks, trajectory planners require a host of information about objects and geometries which can be difficult to provide.

Another body of work on incorporating prior information studies using demonstrations either to initialize a policy [18], [26], infer reward functions using inverse reinforcement learning [27], [28] or to improve the policy throughout the learning procedure [29], [30], [31]. These methods require multiple demonstrations, which can be difficult to collect, especially for assembly tasks, although learning a reward function by classifying goal states [32] may partially alleviate this issue. More recently, manually specifying a policy and learning the residual task has been proposed [8], [9]. In this work we evaluate both residual RL and combining RL with learning from demonstrations.

Previous work has also tackled high precision assembly tasks, especially insertion-type tasks. One line of work focuses on obtaining high dimensional observations, including geometry, forces, joint positions and velocities [33], [34], [35], [36], but this information is not easily procured, increasing complexity of the experiments and the supervision required. Other work relies on external trajectory planning or very high precision control [35], [34], but this can be brittle to error in other components of the system, such as perception. We show how our method not only solves insertion tasks with much less information about the environment, but also does so under noisy conditions.

III. Electric Connector Plug Insertion Tasks

In this work, we empirically evaluate learning methods on a set of electric connector assembly tasks, pictured in Fig. 2. Connector plug insertions are difficult for two reasons. First, the robot must be very precise in lining up the plug with its socket. As we show in our experiments, errors as small as ± 1 mm can lead to consistent failure. Second, there is significant friction when the connector plug touches the socket, and the robot must learn to navigate under this constraint in order to insert the plug. Image sequences of successful insertions are shown in Fig. 1, where it is also possible to see details of the gripper setup that we used to ensure a failure free, fully automated training process. In our experiments, we use a 7 degrees of freedom Sawyer robot with end effector control. The action signal u_t provides the relative end effector movement in Cartesian coordinates,

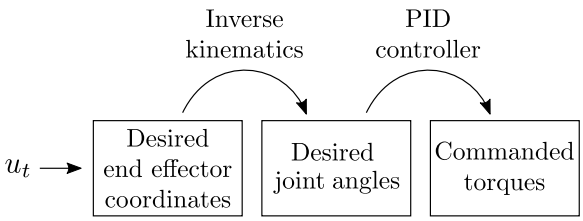

Fig. 3: Illustration of the robot's cascade control scheme. The actions u_t are computed at a frequency of up to 10 Hz, desired joint angles are obtained by inverse kinematics, and a joint-space impedance controller with anti-windup PID control commands actuator torques at 1000 Hz.

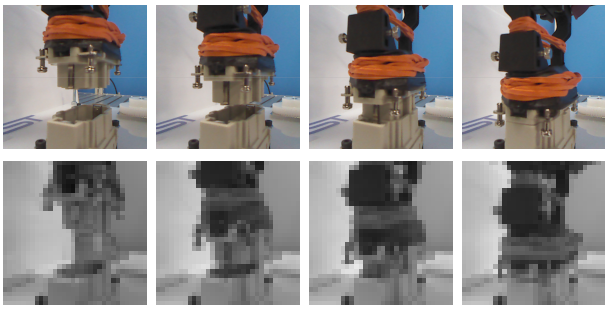

Fig. 4: Successful insertion on the Model-E connector. The image-based RL algorithms receives only the 32×32 grayscale image as the observation.

the action space is 3-dimensional. The robot's underlying internal control pipeline is illustrated in Figure 3.

To comprehensively evaluate connector assembly tasks, we experiment on a variety of connectors. Each connector offers a different challenge in terms of required precision and force to overcome friction. We chose to benchmark the controllers performance on the insertion of a USB connector, a U-Sub connector, and a waterproof Model-E connector manufactured by MISUMI. All the explored use cases were part of the IROS 2017 Robotic Grasping and Manipulation Competition [37], included as part of a task board developed by NIST to benchmark the performance of assembly robots.

A. Connectors

In the following we describe the used connectors, USB, D-Sub, and Model-E. The observed difficulty of the insertion increases in that order.

USB. The USB connector is a ubiquitous, widely-used connector and offers a challenging insertion task. Because the connector becomes smoother and therefore easier to insert over time due to wear and tear, we periodically replace the connector. Of the three tested connectors, the USB connector is the easiest.

D-sub. Inserting this connector requires aligning several pins correctly, and is therefore more sensitive than inserting the USB connector. It also requires more downward force due to a tighter fit.

