BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Oct 2024

Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning

Jianlan Luo[1], Charles Xu[1], Jeffrey Wu[1] and Sergey Levine[1]
[1]Department of Electrical Engineering and Computer Sciences, UC Berkeley

Reinforcement learning (RL) holds great promise for enabling autonomous acquisition of complex robotic manipulation skills, but realizing this potential in real-world settings has been challenging. We present a human-in-the-loop vision-based RL system that demonstrates impressive performance on a diverse set of dexterous manipulation tasks, including dynamic manipulation, precision assembly, and dual-arm coordination. Our approach integrates demonstrations and human corrections, efficient RL algorithms, and other system-level design choices to learn policies that achieve near-perfect success rates and fast cycle times within just 1 to 2.5 hours of training. We show that our method significantly outperforms imitation learning baselines and prior RL approaches, with an average 2x improvement in success rate and 1.8x faster execution. Through extensive experiments and analysis, we provide insights into the effectiveness of our approach, demonstrating how it learns robust, adaptive policies for both reactive and predictive control strategies. Our results suggest that RL can indeed learn a wide range of complex vision-based manipulation policies directly in the real world within practical training times. We hope this work will inspire a new generation of learned robotic manipulation techniques, benefiting both industrial applications and research advancements. Videos and code are available at our project website https://hil-serl.github.io/.

1. Introduction

Manipulation is one of the foundational problems in robotics, and achieving human-level performance on dynamic, dexterous manipulation tasks is a longstanding pursuit in the field (Cui and Trinkle, 2021). Reinforcement learning (RL) holds the promise of enabling autonomous acquisition of complex and dexterous robotic skills. By learning through trial and error, an effective RL method should in principle be able to acquire highly proficient skills that are tailored to the particular physical characteristics of the deployment task. This could result in performance that not only exceeds that of hand-designed controllers but also surpasses human teleoperation. However, realizing this promise in real-world settings has been challenging due to issues with sample complexity, assumptions (e.g., accurate reward functions), and optimization stability. RL methods have been effective for training in simulation (Hwangbo et al., 2019; Lee et al., 2020; Chen et al., 2023; Loquercio et al., 2021), and for training on existing large real-world datasets with the aim of broad generalization (Kalashnikov et al., 2018; 2021). They have also been used with hand-designed features or representations for narrowly tailored tasks (Theodorou et al., 2010; Chebotar et al., 2016). However, developing general-purpose vision-based methods that can efficiently acquire physically complex skills, with proficiency exceeding imitation learning and hand-designed controllers, has been comparatively difficult. We believe that making fundamental progress on this front can unlock new opportunities, which will then enable the development of truly performant robotic manipulation policies.

In this paper, we develop a reinforcement learning (RL) system for vision-based manipulation that can acquire a wide range of precise and dexterous robotic skills. Our system, named Human-in-the-Loop Sample-Efficient Robotic Reinforcement Learning (HIL-SERL), addresses the previously mentioned challenges by integrating a number of components that enable fast and highly performant vision-based RL in the real world.

To address the optimization stability issue, we employ a pretrained visual backbone for policy learning. To handle the sample complexity issue, we utilize a sample-efficient off-policy RL algorithm based on RLPD (Ball et al., 2023) that also incorporates human demonstrations and corrections. Additionally, a

Correspondence: Jianlan Luo(jianlanluo@berkeley.edu), Charles Xu(xuc@berkeley.edu).

Figure 1: Overview of experimental tasks. A subset of tasks considered in this paper, they include whipping out a Jenga block from its tower, flipping an object in a pan, assembling complex devices such as a timing belt, a dashboard, a motherboard, and an IKEA shelf.

well-designed low-level controller is included to ensure safety during policy training. During training, the system queries a human operator for potential corrections, which are then used to update the policy in an off-policy manner. We found this human-in-the-loop correction procedure is crucial for enabling the policy to learn from its mistakes and improve performance, particularly for challenging tasks considered in this paper, which are hard to learn from scratch.

As shown in Fig. 1, the tasks our system solves include dynamically flipping an object in a pan, whipping out a Jenga block from a tower, handing over objects between two arms, and assembling complex devices such as a computer motherboard, an IKEA shelf, a car dashboard, or a timing belt, using either one or two robotic arms. These tasks present significant challenges in terms of complex and intricate dynamics, high-dimensional state and action spaces, long horizons, or combinations thereof. Some of these skills were previously considered infeasible to train with RL directly in real-world settings, such as many of the dual-arm manipulation tasks, or nearly insurmountable with current robotics methods, like timing belt assembly or Jenga whipping. And they require different types of control strategies, such as reactive

closed-loop control for precise manipulation tasks or delicate open-loop behaviors that are otherwise very difficult to prescribe, e.g., Jenga whipping. However, perhaps the most surprising finding is that our system can train RL policies to achieve near-perfect success rates and super-human cycle times on almost all of the tasks with only one hour to 2.5 hours of training time in the real world. Our trained RL policies greatly outperform imitation learning methods that are trained on the same amount of human data, e.g., same number of episodes of demonstrations or corrections, on average 101% improvement in terms of success rate and 1.8x faster in cycle time. The result is significant because it demonstrates that RL can indeed learn a wide range of complex vision-based manipulation policies directly in the real world within practical training times, which was previously considered infeasible with earlier methods. Moreover, RL does so with a superhuman level of performance that greatly exceeds that of imitation learning and hand-designed controllers.

To assess the effectiveness of our system, we compare it against several state-of-the-art RL methods and conduct ablation studies to understand the contribution of each component. The results demonstrate that our system not only outperforms the relevant baselines but also highlights that the impressive empirical results are indeed due to the careful integration of these components. Additionally, we provide a comprehensive analysis of the empirical results, offering insights into the effectiveness of RL-based manipulation. This analysis explores why RL achieves a near-perfect success rate, and further examines the flexibility of RL policies to serve as a general-purpose vision-based policy for acquiring distinct types of control strategies.

In summary, our contributions demonstrate that with the appropriate system-level design choices, RL can effectively solve a wide range of dexterous and complex vision-based manipulation tasks in the real world. Notably, our system is, to the best of our knowledge, the first to achieve dual-arm coordination with image inputs using RL in real-world settings, as well as tasks like whipping out a Jenga block and assembling a timing belt. We also provide a comprehensive analysis of the empirical success of RL-based manipulation, offering insights into the effectiveness of RL-based manipulation. This analysis shapes our understanding of why RL succeeds in these complex tasks and suggests potential directions for further extending RL-based manipulation to even more challenging scenarios.

With the results presented in this paper, we hope this work will serve as a stepping stone for future learning-based robotic manipulation research, and in the long term, could enable robust deployable robotic manipulation skills capable of adapting to diverse environments and tasks, bringing us closer to the goal of general-purpose robotic manipulation.

2. Related Work

The proposed system uses RL to solve dexterous manipulation tasks, thus we survey related works on real-world robotic RL methods and systems, as well as other approaches that address similar dexterous manipulation tasks.

Algorithms and systems for real-world RL Real-world robotic reinforcement learning (RL) requires algorithms that are sample-efficient in handling high-dimensional inputs such as onboard perception and supporting easy specification of rewards and resets. Several algorithms have demonstrated the ability to learn efficiently directly in the real world (Riedmiller et al., 2009; Levine et al., 2016; Luo et al., 2021; Yang et al., 2020; Zhan et al., 2021; Tebbe et al., 2021; Popov et al., 2017; Luo et al., 2019; Zhao et al., 2022; Hu et al., 2024b; Johannink et al., 2019; Hu et al., 2024a; Rajeswaran et al., 2018; Schoettler et al., 2020; Luo et al., 2024a). These include variants of off-policy RL (Kostrikov et al., 2023; Hu et al., 2024b; Luo et al., 2023), model-based RL (Hester and Stone, 2013; Wu et al., 2022; Nagabandi et al., 2019; Rafailov et al., 2021; Luo et al., 2018), and on-policy RL (Zhu et al., 2019). Despite progress, these often require long training times. Our system achieves super-human performance on complex tasks with shorter training times. Other works have been researching on inferring rewards from raw visual observation through success classifiers (Fu et al., 2018; Li et al., 2021), foundation-model-based rewards (Du et al., 2023; Mahmoudieh et al., 2022; Fan et al., 2022), and rewards from videos (Ma et al., 2023b;a). Additionally, to enable autonomous training, there have been a number of algorithmic advances in reset-free learning (Gupta et al., 2021; Sharma et al., 2021; Zhu et al.,

2020; Xie et al., 2022; Sharma et al., 2023) that require minimal human interventions during training. While we do not introduce new algorithms in these areas, our framework effectively integrates existing methods. As detailed in the methods section, using binary classifier-based rewards with demonstrations is effective for the complex tasks in this paper.

One of the most relevant works to our research is SERL (Luo et al., 2024a), which also presents a system for training reinforcement learning (RL) policies for manipulation tasks. Our approach differs from SERL in that: we incorporate both human demonstrations and corrections to train RL policies, whereas SERL relies solely on human demonstrations. While this might appear to be a minor distinction, our results indicate that integrating corrections is crucial for enabling the policy to learn from its mistakes and improve performance, especially for tasks that are challenging for the agent to learn from scratch. Additionally, SERL focuses on simpler tasks with relatively short horizons and does not address dual-arm coordination or dynamic manipulation. Our unique contribution is demonstrating that our approach can effectively learn general-purpose vision-based manipulation policies across a wide range of tasks with varying physical characteristics, setting our system fundamentally different from prior work on SERL.

Dexterous robotic manipulation For some of the tasks considered in this paper, prior works have explored alternative approaches. In insertion tasks, prior works have used model-based approaches (Tang et al., 2016; Jin et al., 2021) and end-effector tooling mechanisms with passive compliance (Morgan* et al., 2021; Su et al., 2022). These methods often rely on state-based models without perception or require task-specific development, limiting robustness and adaptability. Another approach involves using visual servoing in a multi-stage pipeline to align the robotic arm with the target, followed by search primitives for insertions (Spector et al., 2022; Chang et al., 2024; Song et al., 2015). They also face challenges with feature reliability and alignment precision. In contrast, our method employs a much tighter perception-action loop. It learns task-relevant visual features and visuomotor policies in a closed-loop manner, crucial for many of the reactive high-precision tasks. The learned policy can be viewed as an instance of output feedback control from the controls perspective (Astrom and Murray, 2008).

There are also a number of works on tackling the dynamic manipulation tasks (Mason and Lynch, 1993) considered in this paper. Kormushev et al. (2010) utilized a motion capture system and dynamic motion primitives (Ijspeert et al., 2013) to learn flipping an object in the pan. However, our system directly uses pixel inputs, which alleviates the need for precise motion capture systems while achieving significantly higher success rates. Fazeli et al. (2019) proposed a learning method to push out Jenga block from its tower in a quasi-dynamic manner. Our approach, however, employs a whip to dynamically remove the Jenga block, presenting a more challenging task that requires a much more sophisticated control policy. Additionally, while there are studies on flexible object manipulation, such as cable routing (Luo et al., 2024b; Jin et al., 2019), tracing, or untangling (Viswanath et al., 2023; Shivakumar et al., 2023; Viswanath et al., 2022), the timing belt assembly task in our paper demands reactive yet precise coordination between two arms to dynamically adjust both the tensioner and the timing belt. This task is fundamentally different and more challenging than previous works on cable manipulation.

3. Human-in-the-Loop Reinforcement Learning System

In this section, we provide a detailed description of the methods used in the paper. For an animated movie summarizing the presented methods, please refer to the accommodating video.

3.1. Preliminaries and Problem Statement

Robotic reinforcement learning tasks can be defined via an MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \rho, \mathcal{P}, r, \gamma\}$, where $\mathbf{s} \in \mathcal{S}$ is the state observation (e.g., an image in combination with the robot's proprioceptive state information), $\mathbf{a} \in \mathcal{A}$ is the action (e.g., the desired end-effector twist), $\rho(\mathbf{s}_0)$ is a distribution over initial states, $\mathcal{P}$ is the unknown and potentially stochastic transition probabilities that depend on the system dynamics, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which encodes the task. An optimal policy π is one that maximizes the cumulative expected value of the reward, i.e., $E[\sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$, where the expectation is taken with

respect to the initial state distribution, transition probabilities, and policy π. In practice, the policy $\pi(\mathbf{a}|\mathbf{s})$ is usually modeled as a Gaussian distribution parameterized by a neural network.

To implement reinforcement learning algorithms for robotic tasks, we must carefully select appropriate state observation spaces $\mathcal{S}$ and action spaces $\mathcal{A}$. This involves choosing the right combination of cameras, proprioceptive states, and corresponding robot low-level controllers. For all our tasks, we employ a sparse reward function. This function makes a binary decision on whether a task is successful or not using a trained classifier. In this setup, the optimization objective $E[\sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$ aims to maximize the probability of success for each trajectory. Ideally, at convergence, the policy should succeed at every attempt.

Specifically, the core underlying RL algorithm that we build on is RLPD (Ball et al., 2023), which we chose for its sample efficiency and ability to incorporate prior data. At each training step, RLPD samples equally between prior data and on-policy data to form a training batch (Song et al., 2023). It then updates the parameters of a parametric Q-function $Q_\phi(\mathbf{s}, \mathbf{a})$ and the policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ according to the gradient of their respective loss functions:

$$\mathcal{L}_Q(\phi) = E_{\mathbf{s},\mathbf{a},\mathbf{s}'}\left[\left(Q_\phi(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma E_{\mathbf{a}'\sim\pi_\theta}[Q_{\bar{\phi}}(\mathbf{s}',\mathbf{a}')]\right)\right)^2\right] \tag{1}$$

$$\mathcal{L}_\pi(\theta) = -E_{\mathbf{s}}\left[E_{\mathbf{a}\sim\pi_\theta(\mathbf{a})}[Q_\phi(\mathbf{s},\mathbf{a})] + \alpha\mathcal{H}(\pi_\theta(\cdot|\mathbf{s})\right], \tag{2}$$

where $Q_{\bar{\phi}}$ is a *target network* (Mnih et al., 2013), and the actor loss uses entropy regularization with an adaptively adjusted weight α (Haarnoja et al., 2018).

3.2. System Overview

Our system is composed of three major components: the actor process, the learner process, and the replay buffer residing inside the learner process, all operating in a distributed fashion, as illustrated in Fig. 2. The actor process interacts with the environment by executing the current policy on the robot and sends the data back to the replay buffer. The environment is designed to be modular, allowing for flexible configuration of various devices. This includes support for multiple cameras, integration of input devices like SpaceMouse for teleoperation, and the ability to control a variable number of robot arms with different type of controllers. A implemented reward function is required to assess the success of a task, which is trained offline using human demonstrations. Inside the actor process, a human can intervene the robot by using a SpaceMouse, which will then take over the control of the robot from the RL policy. We employ two replay buffers, one to store offline human demonstrations, called the demo buffer, usually on the range of 20-30; the other one for storing the on-policy data, called the RL buffer.

The learner process samples data equally from the demo and RL replay buffers, optimizes the policy using RLPD, and periodically sends the updated policy to the actor process. In the remainder of this section, we will provide details about the design choices we made for each component.

3.3. System Design Choices

The sample efficiency of the proposed system is crucial, as each minute of training to acquire data incurs a cost. Therefore, the training time must remain within a practical range, particularly when handling high-dimensional inputs like images. Additionally, the downstream robotic system must accommodate the RL policy so to ensure a smooth and efficient learning process. For example, the actual low-level robot controller would require great care, particularly for most of the precise contact-rich tasks where the robot physically interacts with objects in the environment. Not only does this controller need to be accurate, but it must also be safe enough that the RL algorithm can explore with random actions during training. Thus, to develop such a system capable of performing sample-efficient policy learning in the real world, we made following design choices.

