From Imitation to Refinement – Residual RL for Precise Assembly

Lars Ankile[1,2,3] Anthony Simeonov[1,2] Idan Shenfeld[1,2] Marcel Torne[1,2] Pulkit Agrawal[1,2]

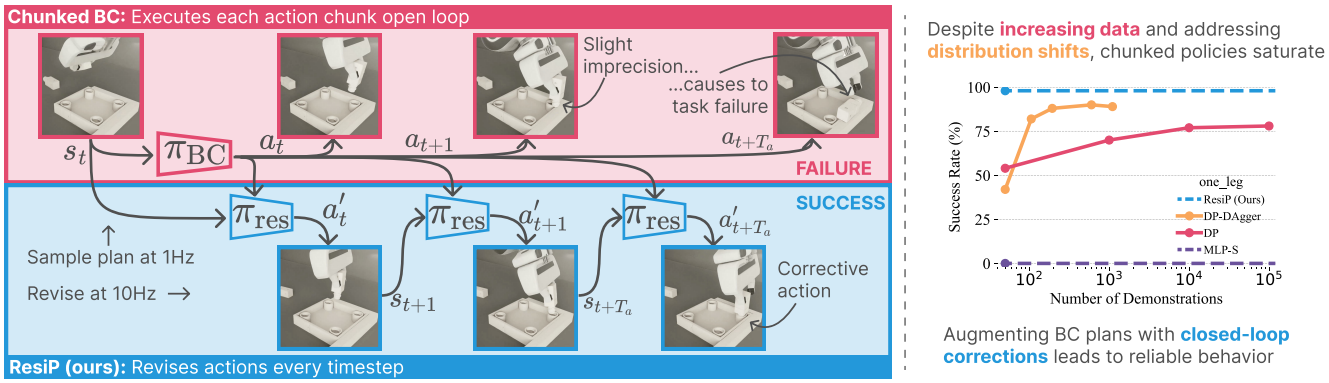


Fig. 1. Left: (Top) Tasks like assembly require long-horizon coordination and high-precision control, at which state-of-the-art BC methods fail due to their chunk-level open-loop nature. (Bottom) Combining a BC trajectory planner with a closed-loop residual policy trained with RL results in surprisingly robust and reactive behaviors. **Right:** Chunking improves performance over standard policy architectures. Still, performance saturates despite increasing data and addressing distribution shifts with DAgger [1]. Combining chunking with closed-loop corrections (`ResiP`) combines the strength of each.

***Abstract*— Recent advances in Behavior Cloning (BC) have made it easy to teach robots new tasks. However, we find that the ease of teaching comes at the cost of unreliable performance that saturates with increasing data for tasks requiring precision. The performance saturation can be attributed to two critical factors: (a) distribution shift resulting from the use of offline data and (b) the lack of closed-loop corrective control caused by action chucking (predicting a set of future actions executed open-loop) critical for BC performance.**

Our key insight is that by predicting action chunks, BC policies function more like trajectory "planners" than closed-loop controllers necessary for reliable execution. To address these challenges, we devise a simple yet effective method, `ResiP` (Residual for Precise Manipulation), that overcomes the reliability problem while retaining BC's ease of teaching and long-horizon capabilities. `ResiP` augments a frozen, chunked BC model with a fully closed-loop residual policy trained with reinforcement learning (RL) that addresses distribution shifts and introduces closed-loop corrections over open-loop execution of action chunks predicted by the BC trajectory planner. Videos, code, and data are available at residual-assembly.github.io.

I. Introduction

Many robotic manipulation tasks, such as assembly, require both long-horizon planning and high-precision control, which remains a significant challenge [2]–[6]. As an example, consider the furniture assembly task depicted in Fig. 1 (Left), where the robot needs to perform a sequence of steps: grasp, then re-orient the table leg into the correct pose, insert the leg, and finally screw the leg into place. This representative long-horizon task spans several task stages over hundreds of timesteps, each dependent on the successful completion of the previous. During critical moments such as insertions, called "bottleneck states," even slight imprecisions can compound and result in task failure, underscoring the need for reliable execution of each phase.

Behavior cloning (BC) is a popular approach for teaching robots various manipulation skills [6]–[16]. Recent innovations in BC, such as diffusion models [17]–[20] and action chunking (predicting a sequence of future actions) [6,17,19, 21], have enabled learning long-horizon, complex behaviors from demonstrations [15,22]. However, our analysis shows fundamental limitations in BC when applied to tasks requiring high precision: performance saturates with increasing data. For example, the success of a diffusion-based BC model on an insertion task shown in Fig. 1 (Left) plateaus at ∼80% even with 100,000 demonstrations (Fig. 1; right). Recent work finds similar performance saturation [23]–[25].

We hypothesize that the performance saturation results from two issues: (i) compounding errors originating from distribution shift as the policy operates on states that increasingly deviate from those seen during training [23]. (ii) The use of action-chunking which enables long-horizon control at the cost of reactivity necessary for reliable execution by rolling out each action chunk open-loop [26]. Therefore, we posit that chunked BC policies are best considered "planners" rather than reactive controllers. For instance, in the `one_leg` assembly task shown in Fig. 1 (left), specific bottleneck states, like insertions, require precise actions at specific time steps. If these critical moments fall within an action chunk, the BC policy cannot make real-time adjustments to compensate for disturbances during execution or inaccuracies in the planned actions.

Reinforcement Learning (RL) is a standard solution to the suboptimal performance of BC polices [27]–[39]. RL

[1]Improbable AI Lab [2]Massachusetts Institute of Technology [3]Harvard University `ankile@mit.edu`

fine-tuning effectively overcomes the problem of distribution shifts by generating training data for states visited by the policy. However, recent advancements in BC architectures present new challenges for direct RL fine-tuning. The main issues are that the structure of diffusion models (iterative refinement) and action-chunked policies (resulting in a large action space) make standard RL algorithms unstable (see Sec. IV-A) or require significant architectural modifications [15,40,41].

Instead of invoking RL, one can overcome distribution shift by leveraging an expert and training with supervised learning such as the Dataset Aggregation (DAgger) [1] algorithm. Our experiments show with roughly 2000 demonstrations, a DAgger style approach achieves a 90% success rate on the `one_leg` task (Fig. 1)—a significant improvement over pure BC but still 8% below the expert. One challenge in using DAgger is that it assumes an expert who can be queried on demand, which is usually unavailable. However, another significant challenge is the performance saturation of DAgger, which we attribute to the lack of reactivity because each action chunk is executed in an open-loop fashion.

Our key insight is that action-chunked BC policies function more like "trajectory planners" than reactive controllers. To mitigate both distribution shift and lack of closed-loop control, we use a simple method of augmenting a frozen, chunked BC model with a small, single-step residual policy [42]–[49] trained via on-policy RL (Fig. 2), termed `ResiP` (*Resi*dual for *P*recise Manipulation). Unlike prior residual frameworks, the BC policy generates trajectory segments at coarse temporal resolution (∼1 second), while the residual predicts closed-loop corrections at each control step (∼0.1 second).

`ResiP` addresses both challenges faced by BC methods: the RL-trained residual overcomes distribution shifts and by closing the sensorimotor loop more frequently it lends reactivity by enabling precise adjustments that chunked policies cannot provide. Using only a sparse task-completion reward, this approach achieves a 98% success rate starting with just 50 expert demonstrations on the `one_leg` task—an 18 percentage point improvement over BC with 100K demonstrations.

Since our method relies on training RL policies, our learning pipeline heavily relies on simulation. Applying our pipeline requires constructing a digital twin of the real-world use case (i.e., real-to-sim [16]), training with `ResiP`, and finally transferring the trained policy to the real world. While our focus in this paper is on `ResiP` and highlighting the performance saturation of purely BC methods, our experiments demonstrate the entire pipeline on several challenging assembly tasks.

II. Method

Our method applies to any task that can be simulated using already available assets (e.g., CAD models) or by scanning real-world objects [16,50]. While this paper focuses on assembly tasks for which CAD models are already available for constructing simulation environments, our method applies to many tasks of practical interest. Note that our method requires CAD models only for training and not during deployment.

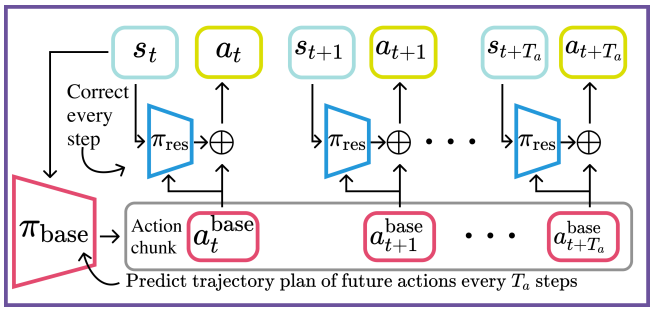


Fig. 2. Overview of `ResiP`. A pretrained chunked base policy predicts an action chunk of T_a future actions. For every timestep, the residual model observes the current state s_t and the predicted base action a_t^{base} and predicts correction.

Our approach from task definition to deployable vision policy consists of three key components (see Fig. 3 for an overview). First, we train a base policy using behavior cloning (BC) on a small set of demonstrations (Sec. II-B) in simulation. Then, we improve this policy's precision by training a residual component with reinforcement learning that makes closed-loop corrections to the base policy's actions (Sec. II-C). Finally, we learn a real-world deployable policy with policy distillation and co-training with a few real-world demonstrations (Sec. II-D).

A. Problem Formulation

We formulate the target task as a discrete-time sequential decision-making problem. At each timestep t, the robot receives an observation $o_t \in \mathcal{O}$ corresponding to the underlying state $s_t \in \mathcal{S}$ of the robot and environment. Given the observation, the policy outputs an action $a_t \in \mathcal{A}$ that is executed in the environment. The action space $\mathcal{A}$ consists of end-effector poses in SE(3) and gripper commands. The state space includes the robot's configuration (end-effector pose, velocity, and gripper state) and the poses of all objects in the scene. Task completion is defined through sparse task completion rewards. We define this reward as an instantaneous signal that provides 1 at the timestep when a pair of objects achieves a pre-defined relative pose corresponding to successful assembly and 0 otherwise, including after the alignment is maintained. See more details about the environments in App. II-A.

B. Base Policy Learning via Behavior Cloning

For each task, we collect a dataset of 50 demonstrations in simulation by teleoperating the robot, $\mathcal{D}_{\text{sim}}: \{\tau_1, ..., \tau_N\}$. Each trajectory, τ, contains system states s_t, and robot actions a_t, i.e., $\tau_i = \{(s_t, a_1), ..., (s_T, a_T)\}$, with T being the trajectory length. We use $\mathcal{D}_{\text{sim}}$ to first train base policy π_{base} with Behavior Cloning (BC), i.e., $\pi_{base} = \text{argmax}_{\pi_{\text{base}}} \mathbb{E}_{(a_t, s_t)\sim\mathcal{D}_{\text{sim}}} [\log \pi_{\text{base}}(a_t|s_t)]$ using the Diffusion Policy (DP) architecture [19], which serves as the starting point for Reinforcement Learning (RL). Consistent with state-of-the-art in BC training [6,19], we use action chunks

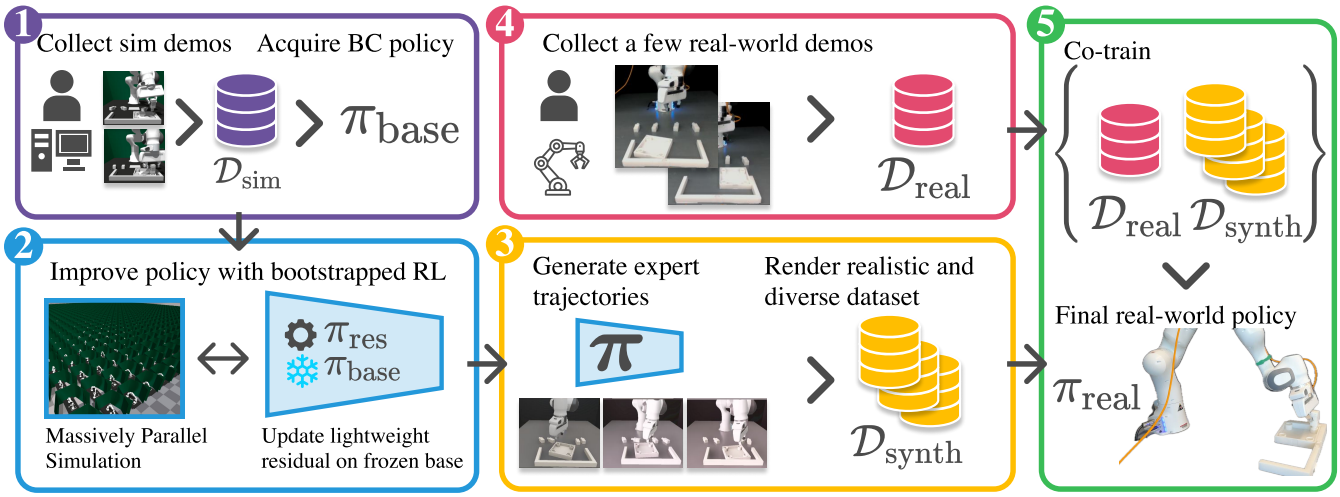


Fig. 3. Sim-to-real pipeline. (1) Beginning with a policy trained with BC in simulation, (2) we train residual policies with RL and sparse rewards. (3) We then distill the resulting behaviors into a policy operating from RGB images. (4) By combining synthetic data with a small set of real demonstrations, (5) we deploy RGB-based policies in the real world.

that predict a set of multiple future actions instead of a single action at every timestep. This chunking approach effectively reduces the task horizon, enabling better performance on long-horizon tasks, but as we show (Sec. IV-B), it sacrifices the ability to make closed-loop corrections. We denote the length of future action sequences predicted by the policy as T_a, the output as $\mathbf{a_t} = [a_t^{\text{base}}, ..., a_{t+T_a}^{\text{base}}]$. When predicting an action chunk $\mathbf{a_t}$ of length T_a, we only execute a subset $[a_t^{\text{base}}, ..., a_{t+T_{\text{exec}}}^{\text{base}}]$, with execution horizon $T_{\text{exec}} \leq T_a$. Similar to [19], we use $T_{\text{exec}} = 8$, and $T_a = 32$ instead of their 16 as we empirically found it performs better.