Model-E. This connector is the most difficult of the three tested connectors as it contains several edges and grooves to align and requires significant downward force to successfully insert the part.

B. Experimental Settings

We consider three settings in our experiments in order to evaluate how plausible it is to solve these tasks with more convenient state representations and reward functions and to evaluate the performance of different algorithms changes as the setting is modified. In all settings, the action space is 3-dimensional and the actions are the relative movement of the end effector.

3.2.1 Visual. In this experiment, we evaluate whether the RL algorithms can learn to perform the connector assembly tasks from vision without having access to state information. The state provided to the learned policy is a 32×32 grayscale image, such as shown in Fig. 4. For goal specification, we use a goal image, avoiding the need for state information to compute rewards. The reward is the pixelwise ℓ_1 distance to the given goal image. Being able to learn from such a setup is compelling as it does not require any extra state estimation and many tasks can be specified easily by a goal image.

3.2.2. Sparse. In this experiment, the reward is obtained by directly measuring whether the connection is alive and transmitting:

$$r = \begin{cases} 1, & \text{if insertion signal detected} \\ 0, & \text{else.} \end{cases} \tag{1}$$

This is the exact true reward for the task of connecting a cable, and can be naturally obtained in many manufacturing systems. As state, the robot is given the Cartesian coordinates of the end effector x_t, relative to the assumed goal location, and the vertical force f_z that is acting on the end effector. The state space is 4-dimensional. Providing relative position measurements allows to apply the same policy at a different goal location after recalibration.

3.2.3. Dense. In this experiment, the robot receives a manually shaped reward based on the distance to the target location x^*. We use the reward function

$$r_t = \alpha \cdot \|x_t - x^*\|_1 + \frac{\beta}{(\|x_t - x^*\|_2 + \varepsilon)} + \varphi \cdot f_z, \tag{2}$$

where $0 < \varepsilon \ll 1$. The hyperparameters are set to $\alpha = -100$, $\beta = 0.002$, and $\varphi = -0.1$. When an insertion is indicated through a distance measurement, the sign of the force term flips, so that $\varphi = 0.1$ when the connector is inserted. This rewards the agent for pressing down after an insertion and showed to improve the learning process. The force measurements are calibrated before each rollout to account for measurement bias and to decouple the measurements from the robot pose. The state space is the same as in the sparse setting.

IV. Methods

To solve the connector insertion tasks, we consider and evaluate a variety of RL algorithms.

A. Preliminaries

In a Markov decision process (MDP), an agent at every time step is at state $s_t \in \mathcal{S}$, takes actions $u_t \in \mathcal{U}$, receives a reward $r_t \in \mathbb{R}$, and the state evolves according to environment transition dynamics $p(s_{t+1}|s_t, u_t)$. The goal of reinforcement learning is to choose actions $u_t \sim \pi(u_t|s_t)$ to maximize the expected returns $\mathbb{E}[\sum_{t=0}^{H} \gamma^t r_t]$ where H is the horizon and γ is a discount factor. The policy $\pi(u_t|s_t)$ is often chosen to be an expressive parametric function approximator, such as a neural network, as we use in this work.

B. Efficient Off-Policy Reinforcement Learning

One class of RL methods additionally estimates the expected discounted return after taking action u from state s, the Q-value $Q(s, u)$. Q-values can be recursively defined with the Bellman equation:

$$Q(s_t, u_t) = \mathbb{E}_{s_{t+1}}[r_t + \gamma \max_{u_{t+1}} Q(s_{t+1}, u_{t+1})] \quad (3)$$

and learned from off-policy transitions (s_t, u_t, r_t, s_{t+1}). Because we are interested in sample-efficient real-world learning, we use such RL algorithms that can take advantage of off-policy data.

For control with continuous actions, computing the required maximum in the Bellman equation is difficult. Continuous control algorithms such as deep deterministic policy gradients (DDPG) [38] additionally learn a policy $\pi_\theta(u_t|s_t)$ to approximately choose the maximizing action. In this paper we specifically consider two related reinforcement learning algorithms that lend themselves well to real-world learning as they are sample efficient, stable, and require little hyperparameter tuning.

Twin Delayed Deep Deterministic Policy Gradients (TD3). Like DDPG, TD3 optimizes a deterministic policy [39] but uses two Q-function approximators to reduce value overestimation [40] and delayed policy updates to stabilize training.