Pretrained Vision Backbones To facilitate the efficiency of the training process, we utilize pretrained vision backbones to process image data. While this approach is now a common practice in computer vision for the purpose of robustness and generalization (Radford et al., 2021; Dosovitskiy et al., 2021; Kolesnikov

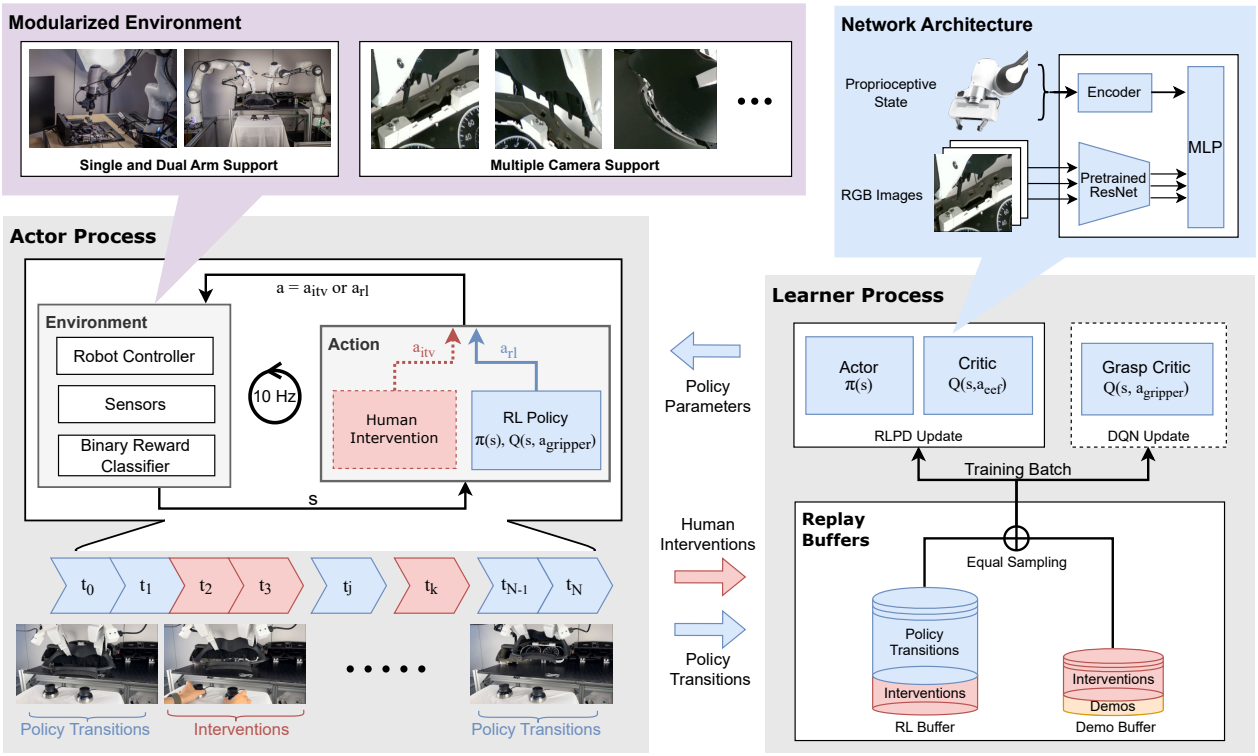

Figure 2: Overview of HIL-SERL. This figure illustrates the architecture of HIL-SERL, which comprises three primary components: the actor process, the learner process, and replay buffers. These components communicate asynchronously to facilitate efficient data flow. The actor process receives updated policy parameters from the learner process, interacts with the environment, and sends collected interaction data to the replay buffers. The environment is modular, supporting various external devices and multiple robotic arms. A human operator can intervene via teleoperation tools, such as a SpaceMouse. The learner process samples data evenly from two replay buffers and updates the policy using RLPD. When gripper control is required, a grasp critic is additionally trained with DQN.

et al., 2020); in RL, this treatment offers additional benefits, such as optimization stability and exploration efficiency (Yang and Wang, 2019; Du et al., 2020), making this approach particularly advantageous for real-world robotic RL training. Our neural network architecture, illustrated in Fig. 2, processes multiple images from cameras using the same pretrained vision backbone. Specifically, we utilize a ResNet-10 model (He et al., 2015), pretrained on ImageNet (Deng et al., 2009), to generate output embeddings. These embeddings are then concatenated and further integrated with processed proprioceptive information, facilitating a more efficient and effective learning process.

Reward Function One crucial aspect of a reinforcement learning system is the reward function, which is used to guide the learning process and determine the quality of the policy. While there are prior works on utilizing reward shaping to accelerate the learning process (Ng et al., 1999; Florensa et al., 2018; 2017), this procedure tends to be task-specific and time-consuming to design. In some complex tasks, it's simply not feasible to perform such reward shaping. We found that using a sparse reward function, alongside human demonstrations and corrections, offers a straightforward and effective setup for a variety of tasks. Specifically, we collect offline data and train a binary classifier for each task, which only grants a positive reward upon task completion and zero otherwise.

Downstream Robotic System To accommodate the policy learning process, we made a few particularly important design choices to the downstream robotic system. To facilitate spatial generalization, we represent the robot's proprioceptive state in a relative coordinate system, allowing for an ego-centric formulation. Essentially, at the beginning of each training episode, the pose of the robot's end-effector was randomized uniformly within a pre-defined area in the workspace. The robot's proprioceptive information is expressed

with respect to the frame of the end-effector's initial pose; the action output from the policy is relative to the current end-effector frame. This procedure simulates physically moving the target when viewed relatively from the frame attached to the end-effector. As a result, the policy can succeed even if the object moves or, as in some of our experiments, is perturbed in the middle of the episode. For tasks involving dealing with contact, we use an impedance controller with reference limiting in the real-time layer to ensure safety as in (Luo et al., 2024a). For dynamic tasks, we directly command feedforward wrenches in the end-effector frame to accelerate the robot arm, while it doesn't perform closed-loop control around acceleration, we found this simple open-loop control to be sufficient for considered tasks. For details regarding the observation representation as well as the controller design, please refer to the supplementary material.

Gripper Control For tasks involving the control of grippers, we employ a separate critic network to evaluate discrete grasping actions. Although this approach might initially seem like an additional overhead or somewhat unconventional, it has proven to be highly effective in practice, particularly when combined with human demonstrations and corrections. The discrete nature of gripper actions in our tasks makes approximating them with continuous distributions more challenging, particularly in the complex tasks considered in this paper. By using discrete actions, we simplify the training process and improve the overall effectiveness of the reinforcement learning system. Specifically, we solve two separate MDPs in these tasks, $\mathcal{M}_1 = \{\mathcal{S}, \mathcal{A}_1, \rho_1, \mathcal{P}_1, r, \gamma\}$ and $\mathcal{M}_2 = \{\mathcal{S}, \mathcal{A}_2, \rho_2, \mathcal{P}_2, r, \gamma\}$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are the continuous and discrete action spaces, respectively. They both take in the same state observations from the environment such as images, proprioception, gripper status and so forth. The discrete action space $\mathcal{A}_2$ consists of all possible discrete actions. For a single gripper, these actions are "open", "close", and "stay". If two grippers are involved, the action space expands to $3^2 = 9$ combinations, accounting for all possible actions each gripper can take. The critic update for $\mathcal{M}_2$ follows standard DQN practice (Mnih et al., 2013) with an additional target network to stabilize training as following:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'}\left[\left(r + \gamma Q_{\theta'}(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_\theta(\mathbf{s}', \mathbf{a}')) - Q_\theta(\mathbf{s}, \mathbf{a})\right)^2\right], \tag{3}$$

where θ' is the target network, which can be obtained by Polyak averaging with current network parameters (van Hasselt et al., 2015). At training or inference time, we first query the continuous actions from the policy in $\mathcal{M}_1$, and then query the discrete actions from the critic in $\mathcal{M}_2$ by taking the argmax over the critic's output; we then apply the concatenated actions to the robot.

3.4. Human-in-the-Loop Reinforcement Learning

With the system-level design choices in place, we now describe the human-in-the-loop procedure that we use to accelerate the learning process. It is well established from the RL theory literature that the sample complexity of learning an optimal policy is closely tied to the cardinality of the state and action spaces as well as the task horizon (Jin et al., 2018; 2020; Azar et al., 2012; Kearns and Singh, 1998), assuming an appropriate exploration policy. These factors collectively serve as proxies for the "upper bound" on the complexity of tasks that can be feasibly solved. Specifically, increases in the size of the state/action spaces, task horizon, or their combinations lead to a proportional rise in the number of samples required to learn an optimal policy; eventually crossing a threshold where real-world training of RL policies becomes impractical.

To tackle this challenge in real-world robotics RL training, we incorporate human-in-the-loop feedback to guide the learning process to help the policy explore more efficiently. Specifically, a human operator supervises the robot during training and provides corrective actions when necessary, as illustrated in Fig. 2. For an autonomous rollout trajectory from time step t_0 to t_N, a human can intervene at any time step t_i where $t_0 \leq t_i < t_N$. During an intervention, the human takes control of the robot for up to N steps. Multiple interventions can occur within a single trajectory, as illustrated by the red segments in Fig. 2. When a human intervenes, their action a_{itv} is applied to the robot instead of the policy's action a_{RL}. We store the intervention data in both the demonstration and RL data buffers. However, we add the policy's transitions (i.e., the states and actions before and after the intervention) only to the RL buffer. This approach has proven effective in enhancing policy training efficiency.

This intervention is crucial in scenarios where the policy leads the robot to an unrecoverable or undesirable state, or when it becomes stuck in a local optimum that would otherwise require a significant amount of time to overcome without human assistance. This procedure is similar to that of HG-DAgger (Kelly et al., 2018), where a human takes over the control of the robot to collect data when the policy is performing poorly; but our approach uses these data to optimize the policy with reinforcement learning rather than supervised learning, similar to Luo et al. (2023). In our setup, the human operator engages with a SpaceMouse 3D mouse to provide corrective actions to the robot.

In the beginning of the training process, the human intervenes more frequently to provide corrective actions, gradually decreasing the frequency as the policy improves. In our experience, we note that the policy improves faster when the human operator issues specific corrections while letting the robot explore on its own otherwise.

3.5. Training Process

To articulate the training process of our system and assist readers in reproducing our results, we provide a detailed walkthrough of the steps involved in each of our experiments.

First, we select cameras that are most suitable for the task. Wrist cameras are particularly useful for facilitating the spatial generalization of the learned policy due to the ego-centric views they provide. However, if wrist cameras alone cannot provide a full view of the environment, we also place several side cameras. For all cameras, we perform image cropping to focus on the area of interest and resize the images to 128x128 for the neural network to process.

Next, we collect data to train the reward classifier, which is a crucial step in defining the reward function that guides the learning process. Typically, we gather 200 positive data points and 1000 negative data points by tele-operating the robot to perform the task. This is approximately equivalent to 10 human trajectories, assuming each trajectory takes about 10 seconds. Using our data collection pipeline, as detailed in the supplementary code, it usually takes around 5 minutes to collect these data points. Additionally, we may collect extra data to address any false negative and false positive issues with the reward classifier. The trained reward classifier generally achieves an accuracy of greater than 95% in the evaluation data set.

We then collect 20-30 trajectories of human demonstrations solving the tasks and use them to initialize the offline demo replay buffer. For each task, we either script a robot reset motion or let the human operator manually reset the task at the beginning of each trajectory, such as the USB pick-insertion task. Finally, we start the policy training process. During this phase, human interventions may be provided to the policy if necessary, until the policy converges. It's also important to note that we should avoid persistently providing long sparse interventions that lead to task successes. Such an intervention strategy will cause the overestimation of the value function, particularly in the early stages of the training process; which can result in unstable training dynamics.

4. Experiment Results

In this section, we discuss our experiments. We first present the experimental setup and the results. We then discuss these results and their implications.

4.1. Overview of Experiments

We conduct experiments across seven diverse tasks covering a range of distinct characteristics, as illustrated in Fig. 3. These tasks encompass a range of manipulation challenges, including dynamic object manipulation (e.g., flipping an object in a pan), precise and delicate manipulation (e.g., inserting an SSD into its matching slot), combined dynamic and precise manipulation (e.g., inserting a component while the target is moving), flexible object manipulation (e.g., assembling a timing belt) and multi-stage tasks with multiple sub-tasks (e.g., assembling an IKEA shelf). We solve these tasks by utilizing either a single robot arm or a dual-arm setup, together with various combinations of observations and actions.

The observation space can include images from wrist-mounted and side cameras, end-effector poses, twists, forces/torques, and the current gripper status of both arms. For dynamic tasks, the action space

directly commands feedforward wrenches in the end-effector frame, which can be roughly thought of as desired accelerations.

For other tasks, the action space can include each arm's 6D Cartesian twist target for the downstream impedance controller, and discrete actions for controlling one or two grippers.

For all tasks, unless otherwise noted, we trained a binary classifier as reward detector, it takes images from wrist and/or side cameras as inputs, and predicts whether the current state is a success or failure completing the current task. To train such classifiers, we collect both positive and negative demonstrations from human operators, we also collect additional potential false positive or false negative examples if necessary. We include details of the training process for each task in the supplementary material. For tasks involving grasping, we also include a small negative penalty for gripper actions to discourage the policy from operating its grippers unnecessarily. Each task also uses either a scripted robot motion or manually human reset to randomize the initial state of the task. Details for setup of each task and policy training can be found in the supplementary material. In the remainder of this section, we will first describe each task in detail, and present relevant results as well as comparisons to other state-of-the-art methods.

4.2. Description of Tasks

In this subsection, we will present descriptions of the tasks in our experiments. We pick our tasks to cover broad range of manipulation challenges, including contact-rich dynamics, dual-arm coordination, flexible object handling, and dynamic manipulation. Here we organize the tasks in a way that similar challenges are presented together. We first present two tasks that require precise manipulation in a contact-rich setting, followed by three tasks that require dual-arm coordination to solve hard tasks including flexible object manipulation. We then proceed to two tasks that require dynamic manipulation. An illustration of each task can be found in Fig. 3.

Motherboard Assembly The motherboard assembly task includes four subtasks: inserting a RAM card into its matching slot, assembling a PCI-E SSD into the motherboard, picking up a free-floating USB cable and inserting it into the slot, and securing the USB cable into a tight-fitting clip.

RAM Insertion In this task, the robot is supposed to insert a RAM card into the matching slot. The process involves two main steps: it first needs to align the RAM card with the narrow openings on both sides of the slot, then proceed with delicate downward motion with appropriate force to insert the RAM card into the slot. The task is considered successful if the RAM card is fully inserted into the slot without triggering the locking mechanism, allowing for easy reset. If desired, an additional downward force can be applied after executing the trained policy to lock the RAM card in place. This task is challenging because applying slightly excessive force can cause the RAM card to tilt within the gripper, leading to failure, while insufficient force may result in the RAM card not being properly inserted into the slot. The RAM card is assumed to be pre-grasped by the robot, though we also periodically place it back to a fixture and regrasp it to introduce grasping variations.

SSD Assembly In this task, the robot is supposed to insert one side of the SSD into its matching slot and then place the other side onto the fixture residing in the motherboard. The task is considered successful if both sides of the SSD are properly mated into their counterparts. This task requires a gentle but precise insertion strategy initially to avoid damaging the contact pins, followed by another precise motion to align the other side with the supporting fixture. The SSD is assumed to be pre-grasped by the robot, though we also periodically place it back to a fixture and regrasp it to introduce grasping variations.

USB Connector Grasp-Insertion In this task, a USB cable is freely placed on a table, and the robot is supposed to grasp the USB connector part, insert it into the corresponding slot and release the gripper. This task is considered successful if the USB connector is fully inserted into the slot and the gripper is released. The difficulty lies in the variability in the initial placement of the USB cable, as does the uncertainty in the

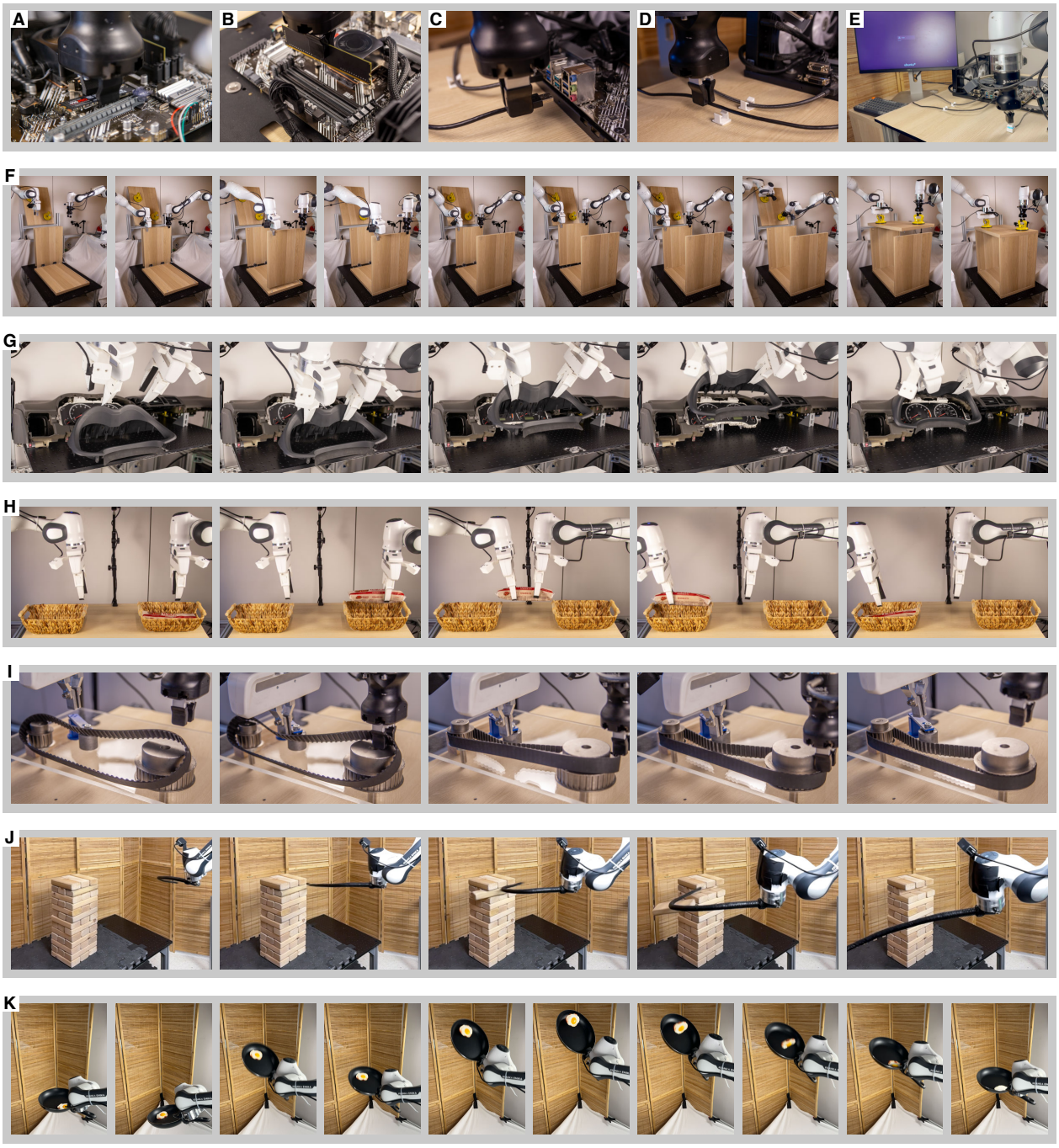

Figure 3: Illustrations of the tasks in our experiments. (A)-(E) A sequence of motherboard assembly tasks: SSD installation, RAM insertion, USB cable grasping and insertion into a slot and a clip, and booting up the computer to ensure motherboard functionality. **(F)** A manipulation sequence to assemble an IKEA furniture: the robot first assembles two side panels, then installs the top panel onto the mounted side panels. **(G)** A manipulation sequence to assemble a car dashboard, two robot arms first grasp the workpiece then align multiple pins to the slots. **(H)** Two arms performing a coordinated handover task. **(I)** Two arms performing a timing belt installation task. **(J)** A manipulation sequence of Jenga whipping task, where the robot needs to extract one Jenga piece from the tower without crashing it. **(K)** The robot is flipping the object in the pan to the opposite side.

grasping pose; the policy must learn to account for such uncertainty during insertion. For example, if an unsuitable grasp was executed, the policy might need to release the object and regrasp it to achieve a better grasp pose.