C. Reactive Control via `ResiP`

Given the initial chunked base policy π_{base} obtained by BC described above, we want to improve the policy to overcome the issues of distribution shift and the lack of reactivity. One way to mitigate the adverse effects of distribution shifts is to fine-tune the BC-trained policy with RL [6]–[16]. We side-step the complications of direct RL fine-tuning of action-chunked policies (see discussion in Sec. II-A) by training a residual [42]–[49] Gaussian Multi-Layer Perceptron (MLP) [51] policy π_{res} using PPO [52]. Our key design choice is that while this residual framework can address distribution shift through either chunk-level or timestep-level corrections (see Sec. III-D.4 for analysis), the latter enables reactivity helpful for precise manipulation.

For each timestep $i = 0, ..., T_a - 1$ within the base policy's action chunk, we form the residual's observation by concatenating the full simulation state (robot proprioceptive information and object poses) with the base policy's predicted action, $s_{t+i}^{\text{res}} = [s_{t+i}, a^{\text{base}} t + i]$. The residual policy then produces a corrective action $a_{t+i}^{\text{res}} \sim \pi_{\text{res}}(\cdot | s_{t+i}^{\text{res}})$ that modifies the base action: $a_{t+i} = a_{t+i}^{\text{base}} + a_{t+i}^{\text{res}}$. The resulting fine-tuned policy is a combination of the pre-trained BC policy π_{base} and the correction policy π_{res}, denoted $\boldsymbol{\pi}$. As demonstrated in Sec. IV-B.3, this per-timestep correction capability is crucial for achieving reliable performance in precision-critical tasks.

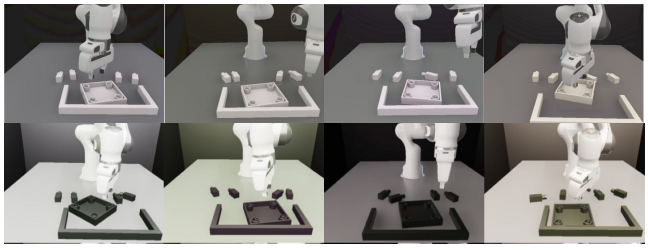


Fig. 4. Photorealistic rendering and domain randomization for sim-to-real transfer. In the first row, we show our nominal experiment setting of rendering parts in the original white color while varying the position, brightness, and hue of the lighting in the scene. In the bottom row, we also introduce variations in the part colors to see how that can help the final policy adapt to unseen part appearances without collecting any additional data.

During training, as in any RL fine-tuning, one needs to adjust the exploration noise to provide sufficient stochasticity to discover new behaviors while maintaining enough precision for task success (see App. X-C for detailed analyses of design choices). The residual policy uses orthogonal initialization [53] with a small gain factor on the final layer, ensuring uncorrelated initial corrections centered around zero—a design choice that aligns with our assumption that the base policy's errors are unbiased in expectation. Since most task failures stem from slight imprecisions in the base policy's actions, this architecture naturally leads to learning small, targeted corrections around the base policy's behavior.

The policy observation for simulation training, $\mathcal{S}$ contains the 6 DoF end-effector pose $\mathbf{T}$, spatial velocity $\mathbf{V}$, and gripper width w_{g}, along with the 6-DoF poses of all the parts in the environment $\{\mathbf{T}^{\text{part}_i}\}_{i=1}^{\text{num\_parts}}$.

D. Sim-to-Real Transfer

We treat π_{res} trained in simulation using privileged state information, $\mathcal{S}$, as a teacher policy π_{teacher} to distill into a student policy π_{student} that takes as input sensory observations

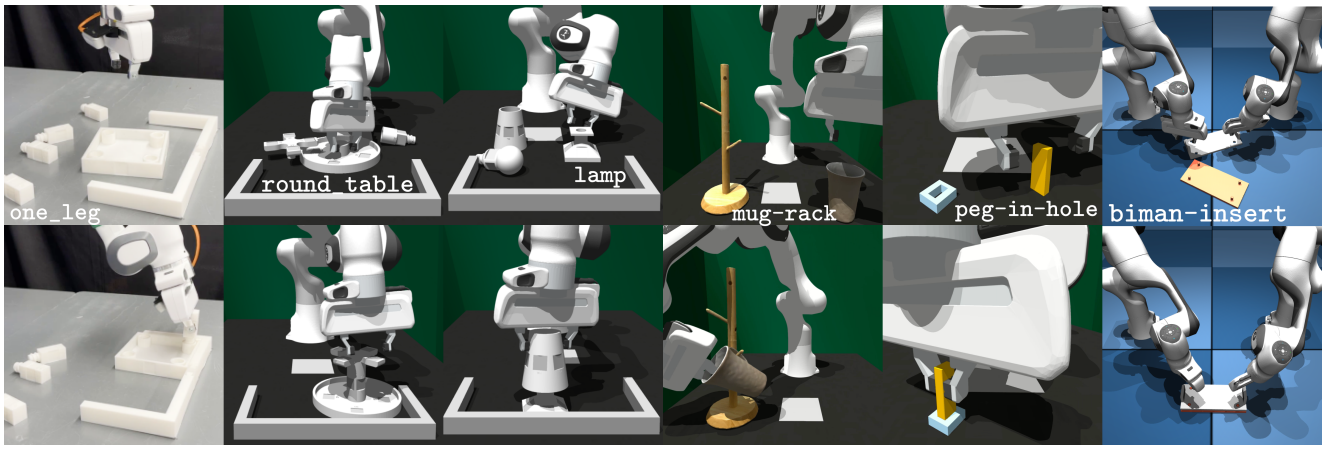


Fig. 5. Initial and goal states for our 6 tasks. The first three tasks, `one_leg`, `round_table`, and `lamp`, are from the FurnitureBench [5] task suite. `mug-rack` we created by scanning objects and importing them into the simulation. `peg-in-hole` is from the Factory [54] task suite. `biman-insert` we created using the simulation-based teleoperation system DART [55]. Our task suite exhibits diverse challenges like long horizons, tight tolerances, bimanual control, and multi-modal success criteria.

and can therefore be deployed in the real world [16,56]–[58]. The real world observation space $\mathcal{O}$ contains the robot end-effector pose $\mathbf{T} \in \mathrm{SE}(3)$, robot end-effector spatial velocity $\mathbf{V} \in \mathbb{R}^6$, the gripper width w_{g}, and RGB images from a fixed front-view camera ($I^{\text{front}} \in \mathbb{R}^{h \times w \times 3}$) and a wrist-mounted camera ($I^{\text{wrist}} \in \mathbb{R}^{h \times w \times 3}$), each with uncalibrated camera poses.

For teacher-student distillation, we collect a dataset of successful trajectories $\mathcal{D}_{\text{synth}} = \{\tau_{\text{synth},1}, ..., \tau_{\text{synth},N}\}$, $\tau_{\text{synth},i} = \{(s_1, a_1), ..., (s_T, a_T)\}$, from $\pi_{teacher}$ initialized with a diverse set of initial object poses. To bridge the sim-to-real gap, we convert the trajectory dataset with only state information ($\mathcal{S}$) into trajectories with sensory observations ($\mathcal{O}$) by re-rendering the trajectories into realistic-looking image observations (see Fig. 4 for examples). This rendering process allows us to introduce visual variations—including object colors, textures, lighting conditions, and camera perspectives—that are not easily obtainable with standard image augmentations or real-world data collection (see App. III for examples and details). We denote this rendered dataset as $\mathcal{D}_{\text{synth-render}}$.

π_{student} is represented with a Diffusion Policy architecture that uses ResNet18 [59] vision encoder pre-trained on robotic manipulation data [60] (see App. I-A.3 for implementation details). To minimize the sim-to-real gap, we combine a small set of real-world demonstrations $\mathcal{D}_{\text{real}}$ (10-40 trajectories) collected directly on the robot with synthetic rendered trajectories from $\mathcal{D}_{\text{synth-render}}$ (approximately 400 demonstrations) [16]. This synthetic-to-real data ratio provides increased domain coverage while not washing out real-world demonstrations. The real demonstrations contain only RGB observations and no ground-truth pose information. This combined dataset $\mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{synth-render}}$ is used to train the student policy π_{student} with the same BC loss as in the simulation-based experiments.

III. Experimental Setup

A. Tasks and Environment

We evaluate our method on a set of challenging assembly tasks of `one_leg`, `round_table`, and `lamp` from the FurnitureBench [5] task suite, the `peg-in-hole` task from [54], a custom mug-hanging task we call `mug-rack`, and a custom bimanual precise insertion task we call `biman-insert`. Examples of all tasks are shown in Fig. 5. Visualizations of initial state distributions and detailed task descriptions are provided in App. II, and task rollouts can be seen on the website.

Multi-part tight-tolerance assembly interactions are simulated using the SDF-based collision geometry representations provided as part of the Factory [54] extension of Nvidia's IsaacGym simulator [61], with demos collected using a SpaceMouse. The `biman-insert` task is implemented using the MuJoCo simulator [62], with the demos provided using the DART teleoperation system from [55].

We define a simple, sparse task-completion reward set to 1 for each task when a pair of parts has been fully assembled and 0 otherwise. For example, in the `lamp` task, the policy receives two binary rewards: the first when the bulb is fully screwed in and the second when the lamp shade is successfully placed. Some of our tasks are long horizon (up to ∼750-1000 steps at 10Hz) and require sequencing of behaviors such as 6-Degree-of-Freedom (DoF) grasping, reorientation, insertion, and screwing (see Fig. 1; Left).

B. System Configuration

The policy operates at 10Hz on a 7 Degrees-of-Freedom (DoF) Franka Emika Panda robot arm for all tasks but `biman-insert`, which operates at 50Hz on two Franka Panda arms (i.e., 14 DoF). The action space consists of the desired end-effector pose $\mathbf{T}^{\text{des}} \in \mathrm{SE}(3)$ (i.e., both position and orientation) and a binary gripper command for opening/closing the parallel-jaw gripper. These desired end-effector poses are converted to joint position targets using

	Methods	one_leg		round_table		lamp		mug-rack	peg-in-hole	biman-insert
		Low	Med	Low	Med	Low	Med	Low	Low	Low
BC	MLP-S	0	0	0	0	0	0	0	2	0
BC	MLP-C	45	10	5	2	8	1	21	2	7
BC	DP	**54**	**26**	**12**	4	7	2	**26**	**5**	**33**
RL	PPO-C	70	28	38	6	32	2	23	4	30
RL	IDQL	57	27	18	3	11	1	31	3	40
RL	ResiP (ours)	**98**	**76**	**94**	**77**	**97**	**70**	**88**	**99**	**93**

TABLE I. Success rates (percentage of successful completions over 1024 evaluation episodes using each method's best-performing checkpoint) for different policy architectures across our task suite. **Top:** Baseline BC methods, where Diffusion Policies (DP) generally outperform MLPs with chunking (MLP-C), while MLPs without chunking (MLP-S) are unable to solve all tasks except peg-in-hole. **Bottom:** RL-based methods, where our proposed residual policy (ResiP) provides significant improvements in success rates over the base BC policy, improvements not seen from the alternative RL methods.

differential inverse kinematics [63], then tracked using a low-level PD controller running at 1KHz with manually specified stiffness and dampening parameters lightly tuned to balance compliance with accurate trajectory tracking.

C. Evaluation protocol

1) Primary Evaluation: We evaluate all methods by executing the policy from an initial state sampled from the same initial state distribution used for collecting demonstrations. We follow the default randomization protocol used in FurnitureBench [5]: at the beginning of each episode, objects are randomly offset from their nominal positions by $\Delta x, \Delta y \in [-1.5, 1.5]$ cm in the horizontal plane (with fixed height z) for low randomization settings and by $\Delta x, \Delta y \in [-5, 5]$ cm for medium randomization. For the U-shaped fixture used to stabilize parts during assembly (the three-sided white "wall" surrounding the parts in the left-most three columns in Fig. 5), we apply additional offsets of $\Delta x, \Delta y \in [-2, 2]$ cm and $\Delta x, \Delta y \in [-4, 4]$ cm from its nominal position for low and medium settings respectively.

We define task success as achieving the target geometric alignment between pairs of parts, as specified by the task reward function in Sec. II-A. For each method, we report the success rate calculated over 1024 evaluation episodes using the best-performing checkpoint, as our goal is to demonstrate the potential capabilities of each architectural approach rather than average performance across training runs.

2) Robustness to Dynamic Disturbances: To further evaluate robustness to dynamic disturbances, we also perform a version of the above evaluation that includes random force perturbations during policy execution. At each timestep, we randomly sample 1% of the objects in the environment and apply a perturbation corresponding to the same randomization described above for initialization. These perturbations simulate unexpected contact forces and dynamic disturbances not seen during training. We evaluate each method's performance degradation under these conditions by comparing success rates with/without perturbations across 1024 rollouts.

3) Real-World Evaluation Protocol: For sim-to-real transfer, we use IsaacSim [64] to render photorealistic trajectories of our simulation data, as it provides better rendering capabilities than IsaacGym [61]. We evaluate policies across a grid of initial object and obstacle poses for real-world experiments that match our low randomization simulation setting ($\Delta x, \Delta y \in [-1.5, 1.5]$ cm). We perform 10 trials for each method and ensure that each method is tested on the same initial object states. Success criteria remain consistent with our simulation experiments: achieving target geometric alignment between assembly parts.

D. Baselines and Ablations

We evaluate our method against several baselines to analyze the importance of action chunking, policy architecture, and closed-loop control learned with Reinforcement Learning (RL). Implementation details and hyperparameters are provided in App. I.

1) Behavior Cloning Baselines: We first evaluate the impact of action chunking by comparing a standard Behavior Cloning (BC) approach using an MLP (MLP-S) with a version that predicts action chunks (MLP-C). We find the Diffusion Policy architecture [19] provides the strongest BC performance and use it as both our primary baseline and the foundation for ResiP.

2) Distribution Shift Analysis: To assess the impact of distribution shifts, we implement **DP-DAgger** [1], which iteratively queries an expert policy to gather corrective demonstrations in states visited by the learned policy. Our experiments use ResiP as the expert. The DAgger policy is trained with the same BC loss and architecture as the nominal DP policy.