Soft Actor Critic (SAC). SAC is an off-policy value-based reinforcement learning method based on the maximum entropy reinforcement learning framework with a stochastic policy [2].

We used the implementation of these RL algorithms publicly available at `rlkit` [41]. For all neural networks in SAC and TD3, we use the same architecture with 2 fully connected layers and 256 nodes per layer.

C. Residual Reinforcement Learning

Instead of randomly exploring from scratch, we can inject prior information into an RL algorithm in order to speed up the training process, as well as to minimize unsafe exploration behavior. In residual RL, actions u_t are chosen by additively combining a fixed policy $\pi_{\mathrm{H}}(s_t)$ with a parametric policy $\pi_\theta(u_t|s_t)$:

$$u_t = \pi_{\mathrm{H}}(s_t) + \pi_\theta(s_t). \quad (4)$$

The parameters θ can be learned using any RL algorithm. In this work, we evaluate both SAC and TD3, explained in the previous section. The residual RL implementation that we use in our experiments is summarized in Algorithm 1.

A simple P-controller serves as the hand-designed controller π_{H} of our experiments. The P-controller operates in Cartesian space and calculates the current control action by

$$\pi_{\mathrm{H}}(s_t) = -k_p \cdot (x_t - x^*), \quad (5)$$

where x^* denotes the commanded goal location. As control gains we use $k_p = [\,1,\, 1,\, 0.3\,]$. This P-controller quickly centers the end effector above the goal position and reaches the goal after about 10 time steps from the reset position, which is located 5cm above the goal.

D. Learning from Demonstrations

Another method to incorporate prior information is to use demonstrations from an expert policy to guide exploration during RL. We first collected demonstrations with a joystick controller. Then, we add a behavior cloning loss while performing RL that pushes the policy towards the demonstrator actions, as previously considered in [30]. Instead of DDPG, the underlying algorithm RL algorithm used is TD3.

V. Experiments

We evaluate our method, which combines residual RL with easy-to-obtain reward signals, on a variety of connector assembly tasks performed on a real robot. In this section, we consider two types of natural rewards that are intuitive for users to provide: an image directly specifying a goal and a binary sparse reward indicating success. For both cases, we report success rates on tasks they solve. We aim to answer the following questions: (1) Can such trained policies provide comparable performance to policies that are trained with densely-shaped rewards? (2) Are these trained policies robust to small variations and noise?

5.1 Vision-based Learning. For the vision-based learning experiments, we use only raw images as observations and for the reward calculation. Sample images that the robot received are shown in Fig. 4. In our experiments, we use 32×32 grayscale images. These scaled-down images were used to simplify the training, and as it can be seen in Fig. 4, the different stages of the insertion are still distinguishable. As reward signal, we use the ℓ_1 distance between the current image and goal image. Although this reward function does not translate to the true distance between the objects, we observed a large difference in reward between a successful and an unsuccessful insertion (e.g. $r = -5$ vs. $r = -200$), which lead to great training results in our experiments. However, since the pixel-based reward is sensitive to small changes in the images, it hat to be ensured that the conditions of the setup did not change between the moment when the goal image was taken and when the training was started.

Algorithm 1 Residual reinforcement learning

Require: policy π_θ, hand-engineered controller π_{H}.
for $n = 0, ..., N-1$ episodes **do**
  Sample initial state $s_0 \sim E$.
  for $t = 0, ..., H-1$ steps **do**
    Get policy action $u_t \sim \pi_\theta(u_t|s_t)$.
    Get action to execute $u'_t = u_t + \pi_{\mathrm{H}}(s_t)$.
    Get next state $s_{t+1} \sim p(\cdot \mid s_t, u'_t)$.
    Store (s_t, u_t, s_{t+1}) into replay buffer $\mathcal{R}$.
    Sample set of transitions $(s, u, s') \sim \mathcal{R}$.
    Optimize θ using RL with transitions.
  end for
end for