USB Cable Clipping This task assumes a USB cable is already plugged into the motherboard, and the robot is tasked to pick up the remaining part of the cable and insert it into a tight-fitting organization clip. The task is considered successful if the USB cable is fully inserted into the clip. The difficulty lies in the variability of the deformable USB cable, as well as the tight insertion phase.

The Whole Assembly We also performed the whole assembly task by chaining the above four subtasks together, using scripted motions to transition between subtasks. A video clip of the entire assembly process can be found on our project website as well as in the supplementary material. The video demonstrates that the computer successfully booted up after executing the whole assembly policy, validating the effectiveness of our method in not only achieving task success but also performing the RL training process gracefully without damaging these delicate components.

IKEA Assembly The IKEA assembly task involves assembling an IKEA shelf with four boards, and is decomposed into three subtasks: the robot needs to first assemble the two side panels into a panel fixed on the table, then mount the top panels into side panels after they are assembled. The task is considered successful if all the pieces are properly assembled into the shelf. For all the subtasks, we assume the panels are pre-grasped by the robot, however, we do periodically place them back to fixtures and regrasp them to introduce grasping variations.

Side Panel Assembly In this task, the top part of the side panel is assumed to be pre-grasped by the robot, though the grasping location can vary due to the heavy weight of the panel during the interactive assembly process, and we do periodically regrasp them from the fixtures. The task is considered successful if the bottom part is properly assembled onto the two matching pins.

Top Panel Assembly After two side panels are assembled, this task demands the robot to mount the top panels onto both of the side panels. The task is considered successful if all four pins of the top panel are properly inserted into the corresponding holes on the side panel. This task is difficult because the top parts of the side panels can move around during the assembly process, and the policy must adapt to these variations to succeed.

The Whole Assembly We also performed the whole assembly task by chaining the three trained policies, using scripted motions to transition between subtasks, which is illustrated in Fig. 3 and the supplementary material. For each subtask, we uniformly randomized the scripted grasping poses by 1 cm in each translation dimension to introduce variations to the policy. This task is considered successful if all the panels are assembled successfully, and each sub-policy is allowed for a maximum of two attempts. For this task, we perform 10 trials for the chained policies, since it's a very long-horizon task.

Car Dashboard Assembly As illustrated in Fig. 3, the car dashboard assembly task involves two stages: both of the arms first need to grasp on appropriate locations of the workpiece and then lift it up, assemble it onto the dashboard. The task is considered successful if all the pins of the workpiece are fully inserted into the corresponding holes on the dashboard. This task requires precise manipulation as well as bimanual coordination: the two arms must coordinate the timing of the motions and gripper closure to lift the workpiece up, rotate it and align multiple pins at the same time.

Object Handover In this task, two robot arms are required to coordinate transferring an object from one basket to the other. The right arm first picks up the object from a basket on the right side. Then, it hands the object over to the left arm, which places it precisely into a basket on the left side. The task is considered successful if the object is placed flat on the right basket. The handover part is challenging because the robots' grippers must coordinate the timing of the motions to prevent the object from falling down.

Timing Belt Assembly In this task, two robot arms collaborate to assemble a timing belt onto pulleys and actuate the tensioner. This task is part of the NIST board assembly challenge (Kimble et al., 2020). The process involves locating and manipulating the randomly placed belt, coordinating precise motions to thread the belt over two pulleys, and simultaneously actuating the tensioner to accommodate the belt. Success is achieved when the belt is properly threaded onto both pulleys and the tensioner is securely tightened. This task presents several challenges. The belt can deform unpredictably during the assembly process, requiring adaptive manipulation. The arms must coordinate precisely, with carefully timed movements to thread the belt effectively. The timing of tensioner adjustments is critical, as the tensioner must allow the belt to be threaded while avoiding jamming. Throughout the process, the policy must continuously adjust to the changing state of the flexible belt and the overall system configuration. The complexity of this task stems from the need to handle a deformable object while precisely coordinating bimanual actions and managing the tensioner mechanism. This requires the policy to develop sophisticated, reactive behaviors to succeed consistently.

Jenga Whipping In this task, the robot is supposed to whip out a specific block from a Jenga tower without toppling over the tower. The nature of this task is largely different from the previous tasks, in that it requires the robot to learn a highly dynamic open-loop behavior, as opposed to the reactive closed-loop behavior required in the previous tasks. The dynamics of this task are intractably complex: the deformable whip travels at a very high speed and interacts with the surrounding compressed air, making its motion difficult to predict. Additionally, determining the precise force needed to remove a specific block without destabilizing the entire tower introduces further complexity due to the intricate contact dynamics involved. It is imperative for the policy to develop a reflex-like behavior by observing the outcomes of its own actions, intuitive physics, and the interactions between the whip and the blocks. This allows the robot to execute precise and consistent motions to successfully remove the target block without causing the tower to collapse. Note for this specific task, we initialize an offline dataset with 30 expert demonstrations rather than using real-time human corrections. This was chosen deliberately, as incorporating human feedback during training would be both impractical and unsuitable given the task's unique characteristics.

Object Flipping In this task, an object is randomly placed on a pan attached to the robot's end-effector, and the robot is tasked to flip the object over a horizontal axis. The task is considered successful if the object is flipped to the opposite side and remains in the pan. Since the initial placement of the object is randomized, the policy must learn to adapt to these variations, e.g., moving the object to a more favorable position before executing the flip motion. The task's nature is similar to the Jenga task, requiring precise and sophisticated open-loop behaviors. However, it also involves a closed-loop component, as the policy might need to reposition the object initially.

4.3. Experimental Results

In this subsection, we present the experimental results for all the tasks mentioned above. For each task, we report the success rate, cycle time, and training time. The training time includes all scripted motion, policy rollouts, intended stops, as well as onboard computation which is carried on a single NVIDIA RTX 4090 GPU. Unless otherwise noted, all results are based on 100 evaluation trials. During these evaluations, we randomize the initial states using either scripted robot motions or human resets. Our evaluation protocol can be found in the supplementary material.

A central claim of this paper is that HIL-SERL outperforms imitation learning methods based on human teleoperation. To substantiate this, it is crucial to compare relevant imitation learning methods fairly under equivalent settings. Naïve imitation learning approaches often suffer from error compounding issues, as noted by (Ross et al., 2011). DAgger and its variants (Ross et al., 2011; Kelly et al., 2018) address this problem by incorporating human corrections to refine the policy through supervised learning. Our method also leverages human corrections, but instead utilizes them to optimize the policy through reinforcement learning based on task-specific rewards. Therefore, we compare our approach to imitation learning by training a baseline with HG-DAgger (Kelly et al., 2018), using the same amount of human demonstrations

Task	Training Time (h)	Success Rate (%)		Cycle Time (s)	
		BC	HIL-SERL (ours)	BC	HIL-SERL (ours)
RAM Insertion	1.5	29	**100** (+245%)	8.3	**4.8** (1.7x faster)
SSD Assembly	1	79	**100** (+27%)	6.7	**3.3** (2x faster)
USB Grasp-Insertion	2.5	26	**100** (+285%)	13.4	**6.7** (2x faster)
Cable Clipping	1.25	95	**100** (+5%)	7.2	**4.2** (1.7x faster)
IKEA - Side Panel 1	2	77	**100** (+30%)	6.5	**2.7** (2.4x faster)
IKEA - Side Panel 2	1.75	79	**100** (+27%)	5.0	**2.4** (2.1x faster)
IKEA - Top Panel	1	35	**100** (+186%)	8.9	**2.4** (3.7x faster)
IKEA - Whole Assembly	–	1/10	**10/10** (+900%)	–	–
Car Dashboard Assembly	2	41	**100** (+144%)	20.3	**8.8** (2.3x faster)
Object Handover	2.5	79	**100** (+27%)	16.1	**13.6** (1.2x faster)
Timing Belt Assembly	6	2	**100** (+4900%)	9.1	**7.2** (1.3x faster)
Jenga Whipping	1.25	8	**100** (+1150%)	–	–
Object Flipping	1	46	**100** (+117%)	3.9	**3.8** (1.03x faster)
Average	–	49.7	**100** (+101%)	9.6	**5.4** (1.8x faster)

(a) Comparison of BC and RL success rates and cycle times for various tasks. All metrics were reported over 100 trials per task, except for the IKEA whole assembly task, which involved 10 trials. For all tasks, BC baselines were trained using HG-DAgger with the same number of episodes and interventions as RL. However, for the Jenga whipping and object flipping tasks, we used "flat" BC, trained on 50 and 200 demonstrations, respectively.

Task	DP	HG-DAgger	BC	IBRL	Residual RL	DAPG	HIL-SERL no demo no itv	HIL-SERL no itv	HIL-SERL (ours)
RAM Insertion	27	29	12	75	0	8	0	48	**100**
Dashboard Assembly	18	41	35	0	0	18	0	0	**100**
Object Flipping	56	46	46	95	97	72	0	**100**	**100**
Average	34	39	31	57	32	33	0	49	**100**

(b) Comparison of various methods on selected tasks. Diffusion Policy (DP) and BC are trained with 200 demonstrations, while HG-DAgger is trained with the same number of episodes and interventions as RL. IBRL, Residual RL, and DAPG are initialized with 200 demonstrations. Our method is also ablated with two versions: one initialized from scratch without demonstrations or corrections, and another initialized from demonstrations but without corrections.

Table 1: Experiment results. **(a)** HIL-SERL against imitation learning baselines. **(b)** HIL-SERL against various other baselines.

and corrections as used in reinforcement learning. We first pretrain a base policy with behavioral cloning (BC) using an equivalent amount of offline human demonstrations as provided to our method. We then run this policy and collect human expert corrections, such that the total amount of trials and interventions matches RL training. Specifically, we run it for the same number of episodes as our method and aim to provide a comparable number of interventions per episode.

This comparison is performed for all tasks except Jenga whipping and object flipping, where interventions are challenging and undesirable. For these tasks, we instead collect 50 and 200 offline demonstrations and train BC policies as baselines. This provides a significantly larger number of demonstrations than our method, which typically requires only 20-30 demonstrations.

In all our experiments, we use success rates and cycle time as primary metrics to compare different methods. To further validate the effectiveness of our approach, we also report the intervention rate over time, demonstrating that our policy improves progressively, reducing the need for interventions. Ideally, the intervention rate trends toward zero, indicating that the policy performs autonomously. The experiment results can be found in Fig. 4 and Table. 1a.

First, as shown in Table. 1, HIL-SERL achieved a success rate of 100% within 1 to 2.5 hours of real-world training on nearly all the tasks. This is a significant improvement over the HG-DAgger baseline, which

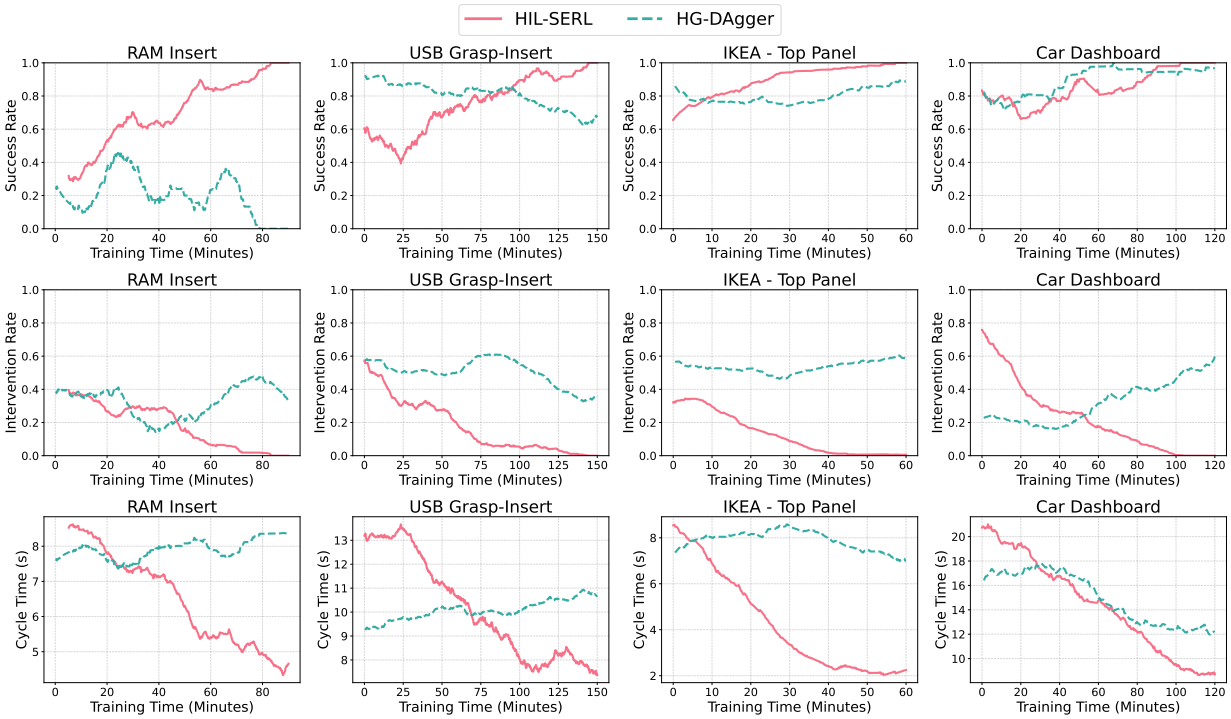

Figure 4: Learning curves for experimental tasks. This figure presents the success rate, cycle time, and intervention rates for both HIL-SERL and DAgger across few representative tasks, displayed as a running average over 20 episodes. For HIL-SERL, the success rate increased rapidly throughout training, eventually reaching 100%, while the intervention rate and cycle time progressively decreased, with the intervention rate ultimately reaching 0%. For HG-DAgger, the success rate fluctuates throughout training episodes and does not necessarily increase as training progresses. Since interventions occur frequently, leading to successful outcomes, the true policy success rate is likely lower than the curve suggests. Additionally, the intervention rate does not consistently decrease over time, indicating that the policy is not steadily improving. This is reflected in the cycle time as well, which shows no improvement, as DAgger lacks a mechanism to enhance performance beyond the provided training data. Additional plots are available in the supplementary material.

achieved an average success rate of 49.7% across all tasks. The performance gap is more pronounced for the tasks that require more complex behaviors – Jenga whipping, RAM stick insertion, and timing belt assembly.

We also report the number of human interventions over time for nearly all of the tasks in Fig. 4. Specifically, we report the intervention rate, for which we calculate the ratio of intervened timesteps to total timesteps within an episode and report a running average over 20 episodes. As shown in the figure, the intervention rate decreases as the training progresses, indicating that the policies are improving and less reliant on human corrections. Additionally, we observe that the total duration of interventions decreases dramatically. Initially, we issue long, sparse interventions when the policy is immature. As the policy improves, shorter interventions are sufficient to correct fewer mistakes. In contrast, the HG-DAgger policies require more frequent interventions to correct the policy, and the total duration of interventions does not necessarily decrease over time. Thus, our method attains better performance with less human supervision.