3) Reinforcement Learning Comparisons: Building on our BC policies, we compare two approaches for RL fine-tuning. First, we evaluate PPO fine-tuning of our chunked MLP policy (PPO-C) [52], treating each action chunk as a single concatenated action. For our diffusion-based policy, we implement IDQL [65], where multiple action chunks are sampled from the diffusion policy and selected based on learned Q-values using the on-policy method of [66].

4) Closed-Loop Control Ablation: To study the importance of per-timestep corrections, we implement **ResiP-C**, a variant of our method that predicts residual corrections at the same frequency as the base policy's action chunks. While the standard ResiP observes the current state and predicts a correction for each timestep, ResiP-C observes

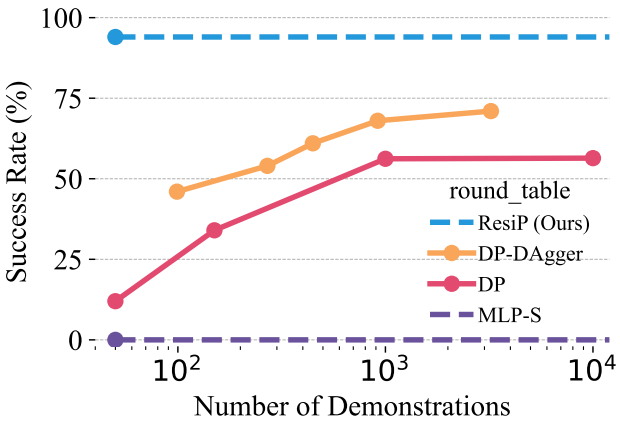


Fig. 6. Scaling behavior comparison on the harder `round_table` task mirrors the trends observed for `one_leg` (Fig. 1 (right)), but with lower overall performance. Just as in `one_leg`, increasing the number of demonstrations for BC training leads to diminishing returns, saturating at 56% success rate. While DAgger's online data collection performs better, it also plateaus at 71%, well below `ResiP`'s 94% success rate. This consistent pattern across tasks, with lower saturation points for the more complex tasks, highlights the limitations of offline approaches and the benefits of learning closed-loop control online.

the current state and all actions in the predicted chunk and predicts a correction of all actions in the chunk. `ResiP-C` uses the same online PPO training procedure as `ResiP`. This chunked correction makes the learning problem more challenging as the residual policy predicts corrections for future timesteps without access to the intermediate states. See Sec. X-A for details.

5) Real-World Baselines: We compare the real-world performance of two policies: (1) **Real-Only**: Diffusion policies trained exclusively on real-world demonstrations $\mathcal{D}_{\text{real}}$, using either 10 or 40 demonstrations, e.g., denoted 10 Real-Only. (2) **Real+Sim**: Following our sim-to-real approach described in Sec. II-D, we combine the same real-world demonstrations with our synthetic rendered dataset, e.g., denoted 10 Real+Sim.

IV. Experimental Results

Our experimental evaluation focuses on three key aspects. First, we analyze how augmenting chunked Behavior Cloning (BC) policy with closed-loop residual Reinforcement Learning (RL) enables reliable execution of precision-critical manipulation tasks (Sec. IV-A). Second, through ablation studies, we identify design choices crucial for `ResiP`'s performance gains (Sec. IV-B). Finally, we evaluate our method on physical robot hardware (Sec. IV-C), demonstrating improved real-world performance through teacher-student distillation while analyzing distillation bottlenecks.

A. Augmenting Trajectory Planning with Reactive Control

In simulation experiments, we evaluate the fundamental limitations of state-of-the-art BC methods on precision-critical tasks. Using a suite of 6 assembly tasks and 50

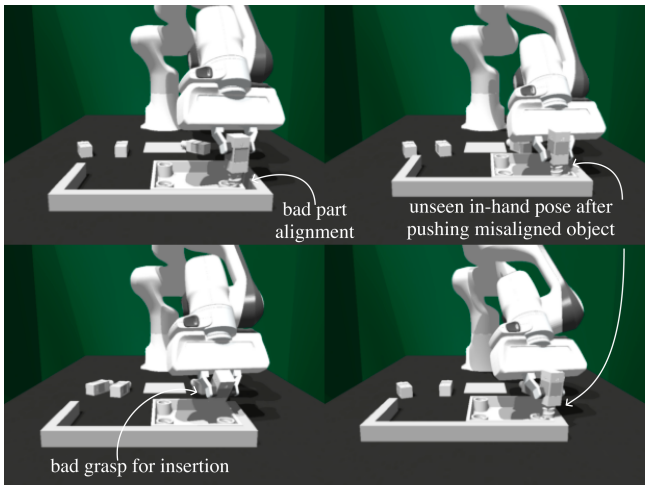


Fig. 7. Examples of the common failures of our DP policies in tasks requiring precise part alignments. These failures can be mitigated by small corrections, making our residual framework, `ResiP`, well-suited to improve the reliability of the nominal DP policy.

demonstrations per task, we find that the Diffusion Policy (DP) architecture struggles with high-precision requirements. Using the `round_table` task as an example, DP achieves a 12% success rate with 50 demonstrations. Increasing the number of demonstrations naturally improves performance. However, as shown in Fig. 6, scaling to 10,000 demonstrations only improves performance to 56% on the `round_table` task. Another potential solution is addressing the distribution shift through online data collection. While our DAgger implementation shows improved performance over pure BC, it plateaus at 71% success on the `round_table` task, still falling significantly short of expert-level performance (similar scaling behavior is observed for the `one_leg` task, see Fig. 1 (Right)).

Our residual learning approach, `ResiP`, addresses these limitations with closed-loop control learned with RL. Starting from the same 50 demonstrations, `ResiP` achieves 94% on `round_table` (up from 12%) and 98% success on `one_leg` (up from 54%), showing significant gains over the alternatives. The most dramatic improvement is seen in the `peg-in-hole` task, where success rates increase from 5% to 99%, highlighting the particular effectiveness of our method when precise local corrections are the primary challenge. Tab. I shows comparisons across our task suite, demonstrating `ResiP`'s consistent advantages over both BC baselines and alternative RL methods.

1) Analyzing Failure Modes: To understand where the large performance improvements stem from, we analyzed failure modes of the DP policies, shown in Fig. 7 and Fig. 1 (Left). In the low randomization setting, we observe that DP's failures primarily arise from small imprecisions: a common error is pushing the leg down before achieving perfect alignment with the hole. Consequently, the object's pose shifts slightly in the gripper, causing an out-of-distribution grasp pose. The residual policy reliably corrects these errors through minimal adjustments: performing small sideways

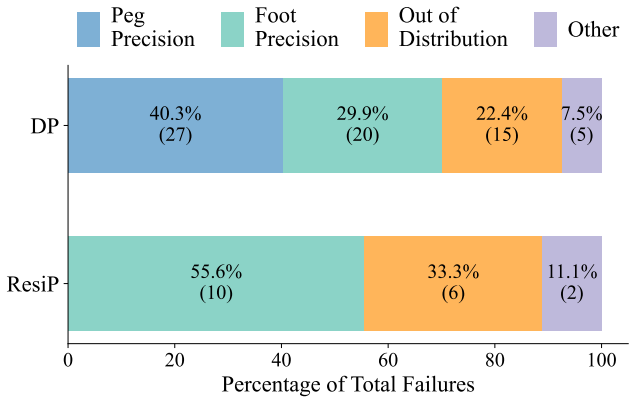


Fig. 8. A qualitative analysis of the failure modes between the pre-trained BC policy and the fine-tuned residual policy for the `round_table` task on medium randomness. For the pre-trained policy, a large portion of the failures (40%) relates to the table leg's insertion precision. In the fine-tuned policy, this share dropped to 0%.

translations while avoiding premature downward force, only allowing insertion once proper alignment is achieved. We also find that the residual policy makes subtle improvements to initial grasps, enabling more precise downstream alignment between the grasped object and the receptacle. Based on these observations, we posit that `ResiP`'s significant performance improvements stem from its ability to provide small adjustments in crucial moments—a natural fit for precision assembly tasks where minor misalignments often cause failure. While the base DP policy struggles with precise alignments and insertions, the learned residual component maintains proper alignment through minimal corrections.

However, the performance saturates at 70%-77% in the medium randomization settings, with parts initialized up to $\pm$5cm from nominal positions. This performance drop aligns with the intuition of the residual policy as improving precision and reactivity primarily as a local correction mechanism. When parts are placed far from their nominal positions, the DP policy may generate trajectories that deviate too far for the local correction mechanism to correct, leading to failures that are not easily corrected. Even if the failures can be corrected, the resulting states may be out-of-distribution for the DP policy, making corrections infeasible.

To better understand the failure modes of DP and `ResiP` in this higher randomness setting, we manually analyzed 75 randomly sampled trajectories for the `round_table` task on medium randomness (see the website for videos). We categorized the observed failures into four primary types:

1) 'Peg Precision' failures that occur during picking, inserting, or screwing operations with the table leg
2) 'Foot Precision' failures during picking, inserting, or screwing the table foot
3) 'Out-of-Distribution' failures when parts are unreachable by the base policy
4) 'Other' failures are attributed to contact-modeling issues in simulation (e.g., object penetration causing unrealistic accelerations) or timeout conditions.

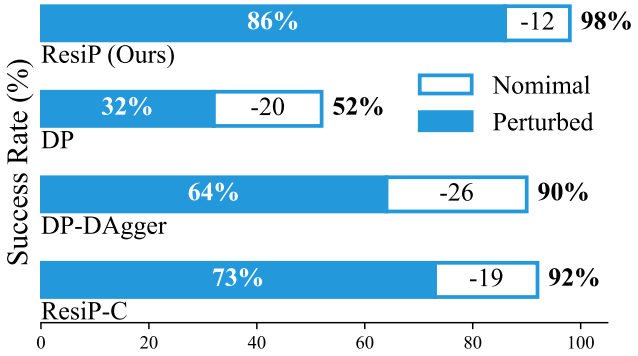


Fig. 9. Performance robustness when introducing random force perturbations during evaluation. At each timestep, we randomly apply forces to the objects in the environment, simulating unexpected disturbances. Our method (`ResiP`), which makes per-timestep corrections, shows greater resilience with only a 12% performance drop compared to nominal conditions. In contrast, chunk-based methods (DP, DP-DAgger, and `ResiP`-C) see larger drops of 19-26%. This difference in robustness highlights the value of closed-loop control for handling dynamic disturbances not seen during training. All methods were evaluated across 1024 episodes, both with and without perturbations.

Our qualitative analysis of these failure modes and recovery behaviors for the `round_table` task, shown in Fig. 8, shows that `ResiP` significantly improves reliability during precise insertion phases compared to the baseline approach, and eliminates the precision errors for "Peg Precision" entirely but that the 'Out-of-Distribution' increases in share.

Surprisingly, `ResiP` also leads to the emergence of qualitatively different behaviors compared to the nominal DP policy. We refer to the website's uncut videos of rollouts for both policies. In these videos, we observe that `ResiP` has discovered alternative grasping strategies not present in the demonstration data—while demonstrations only showed grasping the table foot from above its center, `ResiP` learned to grasp from the side when the center was unreachable. Additionally, `ResiP` exhibits more decisive recovery behaviors, making deliberate adjustments that lead to successful task completion where the nominal BC policy would ineffectively oscillate back and forth.

B. What drives performance of `ResiP`?

This section investigates different aspects of `ResiP` that improve task performance: (1) training stability and sample efficiency across RL methods, (2) the impact of addressing distribution shift through online data collection, (3) the benefits of closed-loop control compared to chunk-based execution, and (4) robustness to dynamic perturbations.

1) Performance and Training Characteristics of RL Methods: Our residual learning approach, `ResiP`, significantly improves upon baseline methods across all tasks. Using 50 demonstrations per task, where `ResiP` achieved 98% on the `one_leg` task (Tab. I), the alternative RL approaches show more limited improvement of 70% for PPO-C and 57% for

Training data	Corner		Grasp		Insert		Screw		Complete	
	Part	Obs	Part	Obs	Part	Obs	Part	Obs	Part	Obs
10 Real-Only	5/10	5/10	5/10	7/10	2/10	3/10	0/10	2/10	0/10	2/10
10 Real+Sim	9/10	9/10	7/10	8/10	0/10	3/10	0/10	3/10	0/10	3/10
40 Real-Only	10/10	8/10	9/10	8/10	6/10	3/10	2/10	3/10	2/10	3/10
40 Real+Sim	10/10	10/10	9/10	10/10	6/10	7/10	5/10	6/10	5/10	6/10

TABLE II. Comparing the effect of combining real-world demonstrations with simulation trajectories from our RL-trained residual policies. Co-training with real and synthetic data improves motion quality and success rate on the `one_leg` task.

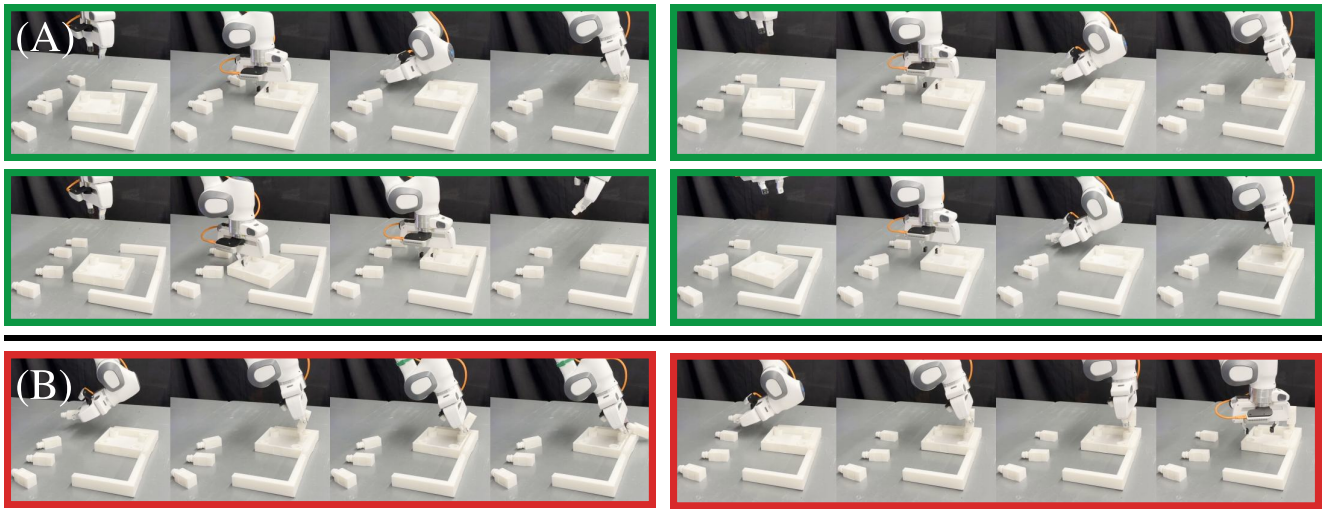


Fig. 10. **(A)** Examples of successful real world assembly from RGB. Co-training with simulation data reduces jerkiness and improves insertion robustness by containing a higher diversity of part poses and insertion locations (see Table II). **(B)** Example failure: difficulty adjusting the insertion angle/position when grasps lead to unseen in-hand part poses.