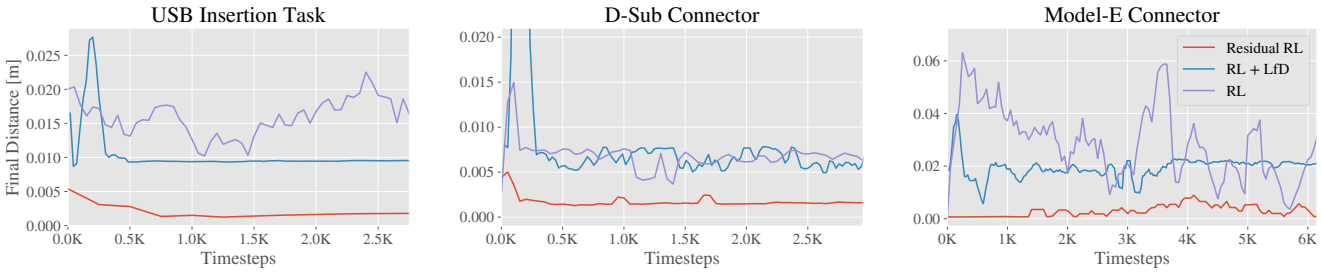

Fig. 5: Resulting final mean distance during the vision-based training. One rollout amounts to 50 time steps. The comparison includes RL, residual RL, and RL with learning from demonstrations. Only residual RL manages to deal with the high-dimensional input and consistently solve all the tasks after the given amount of training. The other methods learn to move downwards, but often get stuck in the beginning of the insertion and fail to recover from unsuccessful attempts.

5.2 Learning from Sparse Rewards. In the sparse reward experiment, we use the binary signal of the connector being electrically connected as the reward signal. This experiment is most applicable to electronic manufacturing settings where the electrical connection between connectors can be directly measured. We only evaluate the sparse reward setting on the USB connector, as it was straightforward to obtain the electrical connection signal.

5.3 Perfect State Information. After evaluating the tasks in the above settings, we further evaluate with full state information with a dense and carefully shaped reward signal, given in Eq. 2, that incorporates distance to the goal and force information. Evaluating in this setting gives us an "oracle" that can be compared to the previous experiments in order to understand how much of a challenge sparse or image rewards pose for various algorithms.

5.4 Robustness. For safe and reliable future usage, it is required that the insertion controller is robust against small measurement or calibration errors that can occur when disassembling and reassembling a mechanical system. To test robustness towards small goal perturbations, we artificially perturb the goal by $\pm 1\,\mathrm{mm}$ in x and in y direction, such that the perturbed goals lie on the edges and corners of a square with side length $2\,\mathrm{mm}$, centered on the exact goal location. This distribution results in 8 perturbed goal locations, on which we evaluate the trained policies. We measure the success rate over 25 policy executions, during which each of the perturbed goal locations is sampled at least 3 times. In addition to translational errors, rotational errors are very likely to occur as well when modifying a robot setup. However, since we calibrate the rotation of the goal location before each experiment, we did not notice rotational errors as the reason for unsuccessful insertions. As our policies only output translational movement and no rotations, a study of the robustness towards rotational errors is left out.

5.5 Exploration Comparison. One advantage of using reinforcement learning is the exploratory behavior that allows the controller to adapt from new experiences unlike a deterministic control law. The two RL algorithms we consider in this paper, SAC and TD3, explore differently. SAC maintains a stochastic policy, and the algorithm also adapts the stochasticity through training. TD3 has a deterministic policy, but uses another noise process (in our case Gaussian) to inject exploratory behavior during training time. We compare the two algorithms, as well as when they are used in conjunction with residual RL, in order to evaluate the effect of the different exploration schemes.

VI. Results

A. Vision-based Learning

The learning curves of the vision-based experiment are presented in Fig. 5, the graphs show the final distance at the end of each 50 step long episode, plotted over the total number of environment steps throughout the training. Our experiments show that a successful and consistent vision-based insertion policy can be learned from relatively few samples using residual RL. In the early stages of the training, the human-engineered controller in residual RL guides the robot towards the point of insertion and due to the RL algorithm's exploration noise, successful insertions are experienced much earlier in the training than with standard RL. During the training on the USB connector, the residual RL policy consistently improved in performance. On the D-Sub and Model-E connector, the residual RL policy became too deterministic and decreased in performance at certain times, but managed to recover and solved the task consistently at the end of the training.

This result suggests that goal-specification through images is a practical way to solve these types of industrial tasks. Although image-based rewards are often very sparse and hard to learn from, in this case the distance between images corresponds to a relatively dense reward signal which is sufficient to distinguish the different stages of the insertion process.