Our method outperforms HG-DAgger due to key advantages of RL. RL explores a wider range of states and directly optimizes task-specific rewards, while DAgger's reliance on human corrections can introduce inconsistencies and limit state exploration. As RL learns from its own state distribution and corrects errors autonomously, it overcomes the constraints of human demonstrations, resulting in more robust policies. These empirical findings align with theoretical results discussed in (Luo et al., 2023), which demonstrate that reinforcement learning (RL) policies can in principle outperform DAgger. The performance gap tends to widen as the suboptimality of human corrections increases, a scenario that is more likely to occur as

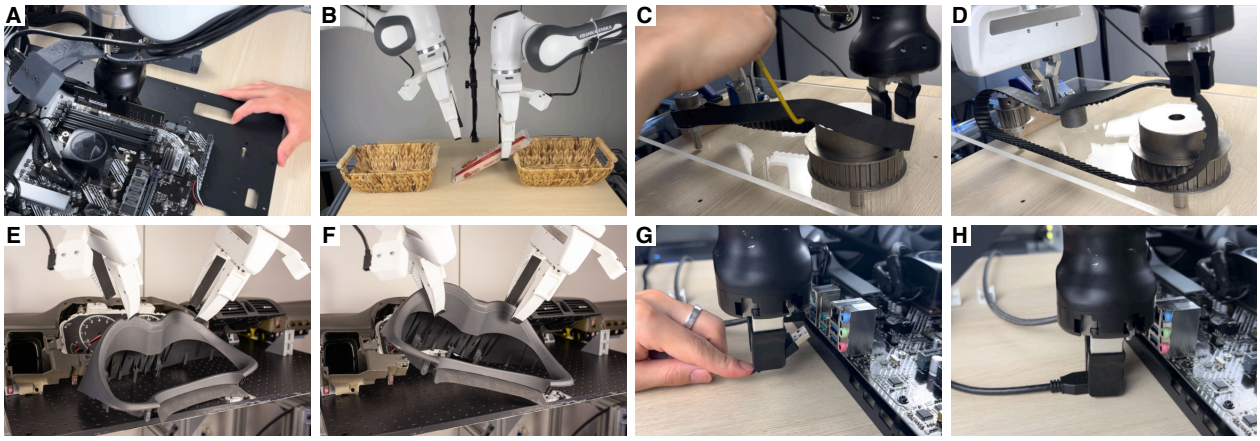

Figure 5: Robustness evaluation for policies learned by our method. **(A)** RAM insertion under external perturbations, such as a moving motherboard. **(B)** Retrying behavior during a handover task after the grippers are forced open. **(C-D)** Reactive responses in the timing belt task, addressing both external disturbances and unexpected deformations during execution. **(E-F)** In the dashboard assembly task, the policy performs re-grasps after one or both grippers are forcibly opened. **(G-H)** In the USB grasp-insertion task, the policy adapts to external disturbances and poor grasps by releasing and regrasping the object.

tasks become more complex.

Another important aspect to consider is the cycle time, or the time it takes to complete the task. On average, the HG-DAgger policies achieve an average cycle time of 9.6 s, while our method achieves an average cycle time of 5.4 s. This indicates an improvement of 1.8 times faster. This improvement is expected, as imitation learning methods lack mechanisms to deal with suboptimality in human demonstrations. In contrast, reinforcement learning (RL) can leverage dynamic programming to optimize for the discounted sum of rewards. For a discount factor $\gamma < 1$, this approach encourages the policy to acquire rewards faster, resulting in shorter cycle times compared to those achieved by imitating human demonstrations.

Out of these experiments, we would note that our method proves to be general and effective across tasks with vastly different physical properties, generating both open-loop and closed-loop policies well suited for each task's specific requirements. For precise manipulation tasks, such as assembling a timing belt or inserting a RAM stick, the policy learns to associate task-relevant visual features with appropriate twist motions. It performs continuous visual servoing behavior, reacting to streaming observations in real-time and adjusting its motion until reaching the target. In contrast, for tasks like Jenga whipping and object flipping, the policy learns to predict potential outcomes of its actions through interaction.

It then precisely refines its motion to achieve the desired outcome while maintaining consistency in execution. We also offer an in-depth analysis of the learned behavior, which we defer to the later section.

Overall, our experiments show that our method is both general and effective. It successfully learns policies for a variety of challenging manipulation tasks using the same approach, achieving high performance across all tasks. Additionally, it achieves such performance within practical training times, even for high-dimensional observations and action spaces, such as those required for bimanual manipulation.

4.4. Robustness Results

To test the zero-shot robustness of policies learned by our method, we provide a set of qualitative results in Fig. 5. These results demonstrate the policy's ability to adapt dynamically to variations on the fly and handle external disturbances, such as objects being intentionally dropped by a human from the gripper, or cases where a human deliberately forces the gripper open during task execution. Corresponding video clips can be found in our supplementary material and website https://hil-serl.github.io/

In the timing belt assembly task, the belt can undergo arbitrary deformation, and the policy is supposed to adapt to these variations during the assembly process. Additionally, we introduce external disturbances to the belt to further test the robustness of the policy. These disturbances include artificially perturbing

the belt's shape or repositioning it dynamically during the assembly process, as in Fig. 5 (C) and (D). In the RAM insertion task, the learned policy successfully inserts the RAM stick even when the target is moving during the insertion process, thanks to the ego-centric representation of the proprioceptive observation, as illustrated in Fig. 5 (A). For the car dashboard assembly and object handover tasks, after the policy grasps the object, we force the gripper to open during task exectution. The policy reacts to this unexpected situation by retrying to grasp the object and proceeding with the rest of the tasks, as presented in Fig. 5 (B), (E) and (F). In the USB connector grasp-insertion task, we varied the initial pose of the USB connector and occasionally forced it out of the gripper to simulate poor grasp poses. The policy adapted by releasing the connector and regrasping it to achieve a better pose for insertion, as seen in Fig. 5 (G) and (H). These robust behaviors are achieved through autonomous exploration during the RL training phase. For example, the policy learns to associate the grasping pose with the downstream insertion task and regrasps the object if necessary. However, these robust behaviors are usually hard to achieve with imitation learning methods, as they lack this mechanism to autonomously explore and learn from the outcomes of their actions.

4.5. Additional Baseline Comparisons

To validate the effectiveness of design choices in our method, we conducted additional comparisons on three representative tasks: car dashboard panel assembly (dual-arm coordination), RAM insertion (precise manipulation), and object flipping (dynamic manipulation). We compare our approach against several state-of-the-art methods to highlight different aspects of its performance. To illustrate the significance of human interventions, we performed ablation studies varying the number of human demonstrations and corrections. To showcase how effectively our method incorporates and leverages human demonstrations, we compared against Residual RL (Johannink et al., 2019), DAPG (Rajeswaran et al., 2018), and IBRL (Hu et al., 2024a). Additionally, we compare against Diffusion Policy (Chi et al., 2024), to ensure that the task difficulty does not stem solely from multi-modality in human demonstrations. These comprehensive comparisons serve to validate our method's effectiveness and capabilities across diverse manipulation scenarios, results are presented in Table. 1.

We first note that RL from scratch, without any demonstrations or corrections, achieved 0% success rate on all tasks. To validate the importance of online human corrections, we increased the number of demonstrations in the offline buffer of SERL tenfold, from the usual 20 to 200. However, without any online corrections, this approach resulted in significantly lower success rates compared to HIL-SERL, including a complete failure (0% success) on complex tasks such as the car dashboard assembly. This confirms the crucial role of online corrections in facilitating policy learning. These results confirmed the crucial role of both offline demonstrations and on-policy human interventions in guiding policy learning, especially for complex manipulation tasks that require continuous reactive behaviors.

For the object flipping task, we trained BC policies using both 20 and 200 demonstrations. The results from these two BC policies were quite similar, with success rates of 47% and 46%, respectively, even though the number of demonstrations increased tenfold. This indicates that merely imitating human demonstrations is insufficient to solve this task, even though it is largely open-loop.

Another important aspect to consider is how our method handles demonstrations compared to others. To compare against the mentioned baselines, we collected 200 demonstrations for each task. Note this number is substantially larger than the number of offline demonstrations in our method, which is usually around 20-30 in the offline replay buffer. For Residual RL and IBRL, we trained behavior cloning (BC) policies with these demonstrations to feed into their algorithm pipeline. For DAPG, we stored these 200 demonstrations in a separate buffer and regularized the policy actions towards them. Overall, our method consistently outperformed these baselines by large margins, as shown in Table. 1.

This can be interpreted as follows. Residual RL depends on a pre-trained BC base policy to facilitate the learning process. However, this approach can be problematic for tasks that require precise manipulation, such as car dashboard assembly or RAM insertion. In these scenarios, imitation learning methods, including BC, often perform inadequately. As a result, the RL policy learning process can experience significant failures. For IBRL, the actor is a hybrid of a BC policy and an RL policy, making the behavior more "BC-like."

This approach struggles on tasks where BC performs poorly. For DAPG, since it directly regularizes the policy actions towards the demonstrations, it is unsurprising that the policy performs similarly to the BC policies. Therefore, it underperforms our method on tasks that require more reactive and complex behaviors.

The effectiveness of our method comes from the off-policy nature of the underlying RL algorithm, which dynamically weights human data based on its relevance to the current policy optimization objective. Different from Johannink et al. (2019); Hu et al. (2024a); Rajeswaran et al. (2018) which heavily relies on the quality of human demonstrations, our method has a mechanism allows for efficient use of human data early in training while enabling the agent to progressively surpass human-level performance. Crucially, it prevents the agent from being constrained by the limitations of human demonstrations, striking a balance between bootstrapping from demonstrations and discovering novel, superior strategies through autonomous exploration.

To compare with the Diffusion Policy (Chi et al., 2024), We trained the policies using 200 demonstrations for each task, which is substantially more than the 20 demonstrations available in the offline replay buffers used by our method. We report the results using the experimented optimal algorithm parameters, such as observation and action chunking length, and the length of the applied action sequence in the supplementary material. On the RAM insertion and car dashboard panel tasks, diffusion policies achieved success rates of 27% and 28%, respectively. On the object flipping task, the success rate is 56%. This performance is significantly lower than our method and even falls short of the HG-DAgger baseline. This outcome is not surprising, as the primary strength of diffusion policies is in learning a more expressive policy distribution to accurately “memorize" robot motions. However, these tasks require more “closed-loop" reactive behaviors, such as continuous visual servoing to correct motion errors. Therefore, the advantage of diffusion policies in learning an expressive policy distribution does not necessarily lead to better performance in these tasks.

5. Result Analysis

To provide deeper insights into our results, we offer a detailed analysis of the learned policies. This analysis focuses on two key aspects: reliability and learned behaviors. We examine why the learned policies consistently achieve high success rates across diverse tasks, investigating the factors that contribute to their robustness. Additionally, we delve into the nature of the behaviors acquired by the policies, particularly exploring the distinction between reactive and predictive strategies. This comprehensive analysis aims to shed light on the underlying mechanisms that contribute to the effectiveness of our approach in solving complex manipulation tasks.

5.1. Reliability of the Learned Policies

One key aspect of HIL-SERL’s performance is its high reliability, achieving a 100% success rate across all tasks. We argue this reliability comes from reinforcement learning’s inherent ability to self-correct through policy sampling, allowing the agent to continuously improve by learning from both successes and failures. In contrast, imitation learning approaches, including interactive methods, lack this self-correction mechanism, making it significantly more challenging to achieve comparable performance with the same amount of data. While there is existing theoretical work on the convergence of Q-learning (Papavassiliou and Russell, 1999; Bhandari et al., 2018; Jin et al., 2020; Yang and Wang, 2019), our analysis focuses on providing an intuitive understanding of the training dynamics.

To illustrate this, we analyze the RAM insertion task, which requires precise manipulation and is easily visualized due to symmetrical randomization in the X and Y directions. We plot heatmaps of state visitation counts across timesteps for different policy checkpoints in Fig. 6, based on the end-effector’s Y and Z positions. Through policy learning, we observe the gradual development of a funnel-like shape connecting the initial states to the target location. This funnel becomes more defined as empty regions are filled, and narrows when approaching the target, indicating increased policy confidence and precision. Over time, the mass concentrates in areas likely to succeed. We then introduce the concept of “critical states", defined as

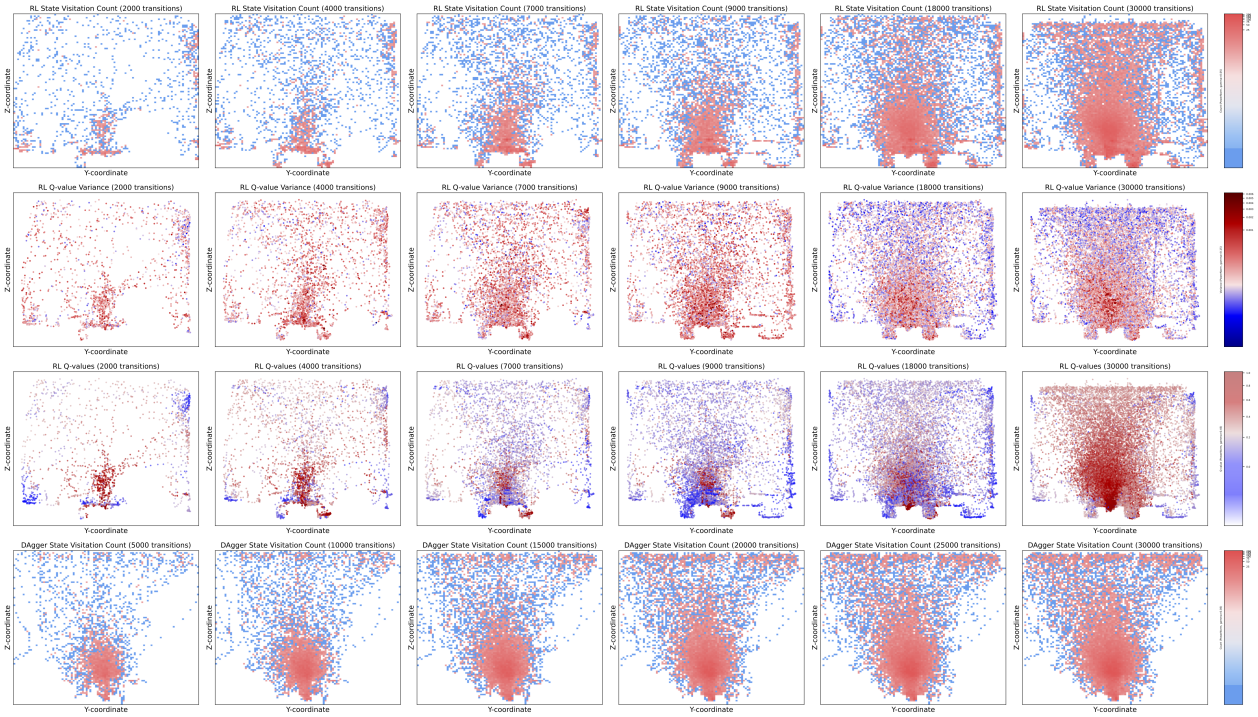

Figure 6: Visualization of policy training dynamics. (A) State visitation heatmaps during HIL-SERL training: The policy progressively forms a "funnel" shape, concentrating more on areas around the demonstrations and corrections, showing robustification in these regions. **(B)** Q-value variance scatter plots throughout training: States within the funnel show increased Q-value variance, indicating that the policy is gaining greater confidence in actions that lead to successful outcomes. **(C)** Q-value scatter plots across training: Critical states, marked by higher Q-value variance, are also associated with high Q-values. **(D)** State visitation heatmaps during HG-DAgger training: The funnel shape is less pronounced, more diffused distribution of visitation density.

states where the Q-function variance is large. We compute this variance using:

$$\mathrm{Var}[Q(\mathbf{s},\mathbf{a})] = \mathbb{E}_{\epsilon\sim\mathcal{U}[-c,c]}\left[\left(Q(\mathbf{s},\mathbf{a}+\epsilon) - \mathbb{E}_{\epsilon\sim\mathcal{U}[-c,c]}(Q(\mathbf{s},\mathbf{a}+\epsilon))\right)^2\right] \tag{4}$$

For each datapoint and its associated policy checkpoint, we add uniform random noise from [-0.2, 0.2] to the action (normalized to [-1, 1]) at every state and compute the Q-function variance using Monte Carlo sampling with 100 samples. A large variance indicates that the state is critical to the policy's success, as taking a different action would result in significantly different (usually much smaller) Q-values. Fig. 6 also shows heatmaps of Q-values and their variances at different states. These plots clearly demonstrate the policy developing a funnel where the most visited states gain higher Q-values and higher Q-value variances. This indicates that the policy is robustifying the region, effectively connecting important states with actions leading to high Q-values through dynamic programming.

In contrast, the heatmap of state visitation counts for HG-DAgger on the same task (fourth row of Fig. 6) shows a much sparser distribution. The funnel shape is less distinct, and the mass is more spread out, with states visited more uniformly compared to the RL case. This is because RL can explore autonomously and use dynamic programming directed by task rewards, while DAgger can only explore around the current policy. Consequently, to achieve similar performance, DAgger may require significantly more demonstrations and corrections, as well as careful attention from the human operator to ensure data quality.