IDQL. Similar trends hold for all tasks as seen in Tab. I.

We also see distinct training characteristics across methods. PPO-C, which directly fine-tunes the chunked MLP policy, exhibits training instability and requires careful KL regularization to avoid collapse [67]–[69]. IDQL faces a different challenge: even with maximum stochasticity in the denoising process, the base DP model produces actions with insufficient variance for effective exploration, limiting the potential for Q-learning-based improvement.

In contrast, `ResiP` demonstrates stable training behavior. Its architecture naturally constrains corrections to be local adjustments to the base policy's absolute pose predictions rather than operating in the full workspace coordinate frame. This local prediction scope, combined with the residual network's small parameter count, prevents the large policy updates that typically destabilize deep RL training [70,71]. This stability provides consistent performance across hyperparameters (see Fig. 28 in App. X-C) and enables multiple gradient steps per collected trajectory, improving sample efficiency.

2) Impact of Distribution Shift: To probe the impact of mitigating distribution shift, we compared the baseline DP against DP-DAgger trained using online data collection via DAgger [72] as described in Sec. III-D. While DP-DAgger significantly outperforms the baseline DP (see yellow vs. red lines in Fig. 1 (right) and Fig. 6), it still trails `ResiP` (blue line) by 9% and 23% for the `one_leg` and `round_table` tasks, respectively. The remaining performance gap suggests that reducing distribution shift with online data collection does not fully explain `ResiP`'s performance benefits.

3) Impact of Closed-Loop Control: To quantify the benefits of per-timestep corrections, we compared `ResiP` against `ResiP`-C, which predicts corrections at the chunk level as described in Sec. III-D.4. Our experiments reveal that chunk-level corrections lead to significantly slower learning progress, with `ResiP`-C requiring significantly more environment interactions to achieve comparable performance (Fig. 26 in App. X-A). Moreover, even with extended training, `ResiP`-C's performance saturates at 92% compared to `ResiP`'s 98% on the `one_leg` task. This performance gap suggests that the ability to make frequent corrections accelerates learning and enables higher terminal performance.

4) Robustness to Dynamic Perturbations: To further evaluate the benefits of closed-loop control, we compare the performance degradation of different methods under dynamic disturbances as described in Sec. III-C. As shown in Fig. 9, methods using chunk-based execution—DP, DP-DAgger, and `ResiP`-C—experience significant performance drops of 19-26 percentage points under these perturbations. In contrast,

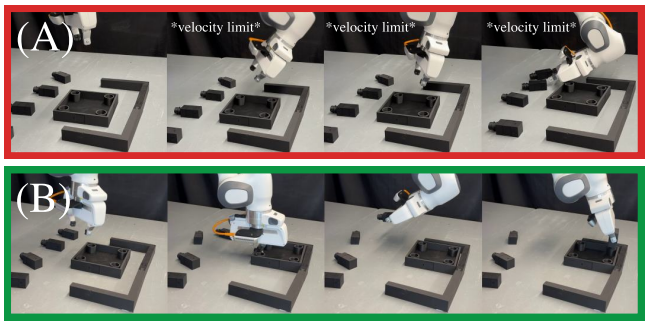


Fig. 11. (A) When changing the part appearances from white to black, the policy trained on real data only (Real-Only) seizes to function. The behavior becomes erratic, and all trials ended with the robot hitting a velocity limit. **(B)** When mixing in synthetic data with more diverse colors (see Fig. 4), the policy (Real+Sim) regains the ability to complete the task, though with lower performance than for white parts.

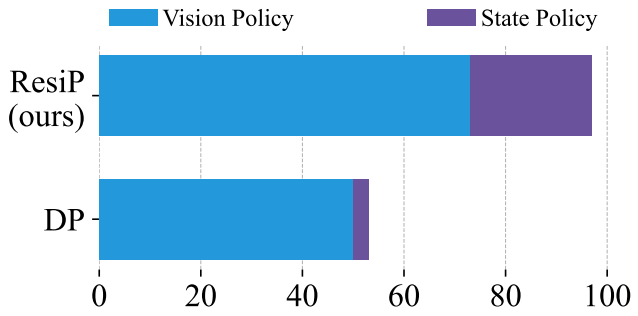


Fig. 12. A comparison of performance between state and vision policies in policy distillation. The Diffusion Policy (DP) baseline achieves similar performance with both state and vision inputs when trained directly on the same 50 demonstrations (left), indicating that the vision modality does not inherently limit performance. However, we observe a significant performance gap when distilling our state-based `ResiP` policy into a vision-based policy (right). This consistent performance gap suggests the challenge lies in the distillation process rather than the vision modality, as direct vision-based training shows no such degradation.

`ResiP`, with its ability to make per-timestep corrections, maintains a more robust performance with a 12% drop. This significant difference in robustness (7-14 percentage points) provides further evidence for the value of closed-loop control in precise manipulation tasks.

C. Real-World Deployment

1) Real-World Performance: In our sim-to-real experiments, following the protocol described in Sec. III-C, we find that the Real+Sim policies achieve significantly higher success rates (50-60%) compared to Real-Only baselines (20-30%) on the `one_leg` task (Tab. II). Qualitatively, the co-trained policy (Real+Sim) exhibits smoother behavior with fewer erratic movements that exceed robot hardware limits. Fig. 10 shows typical behaviors: Row (A) shows successful assembly sequences, while Row (B) demonstrates the primary failure mode—imprecise alignment before release, mirroring challenges observed in simulation.

To assess robustness to visual variations, we tested changing part colors from the white used in data collection to an unseen black. Unsurprisingly, the Real-Only policy fails catastrophically, triggering velocity limits on every trial, as shown in Fig. 11 (A). In contrast, Real+Sim, trained with the same real-world data but with color-randomized synthetic data (see Fig. 4; bottom), maintains basic functionality though with reduced performance compared to the original color scheme. While highlighting the benefits of synthetic data, this also indicates opportunities for improving sim-to-real transfer. See App. V-B for more detailed analysis.

2) Understanding Performance Limitations: To understand the gap between simulation and real-world performance, we hypothesize three potential limiting factors: (1) the change from state- to vision-based observations, (2) the sim-to-real gap, and (3) the policy distillation process. Comparative experiments between image and state-based students show similar performance gaps from the teacher's 98% success rate (Fig. 12), ruling out the change in observation modality as the primary bottleneck. Furthermore, our scaling analysis shows that even increasing synthetic trajectories from 10k to 100k only marginally improves success rates from 78% to 80% (Fig. 1; Right), indicating a fundamental limitation in the policy distillation process itself rather than purely sim-to-real challenges. These findings motivate future investigations into better approaches for vision-based policy distillation, particularly focusing on online learning and closed-loop control. See App. V-A for detailed analysis.

V. Related Works

A. Training diffusion models with reinforcement learning

A fundamental challenge in applying RL to diffusion models is that the final action probabilities are not directly accessible due to the iterative nature of the denoising process, making policy gradient methods difficult to apply. Recent work has explored various approaches to combining diffusion models with RL [40,41,73]–[75]. Some approaches cast diffusion de-noising as a Markov Decision Process [40,74], enabling preference-aligned image generation with policy gradient RL, but suffer from training instability. While [41] introduced more stable direct diffusion policy fine-tuning, their method remains architecture-specific and lacks closed-loop control.

Other approaches include Q-function-based importance sampling [65], advantage weighted regression [76], and return-conditioned supervised learning [17,18,77]. Some methods augment the denoising objective with Q-function maximization [78] or iteratively update the dataset using Q-functions [79]. However, these approaches mainly enable better *extraction* or stitching of existing behaviors rather than learning new, *corrective* behaviors. Recent work on training diffusion policies from scratch [75] uses complex mechanisms like unsupervised clustering and Q-learning ensembles for multi-modal behavior discovery.

Beyond the iterative denoising challenge, modern BC approaches introduce additional complications through action chunking, where policies predict sequences of multiple future actions. While this improves BC performance, it creates significant challenges for RL fine-tuning by expanding the action space—for instance, chunks of 8 actions result in an 8-fold increase in action dimensionality. This issue affects most modern BC architectures like ACT [6,15], as they rely on action chunking. Policy gradient methods struggle with such high-dimensional action spaces, particularly when applied to deep neural networks [80]. Furthermore, recent work shows that RL fine-tuning of large pre-trained models can lead to forgetting of pre-training capabilities [81]. Our method avoids these problems by keeping the base policy frozen and training only a small residual model, which preserves the pre-trained capabilities and enables stable policy gradient training with closed-loop control.

B. Residual learning in robotics

Learning corrective residual components in conjunction with learned or non-learned "base" models has been widely successful in robotics. Common frameworks include learning residual policies that correct for errors made by a nominal behavior policy [42]–[49] and combining learned components to correct for inaccuracies in analytical models for physical dynamics [82]–[84] or sensor observations [85]. Unlike prior residual policy learning approaches, our method uniquely combines RL-based training with running the base at a lower frequency, with the residual providing corrections at every control step.

Residual policies have been used in insertion applications [86], and recent work has applied residual policy learning to the same FurnitureBench task suite we study in this paper [87]. Their approach uses the residual component to model online human-provided corrections via supervised learning, whereas we train our residual policy from scratch with RL using task rewards in simulation.

VI. Discussion

The local nature of our residual policies is designed to complement rather than replace the trajectory planning capabilities of the base policy. While this design enables precise corrections, our approach still relies on the base policy for macro-level behaviors. As such, our proposed method struggles in regimes with very high initial scene randomness, as both the base policies and actions produced via RL exploration struggle to deal with out-of-support initial part poses. Our imitation learning scaling analyses were conducted using a dataset from an RL expert, not human demonstrations. These two demonstration sources will likely have different distributions, which may change the analyses. However, acquiring 100k demonstrations from a human demonstrator was not feasible in the present work. Furthermore, despite showcasing the advantage of incorporating simulation data, sim-to-real for vision-based policies still presents a challenge. There remains a performance gap in both teacher-student distillation and sim-to-real distribution shifts. Future investigations may include better sim-to-real transfer techniques, exploration mechanisms for discovering how to correct large-scale execution errors, tractable interactive learning for real-world policy distillation, and incorporating inductive biases (like [88]) that help generalize to broader initial state distributions.

Acknowledgments

This work was partly supported by the Sony Research Award, the US Government, and the Hyundai Motor Company. The computations in this paper were run on the FASRC cluster, supported by the FAS Division of Science Research Computing Group at Harvard University, and on the MIT Supercloud [89]. Experiment tracking and model checkpoint storage were provided by Weights and Biases.

Author Contributions

LA led the project and implemented most of the code and training infrastructure, including the main residual PPO implementation, and is the primary author.

AS helped conceive of and advise on project goals, led deployment on real hardware and sim-to-real rendering, and helped write the paper.

IS led the implementation of reinforcement learning baselines, helped debug residual PPO implementation, and helped write the paper.

MT provided valuable insights, recommendations, and discussions on reinforcement learning fine-tuning and sim-to-real transfer.

PA advised the project and provided valuable feedback on project framing, contributions, and writing.

References

[1] S. Ross, G. Gordon, and D. 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*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[2] K. Kimble, K. Van Wyk, J. Falco, E. Messina, Y. Sun, M. Shibata, W. Uemura, and Y. Yokokohji, "Benchmarking protocols for evaluating small parts robotic assembly systems," *IEEE robotics and automation letters*, vol. 5, no. 2, pp. 883–889, 2020.

[3] F. Suárez-Ruiz and Q.-C. Pham, "A framework for fine robotic assembly," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 421–426.

[4] Y. Lee, E. S. Hu, and J. J. Lim, "Ikea furniture assembly environment for long-horizon complex manipulation tasks," in *2021 ieee international conference on robotics and automation (icra)*. IEEE, 2021, pp. 6343–6349.

[5] M. Heo, Y. Lee, D. L. Kaist, and J. J. Lim, "FurnitureBench: Reproducible Real-World Benchmark for Long-Horizon Complex Manipulation," *RSS 2023*, 2023, arXiv: 2305.12821v1. [Online]. Available: https://clvrai.com/furniture-bench

[6] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[7] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[8] S. Schaal, "Learning from Demonstration," in *Advances in Neural Information Processing Systems*, vol. 9. MIT Press, 1996. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1996/hash/68d13cf26c4b4f4f932e3eff990093ba-Abstract.html

[9] ——, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[10] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa, "Imitation learning for locomotion and manipulation," in *2007 7th IEEE-RAS international conference on humanoid robots*. IEEE, 2007, pp. 392–397.

[11] P. Agrawal, *Computational sensorimotor learning*. University of California, Berkeley, 2018.

[12] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 5628–5635.

[13] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[14] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[15] M. Drolet, S. Stepputtis, S. Kailas, A. Jain, J. Peters, S. Schaal, and H. B. Amor, "A comparison of imitation learning algorithms for bimanual manipulation," *IEEE Robotics and Automation Letters*, 2024.

[16] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal, "Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation," *arXiv preprint arXiv:2403.03949*, 2024.