Interestingly, during training with standard RL, the policy would sometimes learn to "hack" the reward signal by moving down in the image in front of or behind the socket. In contrast, the stabilizing human-engineered controller in residual RL provides sufficient horizontal control to prevent this. The initial controller also scaffolds the learning process, by providing a very strong initialization that requires the reinforcement learning algorithm to only learn the final phase of the insertion. This produces substantially better performance in conjunction with vision-based rewards.

TABLE I: We report average success out of 25 policy executions after training is finished for each method. For noisy goals, noise is added in form of $\pm 1\,\mathrm{mm}$ perturbations of the goal location. Residual RL, particularly with SAC, tends to be the best performing method across all three connectors. For the Model-E connector, only residual RL solves the task in the given amount of training time.

D-Sub Connector		Goal	
		Perfect	Noisy
Pure RL	Dense	16%	0%
	Images, SAC	0%	0%
	Images, TD3	12%	12%
RL + LfD	Images	52%	52%
Residual RL	Dense	**100%**	60%
	Images, SAC	**100%**	**64%**
	Images, TD3	52%	52%
Human	P-Controller	**100%**	44%

Model-E Connector		Goal	
		Perfect	Noisy
Pure RL	Dense	0%	0%
	Images, SAC	0%	0%
	Images, TD3	0%	0%
RL + LfD	Images	20%	20%
Residual RL	Dense	**100%**	**76%**
	Images, SAC	**100%**	**76%**
	Images, TD3	0%	0%
Human	P-Controller	52%	24%

TABLE II: Average success rate on the USB insertion task. Residual RL and RL + LfD solve the task consistently. Moreover, residual RL stays robust under ± 1mm noise.

USB Connector		Goal	
		Perfect	Noisy
Pure RL	Dense	28%	20%
	Sparse, SAC	16%	8%
	Sparse, TD3	44%	28%
	Images, SAC	36%	32%
	Images, TD3	28%	28%
RL + LfD	Sparse	**100%**	32%
	Images	88%	60%
Residual RL	Dense	**100%**	**84%**
	Sparse, SAC	88%	**84%**
	Sparse, TD3	**100%**	36%
	Images, SAC	**100%**	80%
	Images, TD3	0%	0%
Human	P-Controller	**100%**	60%

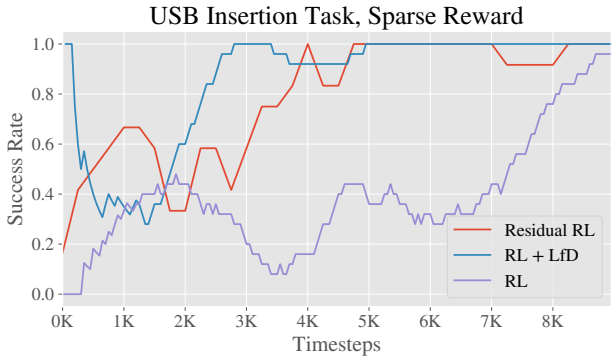

Fig. 6: Learning curves for the training of the USB insertion task with a sparse reward. Success rate is shown; higher is better. One rollout amounts to 50 time steps. Residual RL and RL with learning from demonstrations both solve the task relatively quickly, while RL alone takes about twice as long to solve the task at the same performance.

B. Learning From Sparse Rewards

In this experiment, we compare these methods on the USB insertion task with sparse rewards. The success rates during the training process are reported in Tab. 6. All methods are able to achieve similar performance at the end of the training in the the sparse setting. Learning from demonstrations showed to provide a very good initialization and, after a short decrease in performance due to exploration, managed to reach a continuously succeeding policy in the least amount of training, shortly followed by Residual RL. After twice the amount of time, standard RL also achieved a high success rate during the training. This result shows that we can learn precise industrial insertion tasks from sparse rewards, which can often be obtained much more easily than a dense, shaped reward. In fact, prior work has found that the final policy for sparse rewards can outperform the final policy for dense rewards as it does not suffer from a misspecified objective [42].

C. Perfect State Information

The results of the experiment with perfect state information and dense rewards is shown in Fig. 7. In this case, residual RL outperforms standard RL by a large margin, although the shaped reward function allows standard RL to partially solve the task early in the training. However, the hand-designed shaped reward function makes it harder for the policy to actually perform the full insertion, potentially because the more complex reward landscape provides other competing goals to the policy. The final performance with sparse rewards on the USB insertion task is substantially better.