This type of stabilizing behavior within a funnel has been studied in state-based dexterous manipulation and motion planning (Burridge et al., 1999; Tedrake et al., 2010). However, our approach differs in that we directly leverage perceptual inputs and use RL exploration to autonomously form the funnel. An analogous concept in optimal control is the development of controllers that stabilize around nominal trajectories using local feedback (Astrom and Murray, 2008). In our case, demonstrations and corrections can be regarded as

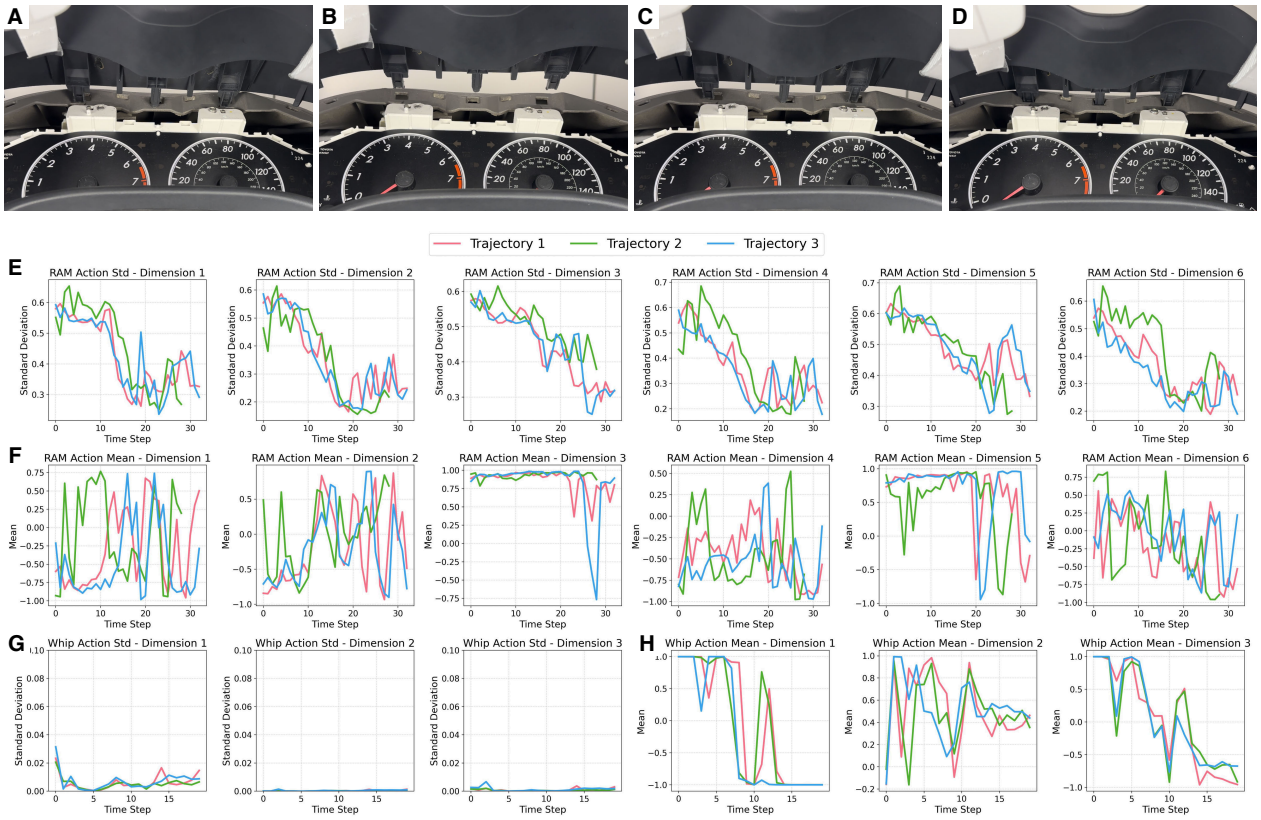

Figure 7: Reactive vs Predictive Behavior. (A-D) A sequence of reactive behaviors in the dashboard assembly task: after getting stuck in contact, the policy breaks the contact by quickly lifting two arms, then re-establishing the contact when approaching the target, finally succeeding in the insertion. **(E)** Variance plots from trained Gaussian policies in the RAM insertion task, showing three trajectories. Initial variance is high but rapidly decreases as the target is approached. **(F)** Mean plots from trained Gaussian policies in the RAM insertion task, with values ranging from -1 to 1. **(G)** Variance plots in the Jenga whipping task, remaining consistently low (near 0), indicating stable execution and open-loop behavior. **(H)** Mean plots in the Jenga whipping task, with values between -1 and 1, demonstrating consistent behavior across three trajectories.

"nominal trajectories" around which RL methods develop funnels for stabilization.

5.2. Reactive Policy and Predictive Policy

To solve most of the high-precision manipulation tasks, we need a closed-loop reactive policy that responds rapidly to immediate sensory feedback, therefore enabling precise adjustments in real-time. On the other hand, for the dynamic manipulation tasks, such as Jenga whipping and object flipping, it is desirable to employ an open-loop predictive policy that plans ahead and execute the motion consistently. To see this, we pick two representative tasks requiring these two different types of policies, Jenga whipping and RAM insertion, for analysis. To visualize the differences between these policy types, we plot the computed actions from the trained Gaussian policies for both tasks in Fig. 7.

For both tasks, we analyzed three successful trajectories by plotting the policies' computed standard deviation and mean over time. From these plots, we observe that while the mean actions cover a wide range of values in both cases, the standard deviations reveal distinct policy behaviors. In the Jenga whipping task, the standard deviation remains consistently low (very close to 0) across time steps. This indicates a highly confident and consistent policy, ideal for tasks where open-loop behaviors are desirable. Similar to a tennis player developing a reflex, the policy learns to execute a precise, pre-planned motion. Through environmental interactions, it refines this motion to minimize prediction errors, resulting in consistent execution. Conversely, the RAM insertion task exhibits a different pattern. Initially, the standard deviation is much higher (around 0.6), reflecting uncertainty when approaching the target early on. However, it

decreases rapidly over time, suggesting an initially coarse approaching motion that becomes more precise when near the target. This task demands a reactive policy capable of error correction in various scenarios, as predictive control over a long horizon is impractical due to the task's precision requirements. This reactive behavior is even more pronounced in complex manipulation tasks such as dashboard panel assembly or timing belt installation. In these cases, the policy must continuously adjust its actions based on sensory feedback, often requiring multiple attempts to achieve success, such as breaking contact and re-approaching the target, as illustrated in Fig. 7. The higher variance in these scenarios indicates the policy's readiness to react swiftly to changing conditions.

It's worthwhile to note that this type of reactive behavior is acquired by the agent through interacting with the environment. In other words, the agent develops this behavior "for free" - we don't explicitly formulate the problem to solve for a specific dynamic behavior. Instead, the desired response emerges naturally as part of the solution through ongoing interactions. Previous work Marcucci et al. (2017); Hogan and Rodriguez (2016); Aceituno-Cabezas and Rodriguez (2020) have attempted to formulate these contact-rich manipulation problems as mixed-integer programming for the resulting hybrid systems, which allows the policy to plan different modes of contact and the accommodating motions. However, these methods can quickly become computationally intractable as the planning horizon increases, since the number of possible contact modes grows exponentially with the length of the planning horizon. Additionally, they require accurate state estimators, which are not always available for many real-world tasks.

In contrast, our method directly leverages perception to learn reactive behaviors upon encountering contact. Through interaction, it encodes essential dynamics required to find a solution, rather than treating these dynamics as part of the problem formulation. Prior approaches, however, incorporate complex or intractable dynamics within the problem formulation itself, making these solutions harder to derive and less scalable.

Overall, our approach demonstrates the flexibility to learn these distinct policy types within a unified algorithmic framework. By interacting with the environment and observing the outcomes of their actions, the method adapts to the specific demands of each task. This adaptability enables the system to effectively address tasks that require diverse behaviors, spanning a wide range of manipulation challenges.

6. Discussion

The presented results substantially advance the published state-of-the-art in robotic manipulation. Our research demonstrates that with the right design choices, model-free RL can actually effectively tackle a variety of complex manipulation tasks using perception inputs, directly training in the real world within a practical timeframe. Trained policies from this approach are highly performant, achieving nearly perfect success rates that substantially exceed those of alternative approaches, such as imitation learning, along with cycle times that are also considerably faster.

Beyond the results themselves, the approach presented in this work can have significant broader impact. It can serve as a general framework for acquiring a wide range of manipulation skills with high performance and adapt to variations. This is particularly valuable in High-Mix Low-Volume (HMLV) manufacturing, or "make-to-order" production (Jina et al., 1997; Shah and Ward, 2003; Gan et al., 2023). Such production methods have substantial potential in major industries such as electronics, semiconductors, automotive, and aerospace due to their need for shorter product life cycles, customization, agility, and flexibility.

We see a number of opportunities for future work. First, our approach can serve as an effective tool for generating high-quality data to train robot foundation models (Brohan et al., 2023b;a; Collaboration et al., 2024; Team et al., 2024; Kim et al., 2024). Given that each task requires a relatively short time to train and the training process is largely autonomous, this framework can be employed to develop a variety of skills. Subsequently, data can be collected by executing the converged policies, which can then be distilled into these generalist models. Second, although the current training time is relatively short, each task still requires training from scratch. We can further reduce this time by pretraining a value function that encapsulates the general manipulation capabilities of solving a range of different tasks with distinct robot embodiments. This pretrained value function can then be quickly fine-tuned to address specific tasks.

We also see some limitations of our approach. Although we successfully address a variety of challenging tasks, it remains uncertain whether this method can be further extended to tasks with significantly longer horizons, where the sample complexity issue becomes more pronounced. However, this challenge might be alleviated through improved pretraining techniques or by employing methods that automatically segment a long-horizon task into a series of shorter sub-tasks, such as a vision-language model. It's also important to note that we did not perform extensive randomization in our experiments, nor did we test the method's generalization capability in unstructured environments. The primary focus of this paper is to demonstrate that the approach can be general-purpose in acquiring a wide range of manipulation skills with high performance. We believe that the randomization issue could be addressed by extending the training duration of the policies with the desired randomization level as in Luo et al. (2021). Additionally, the generalization issue might be resolved by incorporating vision foundation models that are pretrained on large scale diverse datasets.

We hope this work will pave the way for the use of reinforcement learning in solving robotic manipulation problems, achieving high performance and eventually deploying them into the real world.

Acknowledgments

The authors would like to thank Kyle Stachowicz and Qiyang Li for very helpful discussions.

Funding: This research was partly supported by the AI Institute, ONR N00014-20-1-2383, and NSF IIS-2150826.

Author contributions: J.L. designed the research, conceived the idea and devised the main method in the paper, implemented the prototype and performed initial experiments, provided hands-on guidance on all experiments, analyzed and interpreted the results, led the project, wrote and positioned the paper. C.X. contributed to the research design, maintained the main research codebase, prepared experiment hardware, performed a significant amount of experiments, prepared figures and videos for the paper. J.W. contributed to the research design, maintained the main research codebase, performed a large amount of experiments, cleaned up the codebase for public release. S.L. contributed to the research design, advised the project, edited and positioned the paper.

Competing interests: There are no competing interests to declare.

Data and materials availability: Accompanying videos and code can be found on the project website: https://hil-serl.github.io/.

References

B. Aceituno-Cabezas and A. Rodriguez. A global quasi-dynamic model for contact-trajectory optimization. In *RSS*, 2020.

Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, USA, 2008. ISBN 0691135762.

Mohammad Gheshlaghi Azar, Remi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model, 2012. URL https://arxiv.org/abs/1206.6461.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

Jalaj Bhandari, Daniel Russo, and Raghav Singal. A finite time analysis of temporal difference learning with linear function approximation. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1691–1692. PMLR, 06–09 Jul 2018.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023a. URL https://arxiv.org/abs/2307.15818.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023b. URL https://arxiv.org/abs/2212.06817.

R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, 1999. doi: 10.1177/02783649922066385.

Haonan Chang, Abdeslam Boularias, and Siddarth Jain. Insert-one: One-shot robust visual-force servoing for novel object insertion with 6-dof tracking. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2024)*, October 2024. URL https://www.merl.com/publications/TR2024-137.

Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search, 2016. URL https://arxiv.org/abs/1610.00529.

Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023. doi: 10.1126/scirobotics.adc9244. URL https://www.science.org/doi/abs/10.1126/scirobotics.adc9244.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, 2024.

Jinda Cui and J. Trinkle. Toward next-generation learned robot manipulation. *Science Robotics*, 6, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning?, 2020. URL https://arxiv.org/abs/1910.03016.

Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets_and_Benchmarks.html.

N. Fazeli, M. Oller, J. Wu, Z. Wu, J. B. Tenenbaum, and A. Rodriguez. See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion. *Science Robotics*, 4(26):eaav3123, 2019. doi: 10.1126/scirobotics.aav3123. URL https://www.science.org/doi/abs/10.1126/scirobotics.aav3123.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 482–495. PMLR, 13–15 Nov 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents, 2018. URL https://arxiv.org/abs/1705.06366.

Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. *Advances in neural information processing systems*, 31, 2018.

Zhi Lon Gan, Siti Nurmaya Musa, and Hwa Jen Yap. A review of the high-mix, low-volume manufacturing industry. *Applied Sciences*, 13(3), 2023. ISSN 2076-3417. doi: 10.3390/app13031687. URL https://www.mdpi.com/2076-3417/13/3/1687.

Abhishek Gupta, Justin Yu, Tony Z. Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 6664–6671. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9561384. URL https://doi.org/10.1109/ICRA48506.2021.9561384.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Todd Hester and Peter Stone. Texplore: real-time sample-efficient reinforcement learning for robots. *Machine learning*, 90:385–429, 2013.

Francois Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics, 2016. URL https://arxiv.org/abs/1611.08268.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning, 2024a. URL https://arxiv.org/abs/2311.02198.

Zheyuan Hu, Aaron Rovinsky, Jianlan Luo, Vikash Kumar, Abhishek Gupta, and Sergey Levine. REBOOT: reuse data for bootstrapping efficient real-world dexterous manipulation. *arXiv preprint arXiv:2309.03322*, 2024b.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. doi: 10.1126/scirobotics.aau5872. URL https://www.science.org/doi/abs/10.1126/scirobotics.aau5872.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013. doi: 10.1162/NECO_a_00393.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/d3b1fb02964aa64e257f9f26a31f72cf-Paper.pdf.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2137–2143. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/jin20a.html.

Shiyu Jin, Changhao Wang, and Masayoshi Tomizuka. Robust deformation model approximation for robotic cable manipulation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6586–6593, 2019. doi: 10.1109/IROS40897.2019.8968157.

Shiyu Jin, Xinghao Zhu, Changhao Wang, and Masayoshi Tomizuka. Contact pose identification for peg-in-hole assembly under uncertainties. In *2021 American Control Conference (ACC)*, pages 48–53, 2021. doi: 10.23919/ACC50511.2021.9482981.

Jay Jina, Arindam K. Bhattacharya, and Andrew Walton. Applying lean principles for high product variety and low volumes: some issues and propositions. *Logistics Information Management*, 10:5–13, 1997. URL https://api.semanticscholar.org/CorpusID:110348308.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029, 2019. doi: 10.1109/ICRA.2019.8794127.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018. URL https://arxiv.org/abs/1806.10293.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale, 2021. URL https://arxiv.org/abs/2104.08212.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynominal time. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 260–268, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

Michael Kelly, Chelsea Sidrane, K. Driggs-Campbell, and Mykel J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083, 2018. URL https://api.semanticscholar.org/CorpusID:52939433.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Kenneth Kimble, Karl Van Wyk, Joe Falco, Elena Messina, Yu Sun, Mizuho Shibata, Wataru Uemura, and Yasuyoshi Yokokohji. Benchmarking protocols for evaluating small parts robotic assembly systems. *IEEE Robotics and Automation Letters*, 5(2):883–889, 2020. doi: 10.1109/LRA.2020.2965869.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2020. URL https://arxiv.org/abs/1912.11370.

Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3232–3237, 2010. doi: 10.1109/IROS.2010.5649089.

Ilya Kostrikov, Laura M. Smith, and Sergey Levine. Demonstrating A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.056. URL https://doi.org/10.15607/RSS.2023.XIX.056.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986. URL https://www.science.org/doi/abs/10.1126/scirobotics.abc5986.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

Kevin Li, Abhishek Gupta, Ashwin Reddy, Vitchyr H. Pong, Aurick Zhou, Justin Yu, and Sergey Levine. MURAL: meta-learning uncertainty-aware rewards for outcome-driven reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6346–6356. PMLR, 2021. URL http://proceedings.mlr.press/v139/li21g.html.

Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021. doi: 10.1126/scirobotics.abg5810. URL https://www.science.org/doi/abs/10.1126/scirobotics.abg5810.

Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Aparicio Ojea, and Alice M Agogino. Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2062–2069. IEEE, 2018.

Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Aparicio Ojea, Alice M Agogino, Aviv Tamar, and Pieter Abbeel. Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3080–3087. IEEE, 2019.

Jianlan Luo, Oleg Sushkov, Rugile Pevceviciute, Wenzhao Lian, Chang Su, Mel Vecerik, Ning Ye, Stefan Schaal, and Jonathan Scholz. Robust Multi-Modal Policies for Industrial Assembly via Reinforcement Learning and Demonstrations: A Large-Scale Study. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.088.

Jianlan Luo, Perry Dong, Yuexiang Zhai, Yi Ma, and Sergey Levine. Rlif: Interactive imitation learning as reinforcement learning. *arXiv preprint arXiv:2311.12996*, 2023.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16961–16969, 2024a. doi: 10.1109/ICRA57147.2024.10610040.

Jianlan Luo, Charles Xu, Xinyang Geng, Gilbert Feng, Kuan Fang, Liam Tan, Stefan Schaal, and Sergey Levine. Multistage cable routing through hierarchical imitation learning. *IEEE Transactions on Robotics*, 40:1476–1491, 2024b. doi: 10.1109/TRO.2024.3353075.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: language-image representations and rewards for robotic control. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 23301–23320. PMLR, 2023a. URL https://proceedings.mlr.press/v202/ma23b.html.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b. URL https://openreview.net/pdf?id=YJ7o2wetJ2.

Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. Zero-shot reward specification via grounded natural language. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 14743–14752. PMLR, 2022. URL https://proceedings.mlr.press/v162/mahmoudieh22a.html.

Tobia Marcucci, Robin Deits, Marco Gabiccini, Antonio Bicchi, and Russ Tedrake. Approximate hybrid model predictive control for multi-contact push recovery in complex environments. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 31–38, 2017. doi: 10.1109/HUMANOIDS.2017.8239534.