[17] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with Diffusion for Flexible Behavior Synthesis," Dec. 2022, arXiv:2205.09991 [cs]. [Online]. Available: http://arxiv.org/abs/2205.09991

[18] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is Conditional Generative Modeling all you need for Decision-Making?" Jul. 2023, arXiv:2211.15657 [cs]. [Online]. Available: http://arxiv.org/abs/2211.15657

[19] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," Jun. 2023, arXiv:2303.04137 [cs]. [Online]. Available: http://arxiv.org/abs/2303.04137

[20] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, and S. Devlin, "Imitating Human Behaviour with Diffusion Models," Mar. 2023, arXiv:2301.10677 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2301.10677

[21] L. Lai, A. Z. Huang, and S. J. Gershman, "Action chunking as policy compression," *PsyArXiv*, 2022.

[22] L. Ankile, A. Simeonov, I. Shenfeld, and P. Agrawal, "Juicer: Data-efficient imitation learning for robotic assembly," *arXiv preprint arXiv:2404.03729*, 2024.

[23] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[24] A. Yu, G. Yang, R. Choi, Y. Ravan, J. Leonard, and P. Isola, "Lucidsim: Learning agile visual locomotion from generated images," in *8th Annual Conference on Robot Learning*, 2024.

[25] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, S. K. S. Ghasemipour, C. Finn, and A. Wahid, "ALOHA unleashed: A simple recipe for robot dexterity," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=gvdXE7ikHI

[26] Y. Liu, J. I. Hamid, A. Xie, Y. Lee, M. Du, and C. Finn, "Bidirectional decoding: Improving action chunking via closed-loop resampling," *arXiv preprint arXiv:2408.17355*, 2024.

[27] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[28] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1612–1619, 2022.

[29] Y. Lu, K. Hausman, Y. Chebotar, M. Yan, E. Jang, A. Herzog, T. Xiao, A. Irpan, M. Khansari, D. Kalashnikov *et al.*, "Aw-opt: Learning robotic skills with imitation andreinforcement at scale," in *Conference on Robot Learning*. PMLR, 2022, pp. 1078–1088.

[30] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *International Conference on Machine Learning*. PMLR, 2023, pp. 1577–1594.

[31] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, vol. 9, 1996.

[32] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[33] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine, "Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[34] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao *et al.*, "Jump-start reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 34 556–34 583.

[35] H. Hu, S. Mirchandani, and D. Sadigh, "Imitation bootstrapped reinforcement learning," *arXiv preprint arXiv:2311.02198*, 2023.

[36] Q. Zheng, A. Zhang, and A. Grover, "Online decision transformer," in *international conference on machine learning*. PMLR, 2022, pp. 27 042–27 059.

[37] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010.

[38] R. Ramrakhya, D. Batra, E. Wijmans, and A. Das, "Pirlnav: Pretraining with imitation and rl finetuning for objectnav," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 896–17 906.

[39] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine, "Parrot: Data-driven behavioral priors for reinforcement learning," *arXiv preprint arXiv:2011.10024*, 2020.

[40] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine, "Training diffusion models with reinforcement learning," *arXiv preprint arXiv:2305.13301*, 2023.

[41] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz, "Diffusion policy policy optimization," *arXiv preprint arXiv:2409.00588*, 2024.

[42] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[43] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[44] M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid, "Residual reinforcement learning from demonstrations," *arXiv preprint arXiv:2106.08050*, 2021.

[45] T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy, "Residual learning from demonstration: Adapting dmps for contact-rich manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4488–4495, 2022.

[46] J. Carvalho, D. Koert, M. Daniv, and J. Peters, "Residual robot learning for object-centric probabilistic movement primitives," *arXiv preprint arXiv:2203.03918*, 2022.

[47] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in neural information processing systems*, vol. 35, pp. 22 955–22 968, 2022.

[48] S. Haldar, J. Pari, A. Rai, and L. Pinto, "Teach a robot to fish: Versatile imitation from one minute of demonstrations," *arXiv preprint arXiv:2303.01497*, 2023.

[49] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto, "Behavior generation with latent actions," *arXiv preprint arXiv:2403.03181*, 2024.

[50] M. T. Villasevil, A. Jain, V. Macha, J. Yuan, L. L. Ankile, A. Simeonov, P. Agrawal, and A. Gupta, "Scaling robot-learning by crowdsourcing simulation environments," in *RSS 2024 Workshop: Data Generation for Robotics*, 2024.

[51] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[52] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, arXiv:1707.06347 [cs]. [Online]. Available: http://arxiv.org/abs/1707.06347

[53] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.

[54] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, A. Handa, and D. Fox, "Factory: Fast Contact for Robotic Assembly," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[55] Y. Park, J. S. Bhatia, L. Ankile, and P. Agrawal, "Dexhub and dart: Towards internet scale robot data collection," *arXiv preprint arXiv:2411.02214*, 2024.

[56] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[57] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[58] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand reorientation of novel and complex object shapes," *Science Robotics*, vol. 8, no. 84, p. eadc9244, 2023.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[60] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3M: A Universal Visual Representation for Robot Manipulation," Nov. 2022, arXiv:2203.12601 [cs]. [Online]. Available: http://arxiv.org/abs/2203.12601

[61] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

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

[63] R. Tedrake, *Robotic Manipulation*. Course Notes for MIT 6.421, 2024. [Online]. Available: http://manipulation.mit.edu

[64] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[65] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine, "IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies," May 2023, arXiv:2304.10573 [cs]. [Online]. Available: http://arxiv.org/abs/2304.10573

[66] S. Han, I. Shenfeld, A. Srivastava, Y. Kim, and P. Agrawal, "Value augmented sampling for language model alignment and personalization," 2024.

[67] T. G. Rudner, C. Lu, M. A. Osborne, Y. Gal, and Y. Teh, "On pathologies in kl-regularized reinforcement learning from expert demonstrations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 376–28 389, 2021.

[68] A. Galashov, S. M. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, and N. Heess, "Information asymmetry in kl-regularized rl," *arXiv preprint arXiv:1905.01240*, 2019.

[69] J. Schulman, X. Chen, and P. Abbeel, "Equivalence between policy gradients and soft q-learning," *arXiv preprint arXiv:1704.06440*, 2017.

[70] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[71] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[72] S. Ross, G. Gordon, and D. 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*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[73] Y. Fan and K. Lee, "Optimizing ddpm sampling with shortcut fine-tuning," *arXiv preprint arXiv:2301.13362*, 2023.

[74] Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. Lee, "Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models," 2023.

[75] Z. Li, R. Krohn, T. Chen, A. Ajay, P. Agrawal, and G. Chalvatzaki, "Learning multimodal behaviors from scratch with diffusion policy gradient," 2024.

[76] W. Goo and S. Niekum, "Know your boundaries: The necessity of explicit behavioral cloning in offline rl," *arXiv preprint arXiv:2206.00695*, 2022.

[77] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[78] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," *arXiv preprint arXiv:2208.06193*, 2022.

[79] L. Yang, Z. Huang, F. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin, "Policy representation via diffusion probability model for reinforcement learning," 2023.

[80] H. Hu, S. Mirchandani, and D. Sadigh, "Imitation Bootstrapped Reinforcement Learning," Nov. 2023, arXiv:2311.02198 [cs]. [Online]. Available: http://arxiv.org/abs/2311.02198

[81] M. Wołczyk, B. Cupiał, M. Ostaszewski, M. Bortkiewicz, M. Zajkac, R. Pascanu, Ł. Kuciński, and P. Miłoś, "Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem," *arXiv preprint arXiv:2402.02868*, 2024.

[82] A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez, "Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3066–3073.

[83] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects from sensory data," *The International Journal of Robotics Research*, vol. 41, no. 8, pp. 778–797, 2022.

[84] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[85] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[86] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5548–5555.

[87] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei, "Transic: Sim-to-real policy transfer by learning from online correction," *arXiv preprint arXiv:2405.10315*, 2024.

[88] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann, "Neural descriptor fields: Se (3)-equivariant object representations for manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6394–6400.

[89] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas, "Interactive Supercomputing on 40,000 Cores for Machine Learning and Data Analysis," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, Sep. 2018, pp. 1–6, iSSN: 2377-6943. [Online]. Available: https://ieeexplore.ieee.org/document/8547629

[90] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5745–5753.

[91] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[92] J. Levinson, C. Esteves, K. Chen, N. Snavely, A. Kanazawa, A. Rostamizadeh, and A. Makadia, "An analysis of svd for deep rotation estimation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 554–22 565, 2020.

[93] A. R. Geist, J. Frey, M. Zobro, A. Levina, and G. Martius, "Learning with 3d rotations, a hitchhiker's guide to so(3)," 2024.

[94] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-Conditioned Imitation Learning using Score-based Diffusion Policies," Jun. 2023, arXiv:2304.02532 [cs]. [Online]. Available: http://arxiv.org/abs/2304.02532

[95] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang, "Industreal: Transferring contact-rich assembly tasks from simulation to reality," in *Robotics: Science and Systems*, 2023.

[96] X. Zhang, S. Jin, C. Wang, X. Zhu, and M. Tomizuka, "Learning insertion primitives with discrete-continuous hybrid action space for robotic assembly tasks," in *2022 International conference on robotics and automation (ICRA)*. IEEE, 2022, pp. 9881–9887.

[97] O. Spector and D. Di Castro, "Insertionnet-a scalable solution for insertion," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5509–5516, 2021.

[98] Y. Tian, K. D. Willis, B. A. Omari, J. Luo, P. Ma, Y. Li, F. Javid, E. Gu, J. Jacob, S. Sueda *et al.*, "Asap: Automated sequence planning for complex robotic assembly with physical feasibility," *arXiv preprint arXiv:2309.16909*, 2023.

[99] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine, "SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning," Jan. 2024, arXiv:2401.16013 [cs]. [Online]. Available: http://arxiv.org/abs/2401.16013

[100] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

APPENDIX I
IMPLEMENTATION DETAILS

A. Training Hyperparameters

1) State-based behavior cloning: We provide a detailed set of hyperparameters used for training. General hyperparameters for all models can be found in Tab. III, while specific hyperparameters for the diffusion models are in Tab. IV, and those for the MLP baseline are in Tab. V.

TABLE III. Training hyperparameters shared for all state-based BC models

Parameter	Value
Control mode	Absolute end-effector pose
Action space dimension	10
Proprioceptive state dimension	16
Orientation Representation	6D [90]
Max LR	10^{-4}
LR Scheduler	Cosine
Warmup steps	500
Weight Decay	10^{-6}
Batch Size	256
Max gradient steps	400k

TABLE IV. State-based diffusion pre-training hyperparameters

Parameter	Value
U-Net Down dims	$[256, 512, 1024]$
Diffusion step embed dim	256
Kernel size	5
N groups	8
Parameter count	66M
Observation Horizon T_o	1
Prediction Horizon T_p	32
Action Horizon T_a	8
DDPM Training Steps	100
DDIM Inference Steps	4

TABLE V. State-based MLP pre-training hyperparameters

Parameter	Value
Residual Blocks	5
Residual Block Width	1024
Layers per block	2
Parameter count	11M
Observation Horizon T_o	1
Prediction Horizon T_p (S / C)	1 / 8
Action Horizon T_a (S / C)	1 / 8

2) State-based reinforcement learning: Below, we list the hyperparameters used for online reinforcement learning fine-tuning. The parameters that all state-based RL methods methods shared are in Tab. VI. Method-specific hyperparameters for training the different methods are in the tables below, direct fine-tuning of the MLP in Tab. VII, online IDQL in Tab. VIII, and the residual policy in Tab. IX. The different methods were tuned independently, but the same hyperparameters were used for all tasks within each method.

TABLE VI. Hyperparameters shared for all online fine-tuning approaches

Parameter	Value
Control mode	Absolute end-effector pose
Action space dimension	10
Proprioceptive state dimension	16
Orientation Representation	6D [90]
Num parallel environments	1024
Max environment steps	500M
Critic hidden size	256
Critic hidden layers	2
Critic activation	ReLU
Critic last layer activation	Linear
Critic last layer bias initialization	0.25
Discount factor	0.999
GAE [91] lambda	0.95
Clip ϵ	0.2
Max gradient norm	1.0
Target KL	0.1
Num mini-batches	1
Episode length, `one_leg`	700
Episode length, `lamp`/`round_table`	1000
Normalize advantage	true

TABLE VII. Hyperparameters for direct fine-tuning of MLP

Parameter	Value
Update epochs	1
Learning rate actor	10^{-4}
Learning rate critic	10^{-4}
Value function loss coefficient	1.0
KL regularization coefficient	0.5
Actor Gaussian initial log st.dev.	-4.0

TABLE VIII. Hyperparameters for training value-augmented diffusion sampling (IDQL)

Parameter	Value
Update epochs	10
Learning rate Q-function	10^{-4}
Learning rate scheduler	Cosine
Num action samples	20
Actor added Gaussian noise, log st.dev.	-4

TABLE IX. Hyperparameters for residual PPO training

Parameter	Value
Residual action scaling factor	0.1
Update epochs	50
Learning rate actor	$3 \cdot 10^{-4}$
Learning rate critic	$5 \cdot 10^{-3}$
Learning rate scheduler	Cosine
Value function loss coefficient	1.0
Actor Gaussian initial log st.dev.	-1.0

3) Image-based real-world distillation: We use a separate set of hyperparameters for real-world experiments, presented in Tab. X. The main difference is that we found in experimentation that the transformer backbone in [19] worked better than the UNet for real-world experiments. These models are also operating from RGB observations instead of privileged states, and we provide parameters for the image augmentations applied to the front camera in Tab. XI and the wrist camera in Tab. XII.