D. Robustness

In the previous set of experiments, the goal locations were known exactly. In this case, the hand-engineered controller performs well. However, once noise is added to the goal location, the deterministic P-controller struggles. All results of our robustness evaluations are listed in Tab. I and Tab. II. In the presence of a $\pm 1\,\mathrm{mm}$ perturbation, the residual RL controller succeeds more often on the USB and D-Sub tasks, and much more often on the Model-E task, than the other methods. Unlike the P-controller, residual RL consistently solved the Model-E task and overcame goal perturbations in $19/25$ trials. On the USB task, residual RL was able to achieve successful insertions in $21/25$ trials in the presence

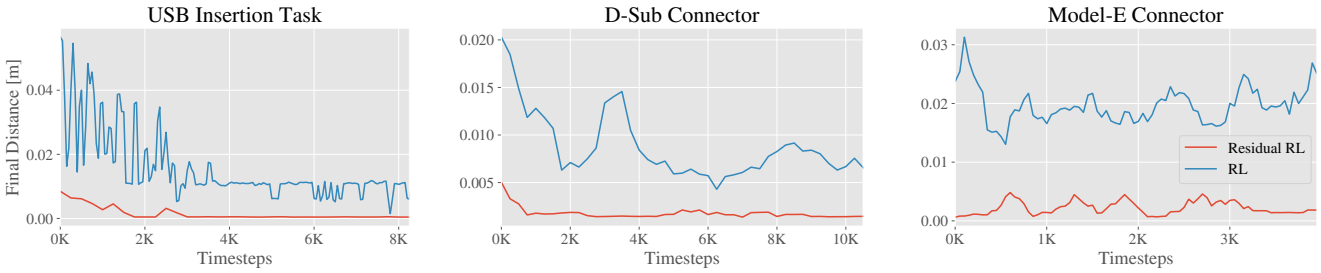

Fig. 7: Plots of the final mean distance to the goal during the state-based training. Final distances greater than $0.01\,\mathrm{m}$ indicate unsuccessful insertions. Here, the residual RL approach performs noticeably better than pure RL and is often able to solve the task during the exploration in the early stages of the training.

of goal perturbations. The agent demonstrably learns small but consistent corrective feedback behaviors in order to move in the right direction during the descent motion, a behavior that is very difficult to specify manually. This behavior illustrates the strength of residual RL. Since the human controller already specifies the general trajectory of the optimal policy, environment samples are only required to learn this corrective feedback behavior.

E. Exploration Comparison

All experiments were also performed using TD3 instead of SAC. Without added goal perturbations, SAC and TD3 perform comparably. However, TD3 is often substantially less robust. These results are likely explained by the exploration strategy of the two algorithms. TD3 has a deterministic policy and fixed noise during training, so once it observes some high-reward states, it quickly learns to repeat that trajectory. SAC adapts the noise to the correct scale, helping SAC stay robust to small perturbations, and because SAC learns the value function for a stochastic policy, it is able to handle some degree of additive noise effectively. In combination with residual RL, TD3 only managed to solve the sparse reward settings and did not train well in the vision-based setting. Here, we found that the outputted action of TD3 approaches the extreme values at the edge of the allowed action space, while SAC executed less extreme actions, which showed to produce better training results.

VII. Conclusion

In this paper, we studied deep reinforcement learning in a practical setting, and demonstrated that deep RL can solve complex industrial assembly tasks with low tolerances. We showed that we can learn insertion policies with raw image observations with either binary outcome-based rewards, or rewards based on goal images. We conducted a series of experiments for various connector type assemblies, and demonstrated the feasibility of our method under challenging conditions, such as noisy goal specification and complex connector geometries. Reinforcement learning algorithms that can automatically learn complex assembly tasks with easy-to-specify reward functions have the potential to automate a wide range of assembly tasks, making this technology a promising direction forward for flexible and capable robotic manipulators.

There remain significant challenges for applying these techniques in more complex environments. One practical direction for future work is focusing on multi-stage assembly tasks through vision. This would pose a challenge to the goal-based policies as the background would be visually more complex. Moreover, multi-step tasks involve adapting to previous mistakes or inaccuracies, which could be difficult but should be able to be handled by RL. Extending the presented approach to multi-stage assembly tasks will pave the road to a higher robot autonomy in flexible manufacturing.

VIII. Acknowledgment

This work was supported by the Siemens Corporation, the Office of Naval Research under a Young Investigator Program Award, and Berkeley DeepDrive.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," in *NIPS Workshop on Deep Learning*, 2013, pp. 1–9.