M.T. Mason and K.M. Lynch. Dynamic manipulation. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, volume 1, pages 152–159 vol.1, 1993. doi: 10.1109/IROS.1993.583093.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Andrew Morgan*, Bowen Wen*, Junchi Liang, Abdeslam Boularias, Aaron Dollar, and Kostas Bekris. Vision-driven compliant manipulation for reliable; high-precision assembly tasks. In *Robotics: Science and Systems XVII*, RSS2021. Robotics: Science and Systems Foundation, July 2021. doi: 10.15607/rss.2021.xvii.070. URL http://dx.doi.org/10.15607/RSS.2021.XVII.070.

Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 1101–1112. PMLR, 2019. URL http://proceedings.mlr.press/v100/nagabandi20a.html.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606122.

Vassilis A. Papavassiliou and Stuart Russell. Convergence of reinforcement learning with general function approximators. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, page 748–755, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In Ali Jadbabaie, John Lygeros, George J. Pappas, Pablo A. Parrilo, Benjamin Recht, Claire J. Tomlin, and Melanie N. Zeilinger, editors, *Proceedings of the 3rd Annual Conference*

on Learning for Dynamics and Control, L4DC 2021, 7-8 June 2021, Virtual Event, Switzerland, volume 144 of *Proceedings of Machine Learning Research*, pages 1154–1168. PMLR, 2021. URL http://proceedings.mlr.press/v144/rafailov21a.html.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.049.

Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27:55–73, 2009.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

Gerrit Schoettler, Ashvin Nair, Jianlan Luo, Shikhar Bahl, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5548–5555, 2020. doi: 10.1109/IROS45743.2020.9341714.

Rachna Shah and Peter T Ward. Lean manufacturing: context, practice bundles, and performance. *Journal of Operations Management*, 21(2):129–149, 2003. ISSN 0272-6963.

Archit Sharma, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn. Autonomous reinforcement learning: Benchmarking and formalism. *arXiv preprint arXiv:2112.09605*, 2021.

Archit Sharma, Ahmed M. Ahmed, Rehaan Ahmad, and Chelsea Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. *CoRR*, abs/2303.01488, 2023. doi: 10.48550/arXiv.2303.01488. URL https://doi.org/10.48550/arXiv.2303.01488.

Kaushik Shivakumar, Vainavi Viswanath, Anrui Gu, Yahav Avigal, Justin Kerr, Jeffrey Ichnowski, Richard Cheng, Thomas Kollar, and Ken Goldberg. Sgtm 2.0: Autonomously untangling long cables using interactive perception. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5837–5843, 2023. doi: 10.1109/ICRA48891.2023.10160574.

Hee-Chan Song, Min-Cheol Kim, and Jae-Bok Song. Usb assembly strategy based on visual servoing and impedance control. In *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 114–117, 2015. doi: 10.1109/URAI.2015.7358873.

Yuda Song, Yifei Zhou, Ayush Sekhari, Drew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid RL: Using both offline and online data can make RL efficient. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=yyBis80iUuU.

Oren Spector, Vladimir Tchuiev, and Dotan Di Castro. Insertionnet 2.0: Minimal contact multi-step insertion using multimodal multiview sensory input, 2022. URL https://arxiv.org/abs/2203.01153.

Jianhua Su, Chuankai Liu, and Rui Li. Robot precision assembly combining with passive and active compliant motions. *IEEE Transactions on Industrial Electronics*, 69(8):8157–8167, 2022. doi: 10.1109/TIE.2021.3108710.

Te Tang, Hsien-Chung Lin, Yu Zhao, Wenjie Chen, and Masayoshi 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)*, pages 162–167, 2016. doi: 10.1109/COASE.2016.7743375.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

Jonas Tebbe, Lukas Krauch, Yapeng Gao, and Andreas Zell. Sample-efficient reinforcement learning in robotic table tennis. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 4171–4178. IEEE, 2021.

Russ Tedrake, Ian R. Manchester, Mark Tobenkin, and John W. Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010. doi: 10.1177/0278364910369189.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11(104):3137–3181, 2010. URL http://jmlr.org/papers/v11/theodorou10a.html.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015. URL https://arxiv.org/abs/1509.06461.

Vainavi Viswanath, Kaushik Shivakumar, Justin Kerr, Brijen Thananjeyan, Ellen Novoseller, Jeffrey Ichnowski, Alejandro Escontrela, Michael Laskey, Joseph E. Gonzalez, and Ken Goldberg. Autonomously untangling long cables, 2022. URL https://arxiv.org/abs/2207.07813.

Vainavi Viswanath, Kaushik Shivakumar, Jainil Ajmera, Mallika Parulekar, Justin Kerr, Jeffrey Ichnowski, Richard Cheng, Thomas Kollar, and Ken Goldberg. Handloom: Learned tracing of one-dimensional objects for inspection and manipulation, 2023. URL https://arxiv.org/abs/2303.08975.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240. PMLR, 2022. URL https://proceedings.mlr.press/v205/wu23c.html.

Annie Xie, Fahim Tajwar, Archit Sharma, and Chelsea Finn. When to ask for help: Proactive interventions in autonomous reinforcement learning. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6bf82cc56a5fa0287c438baa8be65a70-Abstract-Conference.html.

Lin F. Yang and Mengdi Wang. Sample-optimal parametric q-learning using linearly additive features, 2019. URL https://arxiv.org/abs/1902.04779.

Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.

Albert Zhan, Ruihan Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. In *Deep RL Workshop NeurIPS 2021*, 2021.

Tony Z. Zhao, Jianlan Luo, Oleg Sushkov, Rugile Pevceviciute, Nicolas Heess, Jon Scholz, Stefan Schaal, and Sergey Levine. Offline meta-reinforcement learning for industrial insertion. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6386–6393, 2022. doi: 10.1109/ICRA46639.2022.9812312.

Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 3651–3657. IEEE, 2019. doi: 10.1109/ICRA.2019.8794102. URL https://doi.org/10.1109/ICRA.2019.8794102.

Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real world robotic reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rJe2syrtvS.

Supplementary Materials

A. Task Setup and Policy Training Details

In this section, we provide details regarding how each task is set up, including hardware and software; as well as details on policy training.

A.1. RAM Insertion

Fig. 8 shows the hardware setup for the motherboard assembly task, which presents the robot, the camera placements, and the task arrangement.

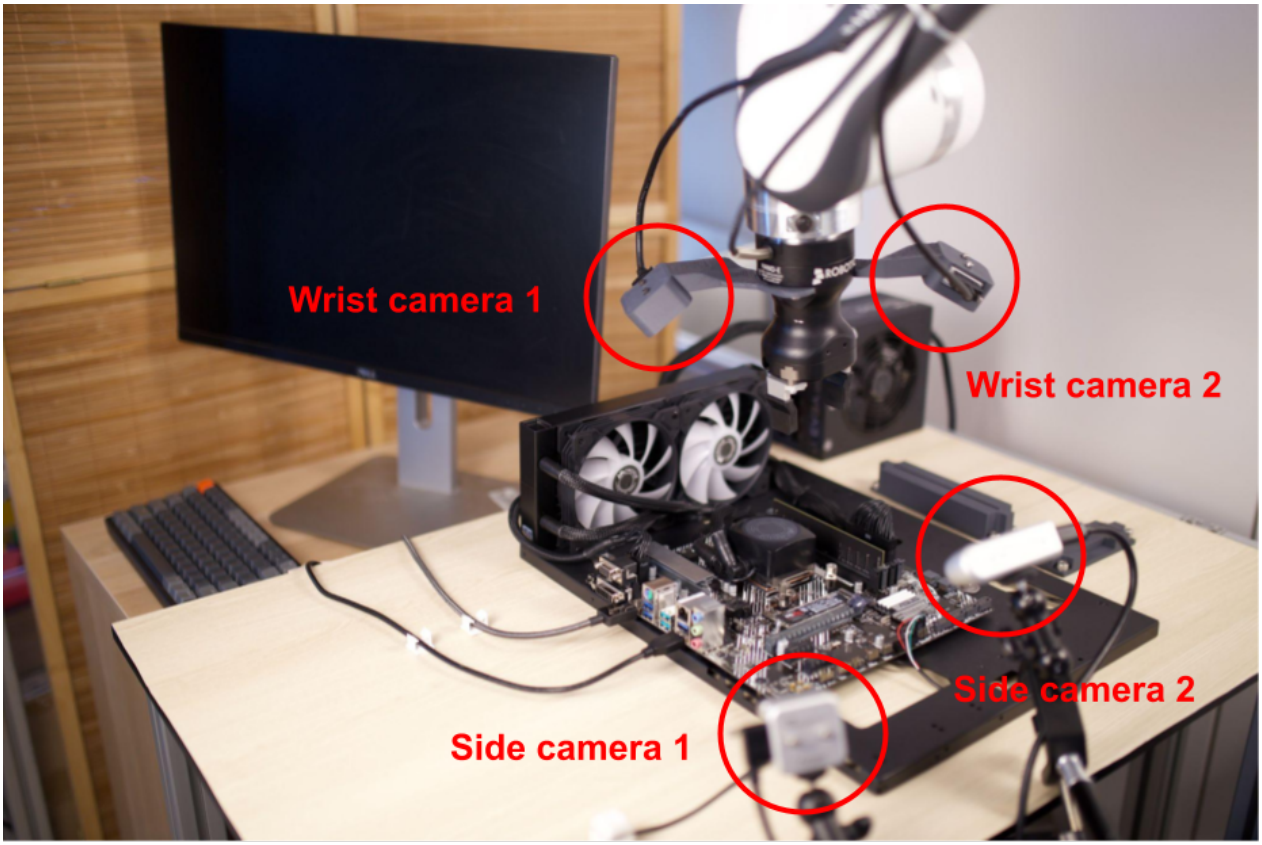

Figure 8: Hardware setup for the motherboard assembly task.

A.1.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 9.

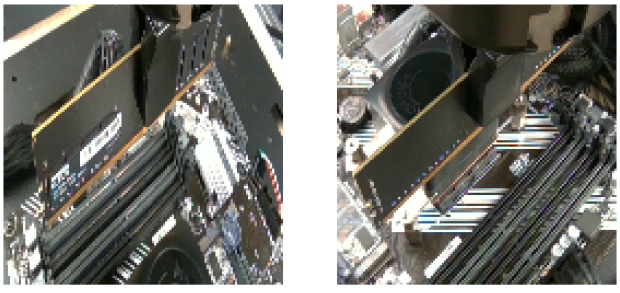

Figure 9: Sample input images from cameras used as inputs to the policy.

A.1.2. Policy Training Details

In Table 2, we report additional details of the policy training for this task.

Parameter	Value
Observation space	wrist_1, wrist_2, tcp_pose, tcp_vel, tcp_f/t
Action space	6D twist
Reward function	Binary classifier
Classifier views	wrist_1, wrist_2,
Classifier accuracy	97%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	100 environment steps
Reset method	Scripted reset
Randomization range	4 cm in x and y, 6 deg in rz
Proprio encoder size	64
Policy MLP size	256x256
Total number of RL transitions	32000
Discount factor	0.97
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 2: Policy training details for the RAM insertion task.

A.2. SSD Assembly

Fig. 8 shows the hardware setup for the motherboard assembly task, which presents the robot, the camera placements, and the task arrangement.

A.2.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 10.

Figure 10: Sample input images from cameras used as inputs to the policy.

A.2.2. Policy Training Details

In Table 3, we report additional details of the policy training for this task.

Parameter	Value
Observation space	wrist_1, wrist_2, side_2, tcp_pose, tcp_vel, tcp_f/t
Action space	6D twist
Reward function	Binary Classifier
Classifier views	wrist_1, wrist_2, side_2
Classifier accuracy	95%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	100 environment steps
Reset method	Scripted reset
Randomization range	2 cm in x and y, 1 deg in rz
Proprio encoder size	64
Policy MLP size	256x256
Total number of RL transitions	21000
Discount factor	0.97
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 3: Policy training details for the SSD assembly task.

A.3. USB Grasp-Insertion

Fig. 8 shows the hardware setup for the motherboard assembly task, which presents the robot, the camera placements, and the task arrangement.

A.3.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 11.

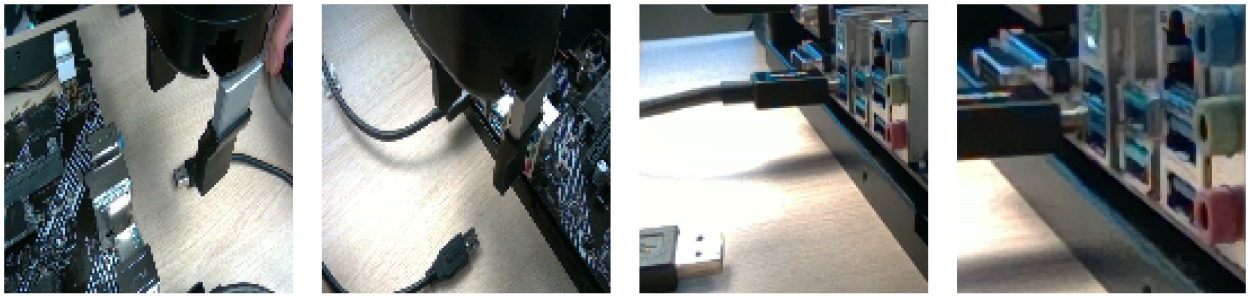

Figure 11: Sample input images from cameras used as inputs to the policy.

A.3.2. Policy Training Details

In Table 4, we report additional details of the policy training for this task.

Parameter	Value
Observation space	wrist_1, wrist_2, side_1, tcp_pose, tcp_vel, tcp_f/t, gripper_pos
Action space	6D twist and 1D discrete gripper control
Reward function	Binary classifier
Classifier views	side_1
Classifier accuracy	96%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	120 environment steps
Reset method	Scripted reset
Randomization range	2 cm in x and y, 10 deg in rz
Proprio encoder size	64
Motion policy MLP size	256x256
Grasp critic MLP size	256x256
Total number of RL transitions	50000
Discount factor	0.98
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 4: Policy training details for the USB grasp and insertion task.

A.4. Cable Clipping

Fig. 8 shows the hardware setup for the motherboard assembly task, which presents the robot, the camera placements, and the task arrangement.

A.4.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 12.

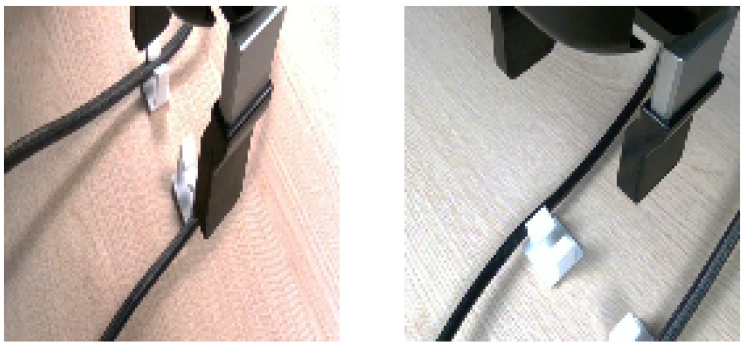

Figure 12: Sample input images from cameras used as inputs to the policy.

A.4.2. Policy Training Details

In Table 5, we report additional details of the policy training for this task.

Parameter	Value
Observation space	wrist_1, wrist_2, tcp_pose, tcp_vel, tcp_f/t, gripper_pos
Action space	6D twist and 1D discrete gripper control
Reward function	Binary classifier
Classifier views	wrist_1, wrist_2
Classifier accuracy	97%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	120 environment steps
Reset method	Human reset
Randomization range	4 cm in x and y, 10 deg in rz
Proprio encoder size	64
Motion policy MLP size	256x256
Grasp critic MLP size	256x256
Total number of RL transitions	28000
Discount factor	0.98
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 5: Policy training details for the cable clipping task.

A.5. IKEA - Side Panel

Fig. 13 shows the hardware setup for the IKEA assembly task, which presents the robot, the camera placements, and the task arrangement.

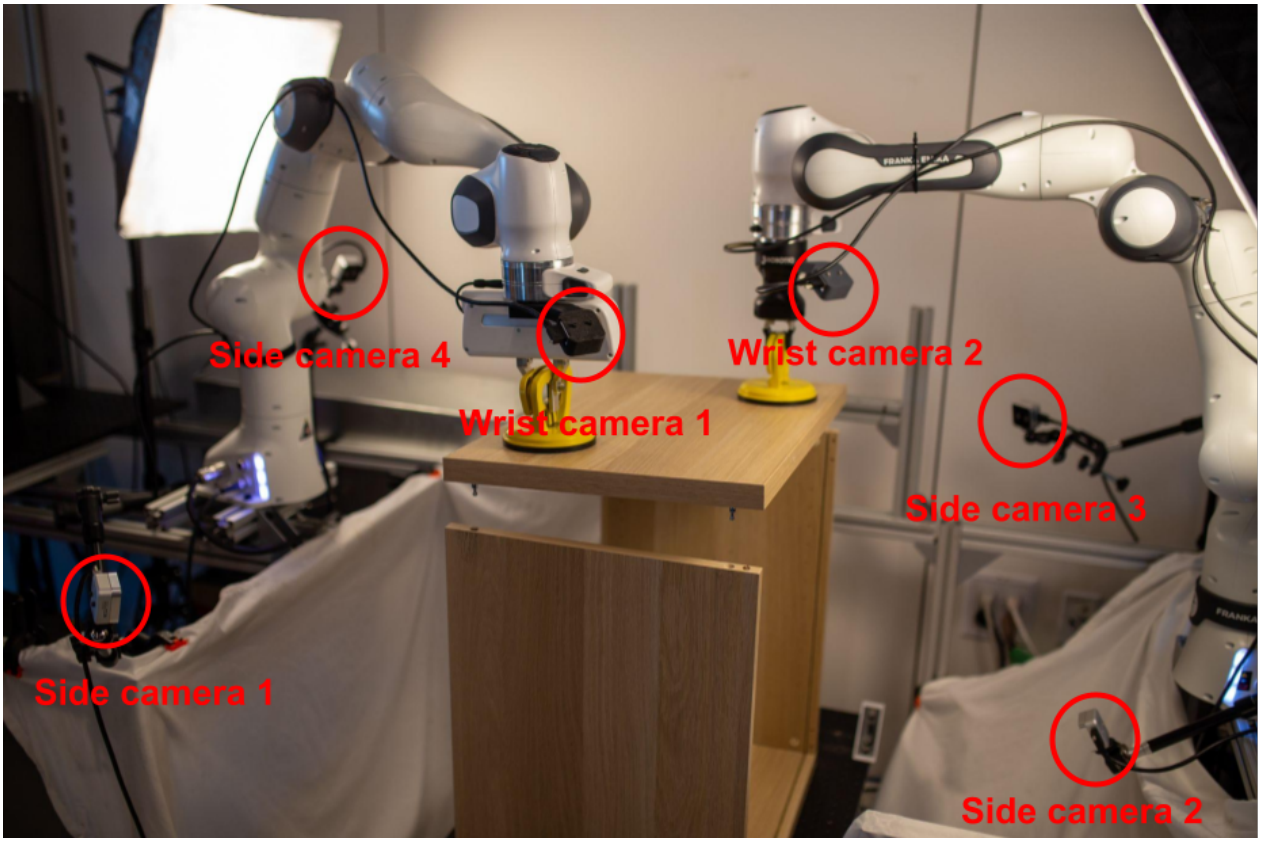

Figure 13: Hardware setup for the IKEA furniture assembly task.