TABLE X. Training hyperparameters for real-world distilled policies

Parameter	Value
Control mode	Absolute end-effector pose
Action space dimension	10
Proprioceptive state dimension	16
Orientation Representation	6D [90]
Max policy LR	10^{-4}
Max encoder LR	10^{-5}
LR Scheduler (both)	Cosine
Policy scheduler warmup steps	1000
Policy scheduler warmup steps	5000
Weight decay	10^{-3}
Batch size	256
Max gradient steps	500k
Image size input	$2 \times 320 \times 240 \times 3$
Image size encoder	$2 \times 224 \times 224 \times 3$
Vision Encoder Model	ResNet18 [59]
Encoder Weights	R3M [60]
Encoder Parameters	2×11 million
Encoder Projection Dim	128
Diffusion backbone architecture	Transformer (similar to [19])
Transformer num layers	8
Transformer num heads	4
Transformer embedding dim	256
Transformer embedding dropout	0.0
Transformer attention dropout	0.3
Transformer causal attention	true

TABLE XI. Parameters for front camera image augmentation

Parameter	Value
Color jitter (all parameters)	0.3
Gaussian blur, kernel size	5
Gaussian blur, sigma	$(0.01, 1.2)$
Random crop area	280×240
Random crop size	224×224
Random erasing, fill value	random
Random erasing, probability	0.2
Random erasing, scale	$(0.02, 0.33)$
Random erasing, ratio	$(0.3, 3.3)$

TABLE XII. Parameters for wrist camera image augmentation

Parameter	Value
Color jitter (all parameters)	0.3
Gaussian blur, kernel size	5
Gaussian blur, sigma	$(0.01, 1.2)$
Random crop	Not used
Image resize	$320 \times 240 \rightarrow 224 \times 224$

B. Action and State-Space Representations

a) Action space: The policies predict 10-dimensional actions consisting of absolute poses in the robot base frame as the actions and a gripper action. In particular, the first 3 dimensions predict the desired end-effector position in the workspace, the next 6 predict the desired orientation using a 6-dimensional representation described below. The final dimension is a gripper action, 1 to command closing gripper and -1 for opening.

b) Proprioceptive state space: The policy receives a 16-dimensional vector containing the current end-effector state and gripper width. In particular, the first 3 dimensions is the current position in the workspace, the next 6 the current orientation in the base frame (the same 6D representation), the next 3 the current positional velocity, the next 3 the current roll, pitch, and yaw angular velocity, and finally the current gripper width.

c) Rotation representation: We use a 6D representation to represent all orientations and rotations for the predicted action, and proprioceptive end-effector pose orientation [90,92]. The poses of the parts in state-based environments are represented with unit quaternions. While this representation contains redundant dimensions, it is continuous, meaning that small changes in orientation lead to small changes in the representation values, which can make learning easier [90,92,93].

This is not generally the case for Euler angles and quaternions. The 6D representation is constructed by taking two arbitrary 3D vectors and performing Gram-Schmidt orthogonalization to obtain a third orthogonal vector to the first two. The resulting three orthogonal vectors form a rotation matrix that represents the orientation. The end-effector rotation angular velocity is still encoded as roll, pitch, and yaw values.

d) Action and state-space normalization: All dimensions of the action, proprioceptive state, and parts pose (for state-based environments), were independently scaled to the range [-1, 1]. That is, we did not handle orientation representations (quaternions/6D [90]) in any particular way. The normalization limits were calculated over the dataset at the start of behavior cloning training. They were stored in the actor with the weights and reused as the normalization limits when training with reinforcement learning. The normalization used here follows the same approach as in previous works such as [19,94]. This normalization method is widely accepted for diffusion models. In [94], the input was standardized to have a mean of 0 and a standard deviation of 1, instead of using min-max scaling to the range of [0, 1]. This approach was not tested in our experiments.

C. Image Augmentation

During training, we apply image augmentation and random cropping to both camera views. Specifically, only the front camera view undergoes random cropping. We also apply color jitter with a hue, contrast, brightness, and saturation set to 0.3. Additionally, we apply Gaussian blur with a kernel size of 5 and sigma between 0.1 and 5 to both camera views.

At inference time, we statically center-crop the front camera image from 320x240 to 224x224 and resize the wrist camera view to the same dimensions. For both the random and center crops, we resized the image to 280x240 to ensure that essential parts of the scene are not cropped out due to excessive movement.

The values mentioned above were chosen based on visual assessment to balance creating adversarial scenarios and keeping essential features discernible. We have included examples of these augmentations below.

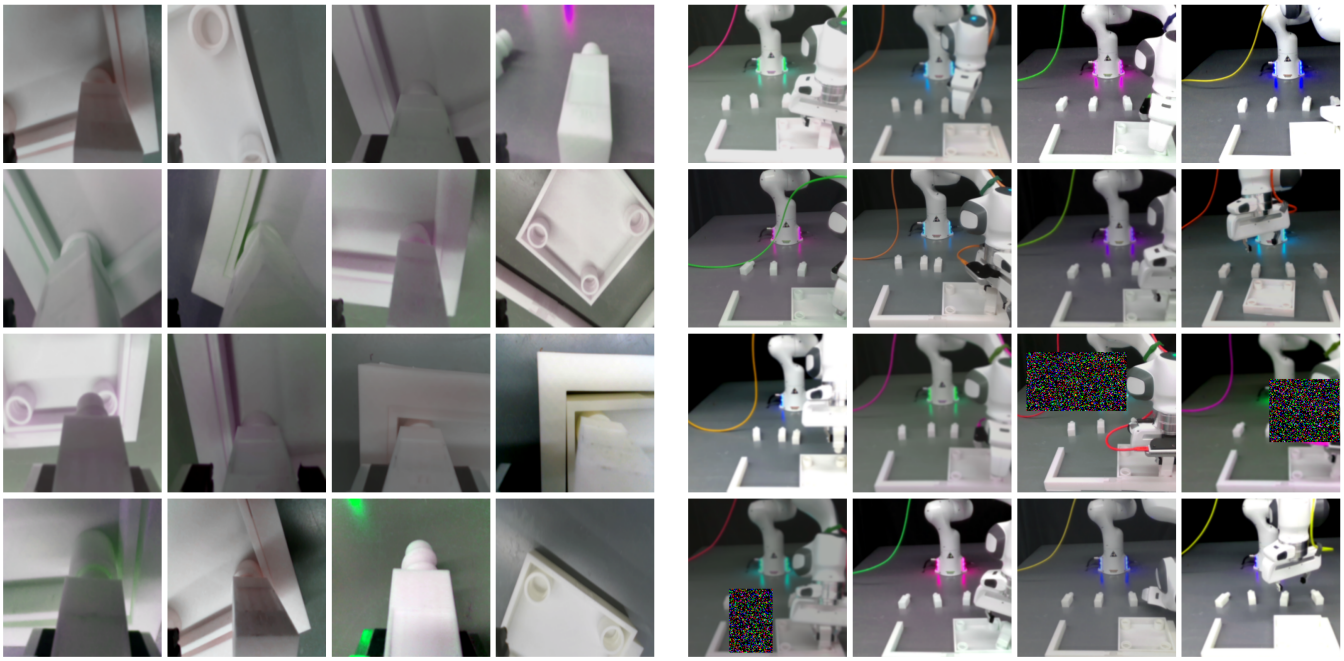


Fig. 13. Left: Examples of augmentations of the wrist camera view, consisting of color jitter and Gaussian blur. **Right:** Examples of augmentations for the front view also consist of color jitter and Gaussian blur augmentations and random cropping.

APPENDIX II
TASKS AND ENVIRONMENT

A. Tasks details and reward signal

1) Furniture assembly tasks: We detail a handful of differentiating properties for each of the three tasks we use in Tab. XIII. `one_leg` involves assembling 2 parts, the tabletop and one of the 4 table legs. The assembly is successful if the relative poses between the parts are close to a predefined assembled relative pose. When this pose is achieved, the environment returns a reward of 1. That is, for the `one_leg` task, the policy received a reward of 1 only at the very end of the episode. For `round_table` and `lamp`, which consists of assembling 3 parts together, the policy receives a reward

signal of 1 for each pair of assembled parts. E.g. for the `lamp` task, when the bulb is fully screwed into the base, the first reward of 1 is received, and the second is received when the shade is correctly placed.

2) Real-to-sim task: `mug-rack`: This task involves the robot picking up a coffee mug and hanging it by the handle on one of two pegs on a rack. See Fig. 14 for task illustration. This task is interesting for two main reasons. First, we don't have any CAD models for the objects. Instead, we used scanned imports of real-world objects (obtained with the ARCode app on the iPhone App Store). Second, the task has inherent multi-modality in that the mug can be hung in one of two ways for each of the two pegs.

The diffusion and residual policy system works well for this task. First, the base diffusion model captures the task's multimodality and sometimes hangs the mug on both pegs. Furthermore, the residual RL procedure keeps this multimodality intact as the base model is frozen.

(a) Example task initialization of the `mug-rack` task.

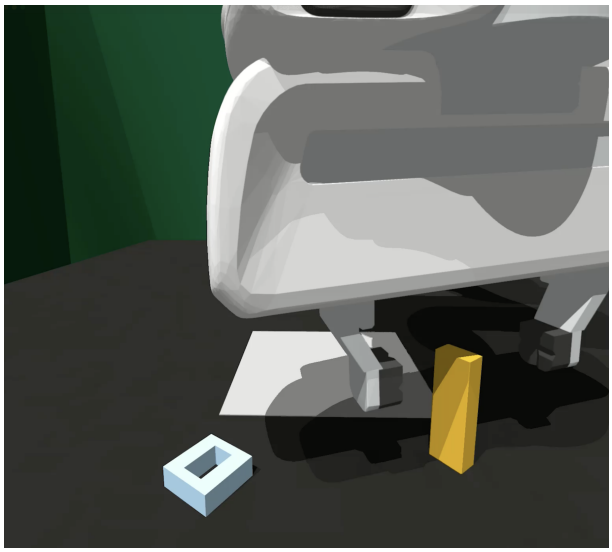
(b) Example of hanging the mug on the lower rack.

(c) Example of hanging the mug on the upper rack.

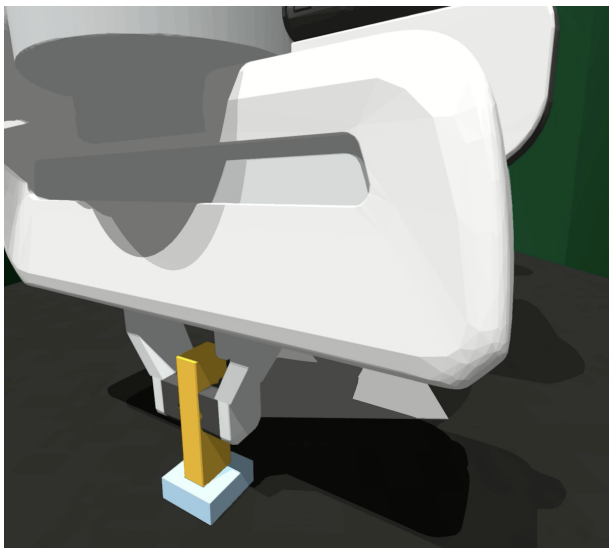
Fig. 14. Overview of the `mug-rack` task to showcase the real-to-sim capabilities one can leverage with our pipeline. This also shows how reward signals can be inferred directly from data instead of being hand-designed. Finally, as the task can be completed in one of several ways, this task also tests the policies' capability to deal with multi-modality.

3) High-precision, Factory task: `peg-in-hole`: To push the limits of precision in simulation, controller, and policy, we pick one of the insertion tasks from the Factory task suite [54], which involves grasping a peg and inserting in a hole with a 0.2mm clearance, i.e., 25x tighter than the FurnitureBench [5] tasks. See Fig. 15 for task illustration.

Our approach also worked out of the box on this task, using the same hyperparameters as for the FurnitureBench tasks. Here, we achieve 5% success rate in pre-training and $\sim$99% in fine-tuning. Good performance at this task is essentially entirely dominated by the ability to locally adjust the peg until it lines up with the hole, and the high final success rate achieved by our approach reflects that the local nature of the corrections learned by our residual policy is well aligned with such task scenarios.



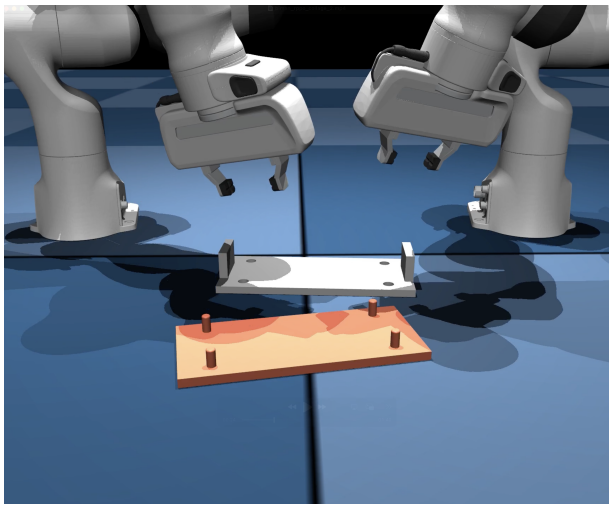
(a) Example task initialization of the `peg-in-hole` task.



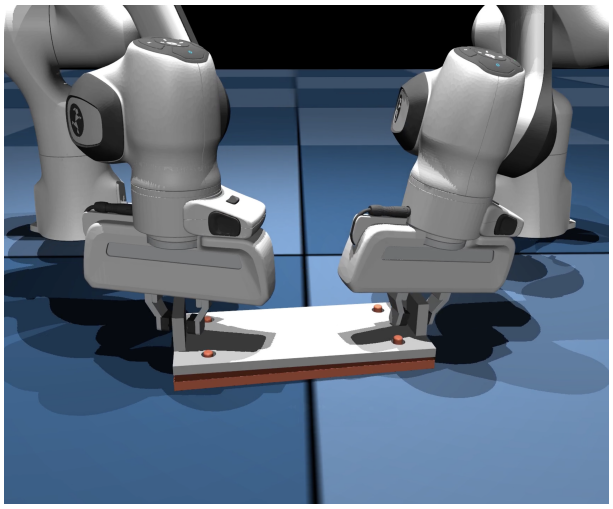
(b) Example of task completion when the peg is fully inserted.

Fig. 15. Overview of the `peg-in-hole` task we add to push the requirement for precision. We find that the pipeline as presented works well with the same hyperparameters used for the furniture tasks.

4) Bimanual, high-precision task: `biman-insert`: To test whether our method, `ResiP`, also works for precise tasks with larger action spaces, we create a simple bimanual industrial assembly task that we term `biman-insert`. See Fig. 16 for an example initial and final state and Fig. 18 for several random initial states. We design the task by creating simple meshes and importing them into the MuJoCo [62] physics engine. We demonstrate the task using the augmented reality-based teleoperation interface DART [55]. All subsequent training uses the same code and hyperparameters as all the other tasks. This task has a relatively short horizon but has a 20-dimensional action space and relatively tight insertion tolerances. We also perform this task at 50 Hz for policy control, showing that our approach is quite general.