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *ICML*, 2018.

[3] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *AAAI*, 2018.

[4] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, 1999.

[5] I. Popov, N. Heess, T. Lillicrap *et al.*, "Data-efficient Deep Reinforcement Learning for Dexterous Manipulation," *CoRR*, vol. abs/1704.0, 2017.

[6] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters, "Active Reward Learning," in *RSS*, 2014.

[7] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual Reinforcement Learning with Imagined Goals," in *NeurIPS*, 2018.

[8] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. Aparicio Ojea, E. Solowjow, and S. Levine, "Residual Reinforcement Learning for Robot Control," in *ICRA*, 2019.

[9] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual Policy Learning," in *ArXiv 1812.06298*, 2018.

[10] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *NIPS*, 1989, pp. 305–313.

[11] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," in *Robotics and Autonomous Systems*, vol. 47, no. 2-3, 2004, pp. 79–91.

[12] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning," *RSS*, pp. 57–64.

[13] L. Pinto and A. Gupta, "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours," *ICRA*, 2016.

[14] A. Giusti, J. J. Guzzi, D. C. Cirean *et al.*, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," in *IEEE Robotics and Automation Letters.*, vol. 1, no. 2, 2015, pp. 2377–3766.

[15] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-Shot Visual Imitation," in *ICLR*, 2018.

[16] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *ICML*, 2011, pp. 465–472.

[17] J. Peters, K. Mülling, and Y. Altün, "Relative Entropy Policy Search," in *AAAI*, 2010, pp. 1607–1612.

[18] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[19] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research (JMLR)*, vol. 17, no. 1, pp. 1334–1373, 2016.

[20] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous Deep Q-Learning with Model-based Acceleration," in *ICML*, 2016.

[21] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining Model-Based and Model-Free Updates for Trajectory-Centric Reinforcement Learning," in *ICML*, 2017.

[22] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine, "SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning," in *ICML*, aug 2019.

[23] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost," in *ICRA*, oct 2018.

[24] D. Kalashnikov, A. Irpan, P. Pastor *et al.*, "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation," in *CoRL*, 2018.

[25] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning Robotic Assembly from CAD," in *ICRA*, 2018.

[26] J. Kober and J. Peter, "Policy search for motor primitives in robotics," in *NIPS*, vol. 97, 2008, pp. 83–117.

[27] C. Finn, S. Levine, and P. Abbeel, "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization," in *ICML*, 2016.

[28] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning." in *AAAI*, 2008, pp. 1433–1438.

[29] T. Hester, M. Vecerik, O. Pietquin *et al.*, "Learning from Demonstrations for Real World Reinforcement Learning," in *AAAI*, 2018.

[30] A. Nair, B. Mcgrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming Exploration in Reinforcement Learning with Demonstrations," in *ICRA*, 2018.

[31] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *RSS*, 2018.

[32] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-End Robotic Reinforcement Learning without Reward Engineering," in *RSS*, 2019.

[33] R. Li, R. Platt, W. Yuan, A. Ten Pas, N. Roscup, M. A. Srinivasan, and E. Adelson, "Localization and Manipulation of Small Parts Using GelSight Tactile Sensing," in *IROS*, 2014.

[34] A. Tamar, G. Thomas, T. Zhang, S. Levine, and P. Abbeel, "Learning from the hindsight plan episodic mpc improvement," in *ICRA*, 2017, pp. 336–343.

[35] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *IROS*, 2017, pp. 819–825.

[36] J. Luo, E. Solowjow, C. Wen, J. Aparicio Ojea, A. Agogino, A. Tamar, and P. Abbeel, "Reinforcement learning on variable impedance controller for high-precision robotic assembly," in *ICRA*, 2019.

[37] J. Falco, Y. Sun, and M. Roa, "Robotic grasping and manipulation competition: Competitor feedback and lessons learned," in *Robotic Grasping and Manipulation*, Y. Sun and J. Falco, Eds. Cham: Springer International Publishing, 2018, pp. 180–189.

[38] 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 *ICLR*, 2016.

[39] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *ICML*, 2018.

[40] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," in *AAAI*, 2016.

[41] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal Difference Models: Model-Free Deep RL For Model-Based Control," in *ICLR*, 2018.

[42] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. Mcgrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight Experience Replay," in *NIPS*, 2017.