A.5.1. Policy Training Details

In Table 6, we report additional details of the policy training for this task.

Parameter	Value
Observation space for side panel 1	wrist_1, side_1, side_2, tcp_pose, tcp_vel, tcp_f/t
Observation space for side panel 2	wrist_2, side_3, side_4, tcp_pose, tcp_vel, tcp_f/t
Action space	12D twist
Reward function	Binary Classifier
Classifier views for panel 1	side_1, side_2
Classifier views for panel 2	side_3, side_4
Classifier accuracy	97%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	100 environment steps
Reset method	Scripted reset
Randomization range	8 cm in x, y, 1 deg in rz
Proprio encoder size	64
Policy MLP size	256x256
Total number of RL transitions for panel 1	31000
Total number of RL transitions for panel 2	36000
Discount factor	0.98
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 6: Policy training details for the IKEA side panel task.

A.6. IKEA - Top Panel

Fig. 13 shows the hardware setup for the IKEA assembly task, which presents the robot, the camera placements, and the task arrangement.

A.6.1. Policy Training Details

In Table 7, we report additional details of the policy training for this task.

Parameter	Value
Observation space	side_1, side_3, side_4, tcp_pose, tcp_vel, tcp_f/t
Action space	12D twist
Reward function	Binary Classifier
Classifier views	side_1, side_3, side_4
Classifier accuracy	95%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	150 environment steps
Reset method	Scripted reset
Randomization range	3 cm in x, y
Proprio encoder size	64
Policy MLP size	256x256
Total number of RL transitions	18000
Discount factor	0.97
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 7: Policy training details for the IKEA top panel task.

A.7. Car Dashboard Assembly

Fig. 14 shows the hardware setup for the dashboard installation task, which presents the robot, the camera placements, and the task arrangement.

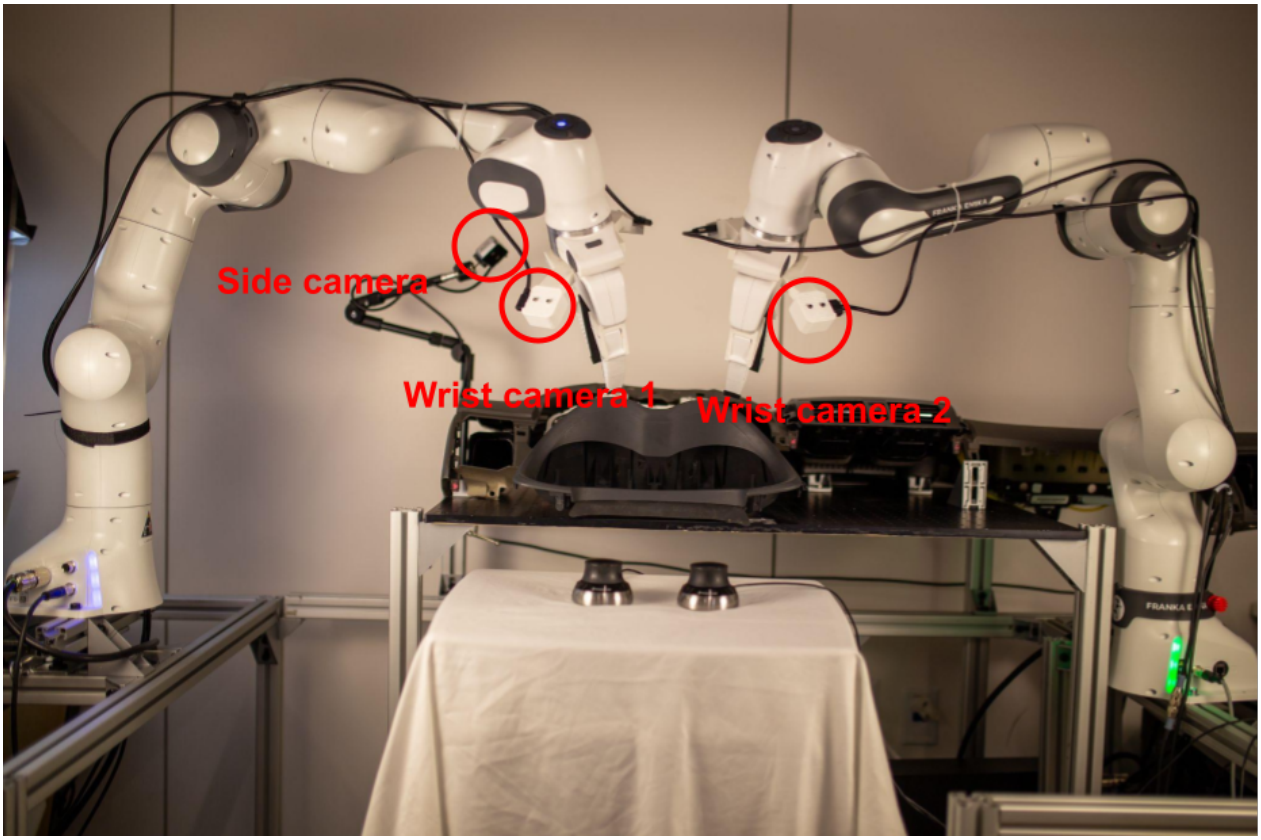

Figure 14: Hardware setup for the car dashboard installation task.

A.7.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 15.

A.7.2. Policy Training Details

In Table 8, we report additional details of the policy training for this task.

Figure 15: Sample input images from cameras used as inputs to the policy.

Parameter	Value
Observation space	wrist_1, wrist_2, side, tcp_pose, tcp_vel, tcp_f/t, gripper_pos
Action space	12D twist and 1D discrete gripper control
Reward function	Binary classifier
Classifier views	wrist_1, wrist_2, side
Classifier accuracy	98%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	200 environment steps
Reset method	Human reset
Randomization range	2 cm in x and y
Proprio encoder size	64
Motion policy MLP size	256x256
Grasp critic MLP size	256x256
Total number of RL transitions	36000
Discount factor	0.97
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 8: Policy training details for the car dashboard assembly task.

A.8. Object Handover

Fig. 16 shows the hardware setup for the object handover task, which presents the robot, the camera placements, and the task arrangement.

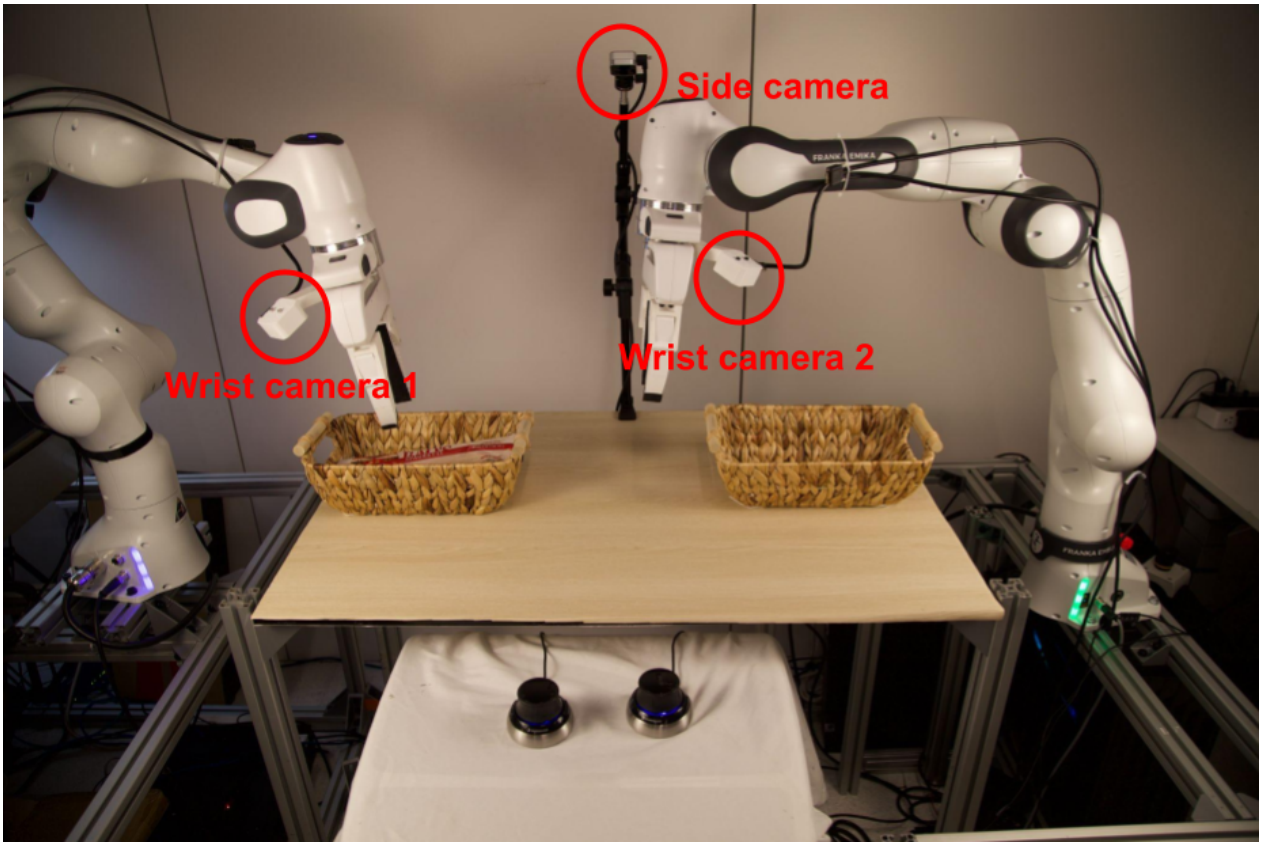

Figure 16: Hardware setup for the object handover task.

A.8.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 17.

Figure 17: Sample input images from cameras used as inputs to the policy.

A.8.2. Policy Training Details

In Table 9, we report additional details of the policy training for this task.

Parameter	Value
Observation space	wrist_1, wrist_2, side, tcp_pose, tcp_vel, gripper_pos
Action space	12D twist and 1D discrete gripper control
Reward function	Binary classifier
Classifier views	side
Classifier accuracy	99%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	200 environment steps
Reset method	Human reset
Randomization range	None
Proprio encoder size	64
Motion policy MLP size	256x256
Grasp critic MLP size	256x256
Total number of RL transitions	43000
Discount factor	0.97
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 9: Policy training details for the object handover task.

A.9. Timing Belt Assembly

Fig. 18 shows the hardware setup for the timing belt assembly task, which presents the robot, the camera placements, and the task arrangement.

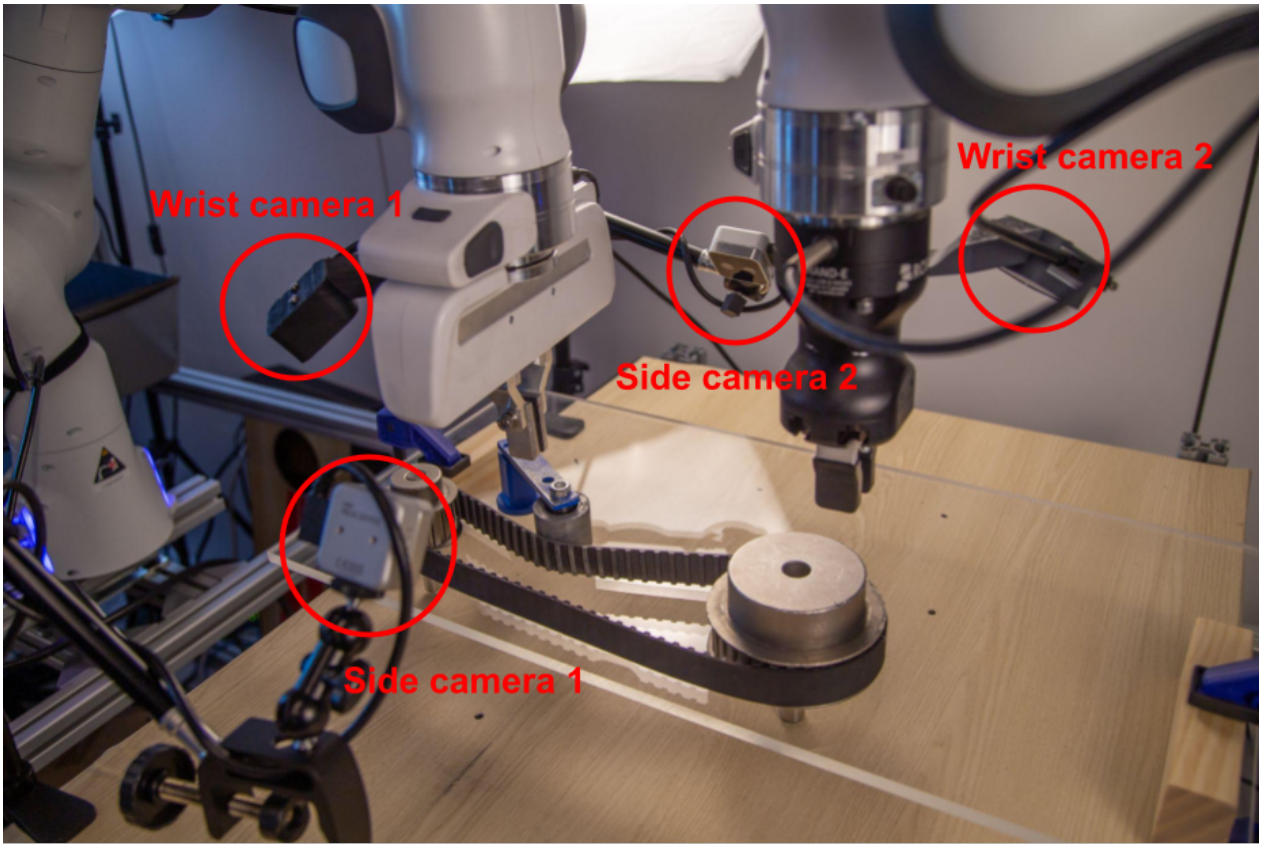

Figure 18: Hardware setup for the timing belt assembly task.

A.9.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 19.

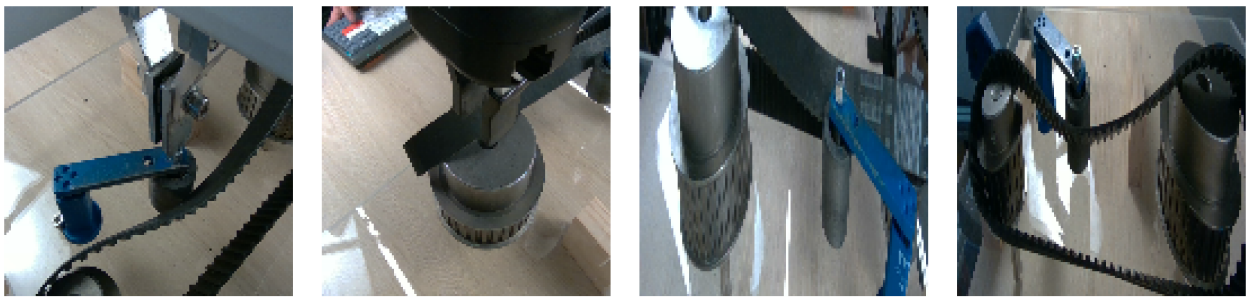

Figure 19: Sample input images from cameras used as inputs to the policy.

A.9.2. Policy Training Details

In Table 10, we report additional details of the policy training for this task.

Parameter	Value
Observation space	wrist_1, wrist_2, side_1, side_2, tcp_pose, tcp_vel, tcp_f/t
Action space	12D twist
Reward function	Binary classifier
Classifier views	side_1, side_2
Classifier accuracy	96%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	200 environment steps
Reset method	Human reset
Randomization range	2 cm in x and y
Proprio encoder size	64
Policy MLP size	256x256
Total number of RL transitions	108000
Discount factor	0.97
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 10: Policy training details for the timing belt assembly task.