(a) Example task initialization of the `biman-insert` task.



(b) Example of task completion when the plate is fully inserted.

Fig. 16. Overview of the `biman-insert` task we add to push the requirement for precision and bimanual coordination. We find that the pipeline as presented works well with the same hyperparameters used for the furniture tasks and that the increased action space poses no problem for mastering the task.

TABLE XIII. Task Attribute Overview

	one_leg	round_table	lamp	mug-rack	peg-in-hole	biman-insert
Mean episode length	∼500	∼700	∼600	∼150	∼200	∼400
# Parts to assemble	2	3	3	2	2	2
Num rewards	1	2	2	1	1	1
Dynamic object	✗	✗	✓	✗	✗	✗
# Precise insertions	1	2	1	0	1	1
# Screwing sequences	1	2	1	0	0	0
Precise grasping	✗	✓	✗	✗	✗	✗
Insertion occlusion	✗	✓	✗	✓	✗	✗
Control frequency	10 Hz	10 Hz	10 Hz	10 Hz	10 Hz	50 Hz
Degrees-of-Freedom	7	7	7	7	7	14

B. Details on randomization scheme

The "low" and "medium" randomness settings we used for data collection and evaluation reflect how much the initial part poses may vary when the environment is reset. We tuned these conditions to mimic the levels of randomness introduced in the original FurnitureBench suite [5]. However, we found that their method of directly sampling random poses often leads to initial part configurations colliding, requiring expensive continued sampling to eventually find an initial layout where all parts do not collide.

Our modified randomization scheme instead initializes parts to a single pre-specified set of feasible configurations. Then, it applies a randomly sampled force and torque to each part (where the force/torque magnitudes are tuned for each part and scaled based on the desired level of randomness). This scheme allows the physics simulation to ensure parts stay out of collision while providing a controlled amount of variation in the initial scene randomness.

The second way we modified the randomization scheme was to randomize the position of the U-shaped obstacle fixture and the parts (the obstacle fixture was always kept in a fixed position in [5]). We reasoned that, for visual sim-to-real without known object poses, we could only imperfectly and approximately align the obstacle location in the simulated and real environment. Rather than attempting to make this alignment perfect, we instead trained policies to cover some range of possible obstacle locations, hoping that the real-world obstacle position would fall within the distribution the policies have seen in simulation. Fig. 17 shows examples of our different randomness levels for each task in simulation.

C. Adjustments to FurnitureBench simulation environments

In addition to our modified force-based method of controlling the initial randomness, we introduced multiple other modifications to the original FurnitureBench environments proposed in [5] to enable the environment to run fast enough to be feasible for online RL training. With these changes, we could run at a total of ∼4000 environment steps per second across 1024 parallel environments. The main changes are listed below:

1) Vectorized reward computation, done check, robot, part, and obstacle resets, and differential inverse kinematics controller.
2) Removed April tags from 3D models to ensure vision policies would not rely on tags to complete the tasks. We tried to align with the original levels of randomness, but only to an approximation.
3) Deactivate camera rendering when running the environment in state-only mode.
4) Correct an issue where the physics was not stepped a sufficient amount of time for sim time to run at 10Hz, and subsequently optimize calls to fetch simulation results, stepping of graphics, and refreshing buffers.
5) Artificially constrained bulb from rolling on the table until robot gripper is nearby as the rolling in the simulator was exaggerated compared to the real-world parts.

APPENDIX III
RGB SIM2REAL TRANSFER

a) Visualization of overlap in action space in real and sim: For data from the simulation to be useful for *increasing* the support of the policy for real-world deployment, we posit that it needs to *cover* the real-world data. We visualize the distributions of actions in the training data in Fig. 19. Since actions are absolute poses in the robot base frame, we can take the x, y, z coordinates for all actions from simulation and real-world demonstration data and plot them. Each of the 3 plots is a different cross-section of the space, i.e., a view from top-down, side, and front. In general, we see that the simulation action distribution is more spread out and mostly covers real-world actions.

b) Visual Domain randomization: In addition to randomizing part poses and the position of the obstacle, we randomize parts of the rendering which is not easily randomized by simple image augmentations, like light placement (changing shadows), camera pose, and individual part colors. See Fig. 20 for examples of front-view images obtained from our domain randomization and re-rendering procedure.

Fig. 17. Examples of initial scene layouts for the tasks from the FurnitureBench task suite [5], `one_leg`, `lamp`, and `round_table`, with different levels of initial part pose and obstacle fixture randomness.

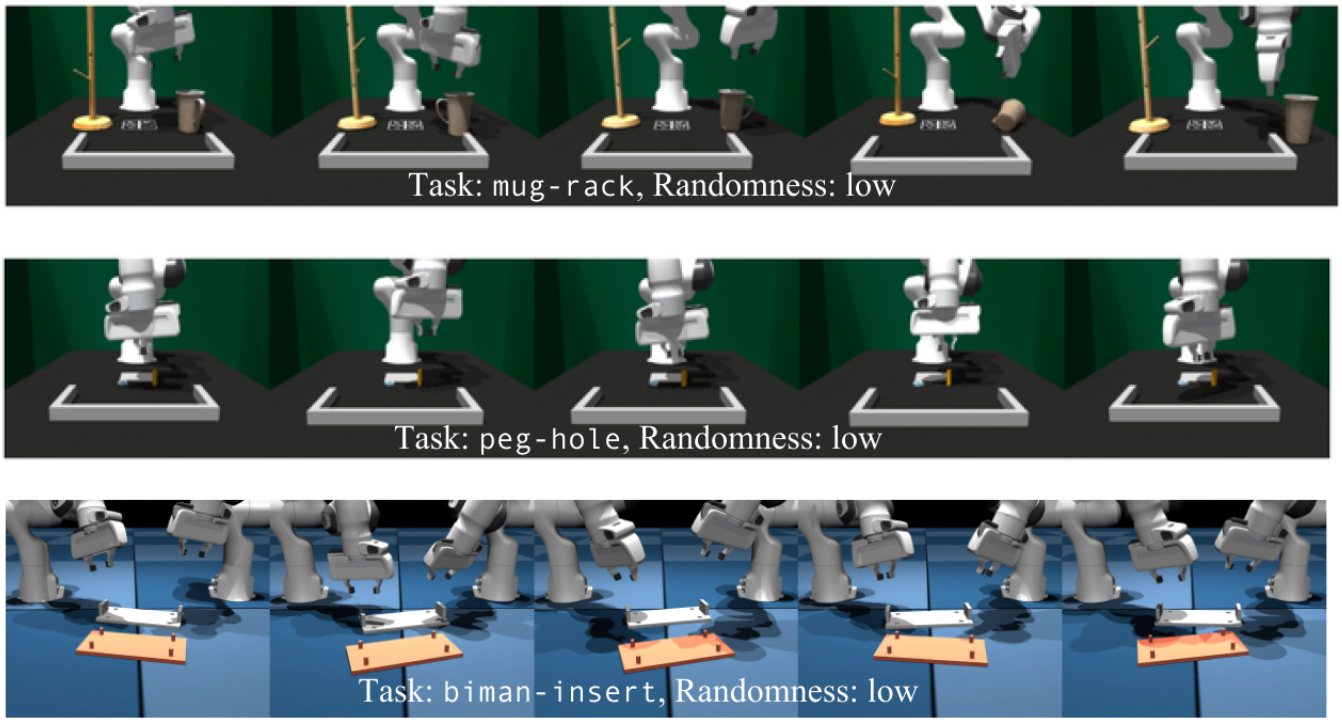


Fig. 18. Examples of initial scene layouts for the 3 non-FurnitureBench tasks, `mug-rack`, `peg-in-hole`, and `biman-insert`, for their default level of randomness.

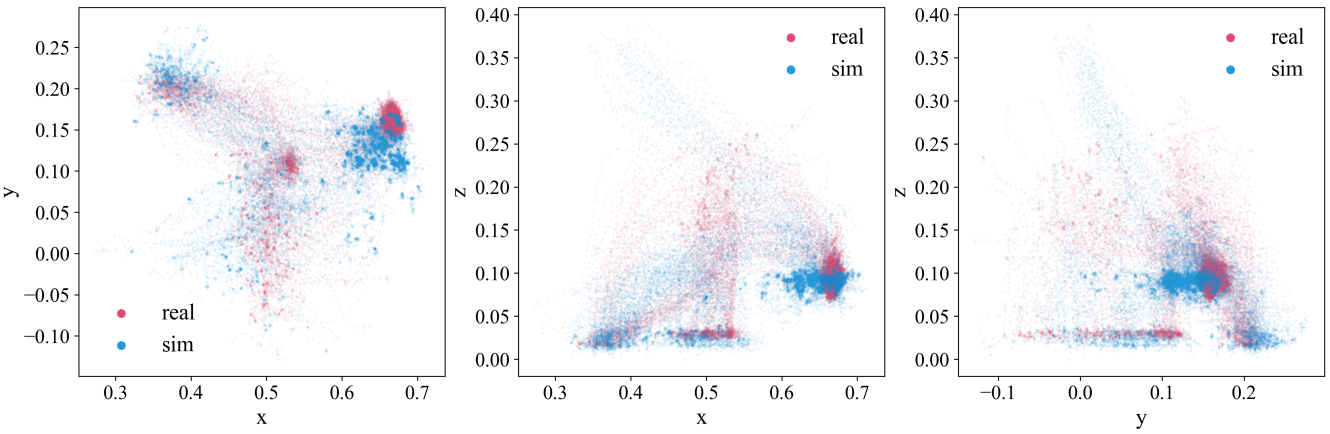


Fig. 19. Plots of the x, y, z action coordinates in the demo datasets for the `one_leg` task in the real world and the simulator. That is, each dot represents one action from one of the 40/50 trajectories. Red is from real-world demos, and blue is from the simulator. **Left:** Top-down view, showing the x, y positions in the workspace visited. In the top right, the insertion point is shown, where we see that the simulator has a wider distribution but could have covered better in the positive y-direction. **Middle:** Side-view of the actions taken in the x, z plane. The insertion point is to the right in the plot; again, we see more spread in the simulation data. **Right:** Front view of the y, z actions.

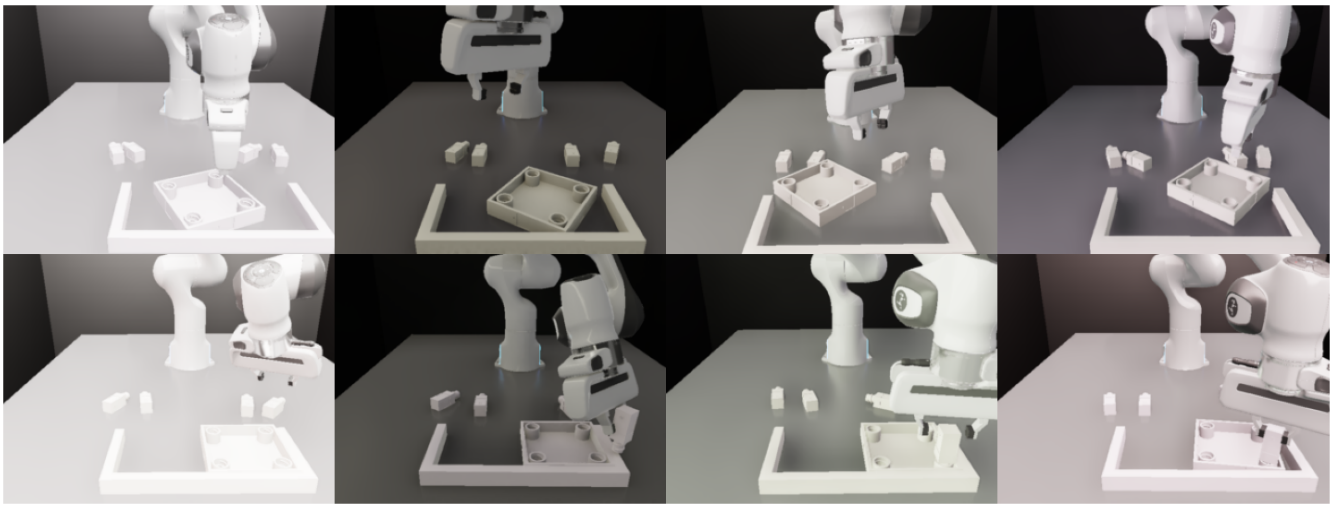


Fig. 20. Examples of the randomization applied when rendering out the simulation trajectories used for co-training for the real-world policies.

APPENDIX IV
VISUALIZATION OF RESIDUAL POLICY ACTIONS

We hypothesize that the strength of the residual policy is that it can operate locally and make corrections to the base action predicted by the pretrained policy operating on the macro scale in the scene. We show an example of this behavior in Fig. 21. Here, we visualize the base action with the red line, the correction predicted by the residual in blue, and the net action of the combined policy in green.

We find that the residual has indeed learned to correct the base policy's actions, which often leads to failure. One common example is for the base policy to be imprecise in the approach to the hole during insertion, pushing down with the peg not aligned with the hole, causing the peg to shift in the gripper, which leads to a grasp-pose unseen in the training data and the policy inevitably fails. The residual policy counteracts the premature push-down and correct the placement towards the hole, improving task success. See video examples of this behavior on the accompanying website: `https://residual-assembly.github.io/`.

APPENDIX V
EXTENDED VISION-BASED RESULTS AND ANALYSES

A. *Performance Impact of Distillation*

Next, we study how the quantity and quality of data generated by a `ResiP` policy impact the performance of vision-based student policies in real-world evaluations. We generate this data by collecting successful trajectories from the `ResiP` teacher across varied initial states and rendering corresponding camera observations. A vision-based student policy—which shares the same architecture as the teacher but includes an additional image encoder—distilled from ∼1,000 teacher trajectories reached 73% success on `one_leg`, outperforming the 50% achieved by training the vision policy directly on human demos (see Fig. 12). However, we observe a performance gap between the RL-trained `ResiP` teacher (98%) and the distilled vision-based student (73%), even after performance saturates with additional data. To investigate whether this gap stems from the change to visual input, we compared distillation performance between image-based and state-based students using the same number of trajectories. Their comparable performance suggests that the modality shift is not the primary cause of the performance gap. While DAgger-style online distillation might improve performance, we focused on offline distillation as it better reflects real-world deployment constraints.