A.10. Jenga Whipping

Fig. 20 shows the hardware setup for the Jenga whipping task, which presents the robot, the camera placements, and the task arrangement.

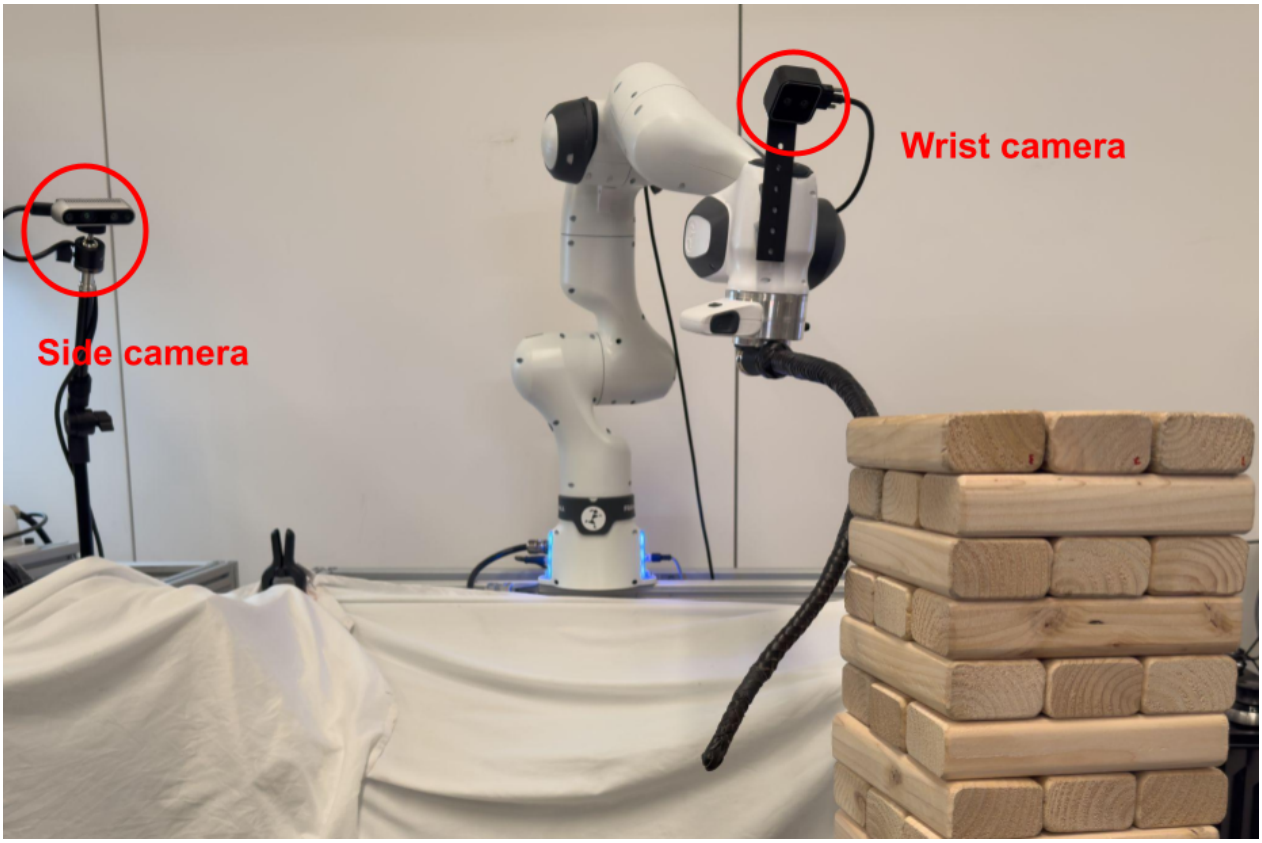

Figure 20: Hardware setup for the Jenga whipping task.

A.10.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 21.

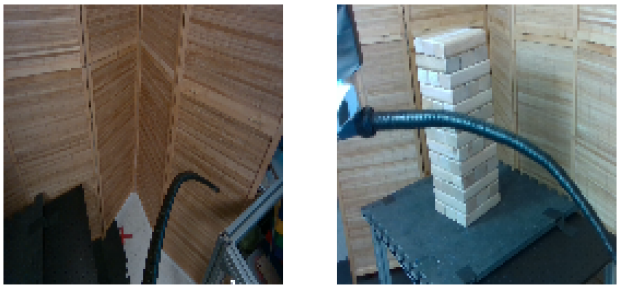

Figure 21: Sample input images from cameras used as inputs to the policy.

A.10.2. Policy Training Details

In Table 11, we report additional details of the policy training for this task.

Parameter	Value
Observation space	wrist, global, tcp_pose, tcp_vel, q, dq
Action space	Feedforward wrench F_x, F_z, τ_z
Reward function	Human annotation in the end of an episode
Environment update frequency	10 HZ
Max episode length	20 environment steps
Reset method	Human reset
Randomization range	None
Initial offline demonstrations	30
Proprio encoder size	64
Policy MLP size	256x256
Total number of RL transitions	10000
Discount factor	0.96, but every episode was run to maximum length
Optimizer	Adam
Learning rate	3e-4, decayed to 3e-5 when reaching 70% success rate
Image augmentation	Random crop

Table 11: Policy training details for the Jenga whipping task.

A.11. Object Flipping

Fig. 22 shows the hardware setup for the object flipping task, which presents the robot, the camera placements, and the task arrangement.

Figure 22: Hardware setup for the object flipping task.

A.11.1. Cropped Images

We cropped the images to focus on the task-relevant parts of the scene, as shown in Fig. 23.

Figure 23: Sample input images from cameras used as inputs to the policy.

A.11.2. Policy Training Details

In Table 12, we report additional details of the policy training for this task.

Parameter	Value
Observation space	wrist, side, tcp_pose, tcp_vel, q, dq
Action space	Feedforward wrench F_x, F_z, τ_y
Reward function	Binary classifier
Classifier views	wrist
Classifier accuracy	97%
Initial offline demonstrations	20
Environment update frequency	10 HZ
Max episode length	100 environment steps
Reset method	Scripted reset
Randomization range	None
Proprio encoder size	64
Policy MLP size	256x256
Total number of RL transitions	25000
Discount factor	0.985
Optimizer	Adam
Learning rate	3e-4
Image augmentation	Random crop

Table 12: Policy training details for the object flipping task.

B. Reward Classifier Training Details

For the reward classifiers, we used a pre-trained ResNet-10 model as the feature extractor, and connected it to a two-layer MLP, and we then trained the network on the collected dataset with cross-entropy loss. The classifier was trained using the Adam optimizer with a learning rate of 3e-4. The total number of training iterations was 100.

To collect the training dataset, we teleoperated the robot to perform the task and recorded the images and labels with a SpaceMouse. We clicked the SpaceMouse button when the robot successfully completed the task, and marked those images with labels as 1. Otherwise, we marked the labels as 0. In some of the tasks, we also recorded additional false positive and false negative samples to improve the classifier performance. We represent a few such examples in Fig. 24 to help readers understand how to train such classifiers.

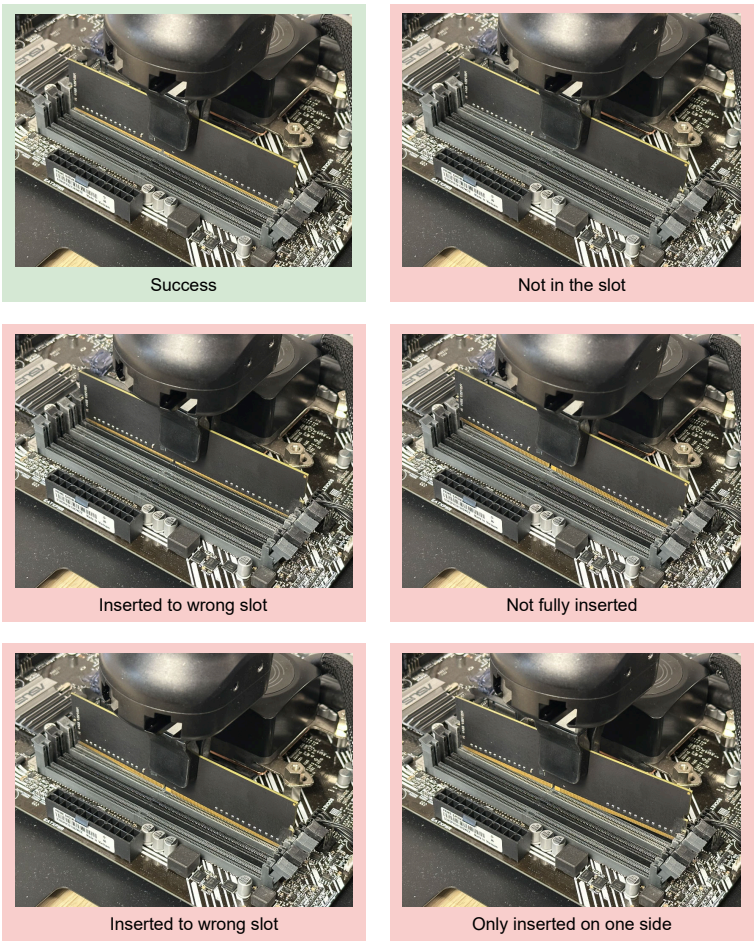

Figure 24: Sample images collected to train the reward classifier for the RAM insertion task.

C. Diffusion Policy Training Details

In this section, we provide detailed parameters for how we train the diffusion policy baselines, presented in Table. 13.

Task Name	Number of Demos	Observation Chunking Size	Action Prediction Horizon	Action Chunking Size
RAM Insertion	200	1	8	2
Dashboard Assembly	200	1	8	4
Object Flipping	200	1	1	1

Table 13: Diffusion policy training details.

D. Robot Controller and Proprioceptive Information Representation

In this section, we detail the implementation of the robot controller and the representation of the proprioceptive information for the robots.

D.1. Proprioceptive Information Representation

Let the robot's base frame be $\{s\}$; for the i-th episode of rolling out the policy, we denote $\{b_t^{(i)}\}$ as the end-effector frame expressed w.r.t. $\{s\}$ at a particular time step t; where $1 \leq i \leq M$, $0 \leq t \leq N$. For each episode, $\{b_0^{(i)}\}$ is sampled from a uniform distribution specifying the area of randomization. We want to express such proprioceptive information with respect to $\{b_0^{(i)}\}$. Thus, the policy will be applicable to a new location provided that the relative spatial distance between the robot's end-effector and the target remains consistent. This approach prevents overfitting to specific global locations within the reference frame $\{s\}$. We achieve this by applying the following homogeneous transformation:

$$T_{b_0^{(i)} b_t^{(i)}} = T_{b_0^{(i)}}^{-1} \cdot T_{b_t^{(i)}}$$

where we use T_{ab} to denote the homogeneous transformation matrix between frame $\{a\}$ and $\{b\}$. We feed the position and rotation information extracted from $T_{b_0^{(i)} b_t^{(i)}}$ to the policy. Here we use T_{ab} to denote the homogeneous transformation matrix between frame $\{a\}$ and $\{b\}$, defined as:

$$T_{ab} = \begin{bmatrix} R_{ab} & p_{ab} \\ 0_{1\times 3} & 1 \end{bmatrix}. \tag{5}$$

For most of the tasks, the policy generates a six-degree-of-freedom (6 DoF) twist action, which is expressed in the reference frame from which it currently receives observations, i.e., $\{b_t^{(i)}\}$. Mathematically, the 6 DoF twist action $\mathcal{V}_t^{(i)}$ is expressed in frame $\{b_t^{(i)}\}$ at timestep t. To interface with the robot's control software, which expects actions $\mathcal{V}_t^{(i)'}$ expressed in the base frame $\{s\}$, we apply the adjoint mapping:

$$\mathcal{V}_t^{(i)'} = [\mathrm{Ad}_t^{(i)}]^{-1} \mathcal{V}_t^{(i)}$$

where $[\mathrm{Ad}_t^{(i)}]$ is a function of the homogeneous transformation $T_{b_0^{(i)}}$ defined as:

$$[\mathrm{Ad}_t^{(i)}] = \begin{bmatrix} R_{b_t^{(i)}} & 0_{3\times 3} \\ [p_{b_t^{(i)}}] \times R_{b_t^{(i)}} & R_{b_t^{(i)}} \end{bmatrix}. \tag{6}$$

For two dynamic manipulation tasks, the policy generates a 3 DoF feedforward wrench action, which is also expressed in the reference frame from which it currently receives observations, i.e., $\{b_t^{(i)}\}$. It will then be sent to the low-level robot controller as setpoints, which then will be converted to joint torques for execution by multiplying the transpose of the Jacobian matrix at the current timestep t.

D.2. Robot Controller

For most of the tasks, the low-level robot controller is an impedance controller running at 1000 Hz, which accepts 10 Hz setpoints computed by the policy. As discussed in (Luo et al., 2024a), we perform additional treatment on it to ensure the stability of the training process in most contact-rich manipulation tasks. Consider a typical impedance controller without feedforward term:

$$F = k_p \cdot e + k_d \cdot \dot{e} + F_{ff} + F_{cor}, \tag{7}$$

where $e = p - p_{ref}$, p is the measured pose, and p_{ref} is the target pose computed by the upstream controller, F_{ff} is the desired feedforward force, F_{cor} is the Coriolis force. This objective will then be converted into joint space torques by multiplying Jacobian transpose and offset by nullspace torques. It acts like a spring-damper system around the equilibrium set by p_{ref} with the stiffness coefficient being k_p and the damping coefficient

being k_d. As described above, this system will yield large forces if p_{ref} is far away from the current pose, which can lead to a hard collision or damage when the arm is in contact with something. Therefore it's crucial to constrain the interaction force generated by it. However, directly reducing gains will hurt the controller's accuracy. Thus, we should bound e so that $|e| \leq \Delta$, and then the generated force from the spring-damper system will be bounded to $k_p \cdot |\Delta| + 2k_d \cdot |\Delta| \cdot f$, f is the control frequency.

For the two dynamic manipulation tasks, we run a feedforward wrench controller at 1000 Hz, which accepts 10 Hz setpoints computed by the policy. It converts the desired wrench into joint torques by multiplying the Jacobian transpose and offset by nullspace torques.

E. Policy Training Plots

In this section, we provide additional plots for HIL-SERL policy training for all tasks.

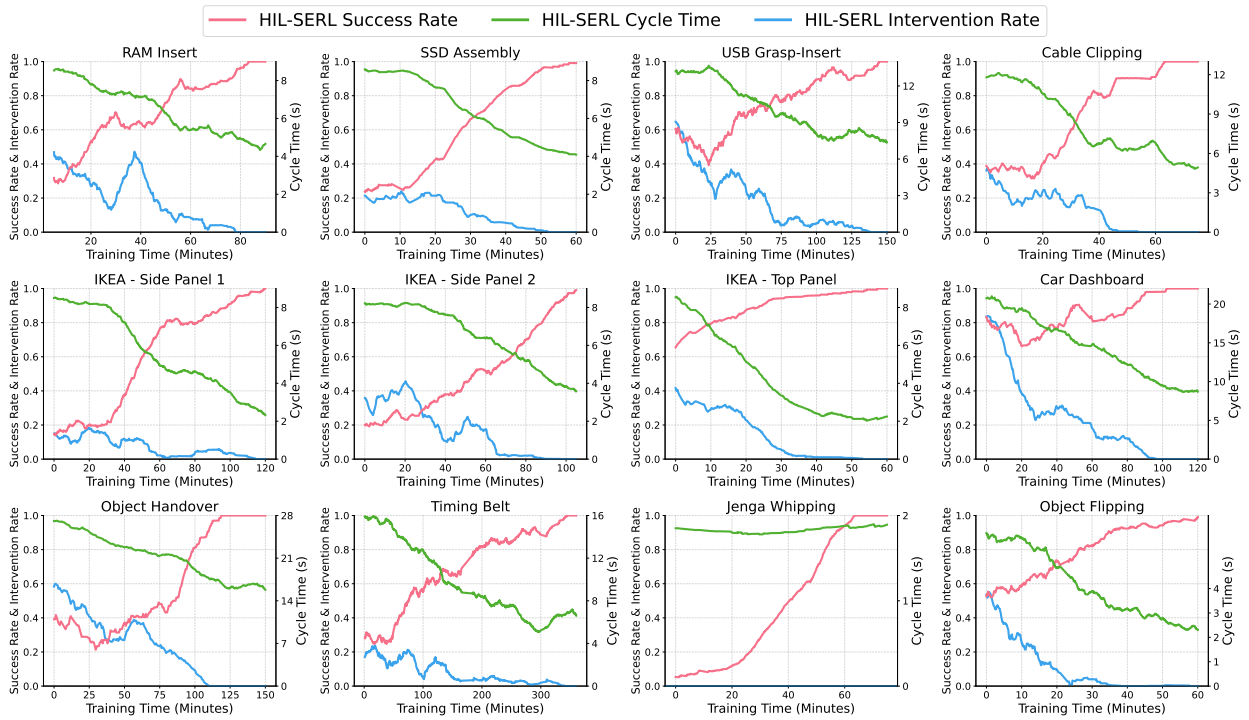

Figure 25: Learning curves for experimental tasks. This figure presents the success rate, cycle time, and intervention rates for both HIL-SERL across all experiment tasks, displayed as a running average over 20 episodes. The success rate increased rapidly throughout training, eventually reaching 100%, while the intervention rate and cycle time progressively decreased, with the intervention rate ultimately reaching 0%.