Therefore, we examine the impact of the distillation dataset size. Here, we scale up the number of state-based rollouts from the trained RL policy and distill these to a state-based student. In Fig. 1 (Right), we observe that performance increases with more data, from 78% success rate at 10k trajectories to 80% at 100k trajectories, though not reaching the teacher policy's 98% success rate. The same trend is evident in Fig. 6 (though the saturation occurs earlier). These results demonstrate how simulation-based distillation can complement existing training approaches by enabling the rapid generation of large-scale synthetic datasets at a minimal cost. Beyond the data volume advantages, our RL teacher exhibits qualitatively different behaviors, such as faster movements and improved corrective actions, suggesting that this synthetic data captures valuable task strategies that could be expensive or impractical to demonstrate manually.

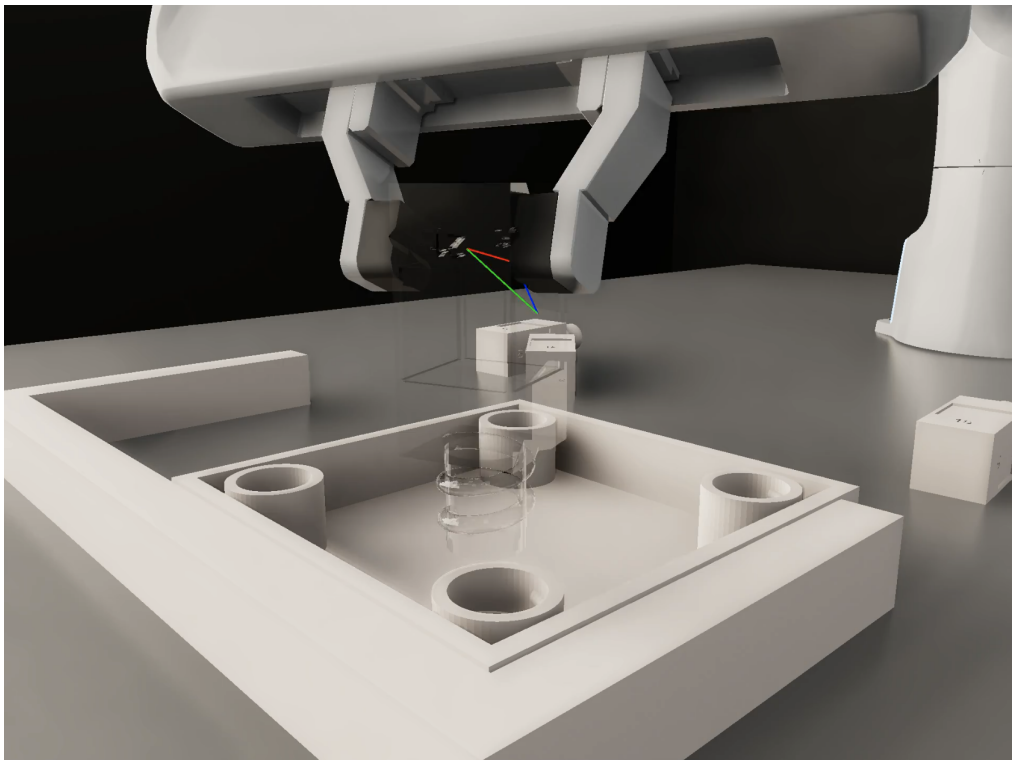


Fig. 21. Visualization of the effect of the residual policy during insertion, the phase requiring the most precision. The red line shows the action commanded by the base policy. The blue is the correction predicted by the residual, and the green is the net action. The residual learns to correct actions that typically lead to failure.

While DAgger [1] demonstrates strong sample efficiency - achieving better performance than BC with just ∼10k gradient steps (800 rollouts)—it requires an expert policy for online data collection. Although we could theoretically apply DAgger in simulation using our RL-trained expert policy (as demonstrated by [16] for point cloud inputs), this introduces significant complexity to the pipeline and its effectiveness for RGB image-based distillation remains an open question for future work. Instead, we demonstrate that a simple approach combining offline rendering of synthetic trajectories with real-world co-training can achieve reasonable performance.

B. Real-World Evaluation

Finally, we evaluate the real-world performance of a sim-to-real policy trained on a mixture of a few (10/40) real-world demonstrations (**Real+Sim**) and simulation data generated by the trained residual RL policy. We compare the co-trained policy to a baseline model trained only on real-world demonstrations (**Real-Only**). We compare the success rates achieved by each policy on two sets of 10 trials for the `one_leg` task. In the first set, we randomize part poses, while in the second set, we randomize obstacle poses (i.e., insertion location in the workspace).

We compare the co-trained policy to a baseline model trained only on real-world demonstrations. We define an evaluation grid spanning the same ranges as the low randomization setting from the FurnitureBench simulation environment. We evaluate each policy on two sets of 10 trials for the `one_leg` task, with grid points sampling either part poses or obstacle poses (i.e., insertion location in the workspace). Each method is evaluated on the same set of grid positions to ensure fair comparison.

The results in Tab. II show that incorporating simulation data improves real-world performance (e.g., increasing task completion rate from 20-30% to 50-60%). Qualitatively, the sim-to-real policy exhibits smoother behavior and makes fewer erratic movements that might exceed the robot's physical limits. Fig. 10 illustrates this through example trajectories: Row (A) shows successful executions where the robot completes the full assembly sequence, while Row (B) demonstrates the most common failure mode where, despite successfully grasping and transporting the parts, the policy fails to achieve precise alignment between the table leg and the hole before releasing. This misalignment failure pattern mirrors what we observe in the simulation, suggesting consistent challenges in achieving the required precision for insertion tasks.

To further probe the robustness conferred by training in simulation, we created a task variation where the part colors are changed from black to white. When rolling out the policy trained on real demos of white parts, **Real-Only**, the robot exhibited erratic behavior that caused the hardware to reach velocity limits on every trial we ran, as shown in Fig. 11 (A). When including synthetic data rendered with parts in black, the resulting policy (**Real+Sim-DR**) can perform the task again

(see Fig. 11). The resulting performance was still inferior to the performance on white parts, which motivates further work on closing the sim-to-real gap.

C. Quantitative Results Failure Mode Breakdown

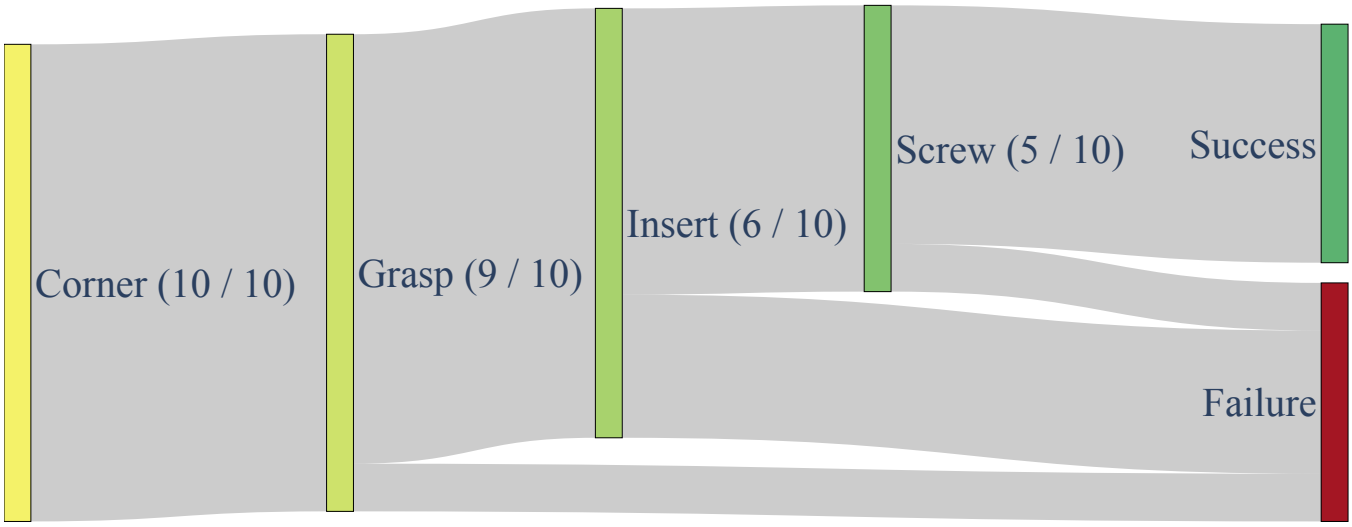


Fig. 22. Sankey diagram for the success rate and failure points for the real-world rollouts with 40 real and 350 simulation demos.

The diagram in Fig. 22 shows how successful and failed completion of individual sub-skills along the `one_leg` task amount to our overall final success rates reported in Tab. II (bottom row, corresponding to "40 real + 350 sim" with random initial part poses and a fixed obstacle pose).

D. Extension of Pipeline to Unseen Settings

Here, we conduct further qualitative experiments to evaluate whether our simulation-based co-training pipeline can make policies more robust to real-world parts with visual appearances that are unseen in real world demos. To test this, we 3D printed the same set of parts used in the `one_leg` task in black, and rolled out various policies on these black parts (rather than the white-colored parts used throughout our other experiments). This setting is especially relevant in industrial domains where parts can come in a variety of colors to which the assembly system must be invariant (e.g., the same piece of real-world furniture usually comes in many colors).

When deploying the policy trained on the same 40 demos as in the main experiment, which only had *white*, the policy cannot come close to completing the task. The behavior is highly erratic and triggered the velocity limits of the Franka on every trial we ran. We compare this baseline policy trained on differently colored parts to a policy co-trained on both real and synthetic data from simulation. However, when creating the synthetic dataset for this test, we added in additional randomization of part color, with an emphasis on black or gray colors in this case, as shown in Fig. 24. When we co-train a policy on a mix of the same real-world demos containing *only* white parts as before, with a dataset of 400 synthetic demos with *varying* part colors, the resulting policy can complete the task, as illustrated in Fig. 24 (and even when it fails at the entire task sequence, the predicted motions are much more reasonable than the erratic policy which has overfit to real-world parts of a specific color).

For example videos, please see the accompanying website: `https://residual-assembly.github.io/`. We note, however, that the resulting policy is considerably less reliable than the corresponding policy rolled out with white parts, which illustrates that there is still a meaningful sim2real gap.

APPENDIX VI
EXPANDED RELATED WORK

a) Learning robotic assembly skills: Robotic assembly has been used by many as a problem setting for various behavior learning techniques [45,95]–[98]. Enabling assembly that involves multi-skill sequencing (e.g., fixturing → grasping → insertion → screwing) directly from RGB images has remained challenging, especially *without* explicitly defining sub-skill-specific boundaries and supervision. Concurrent work [87] explores a similar framework to ours on FurnitureBench tasks [5], but instead supervises learned policies on a per-skill basis and incorporates 3D point clouds. IndustReal [95] also leverages RL in simulation to train high-precision skills for tight-tolerance part insertion in the real world. However, they train their RL

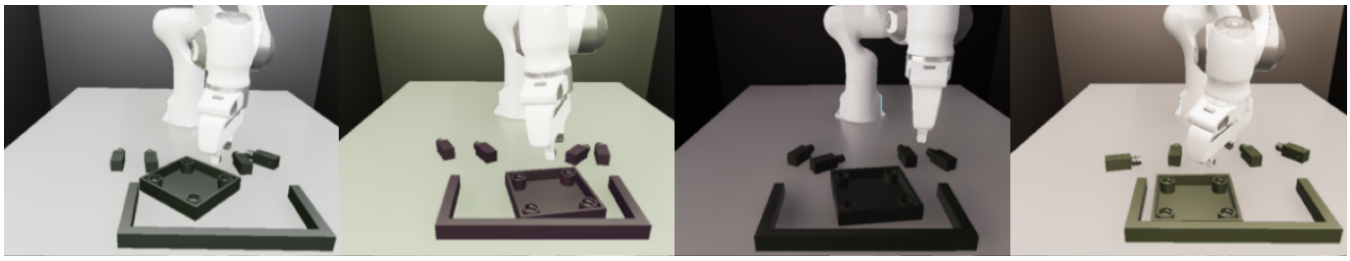


Fig. 23. Randomizing the visual appearance of the scene in the simulator allows for more fine-grained control and varying attributes that are hard to isolate in standard image augmentation techniques. Here, we illustrate how we can easily cover a larger space of part appearances without jittering the colors of everything else in the scene in tandem.

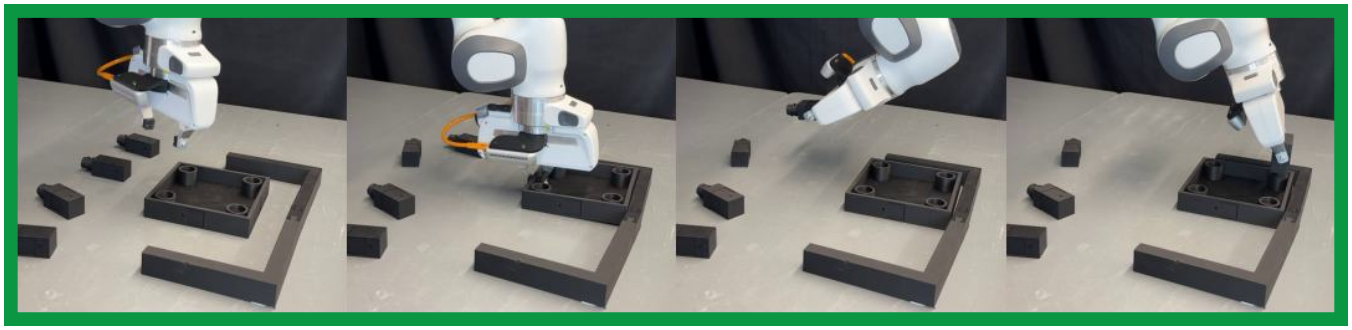


Fig. 24. An example of a successful rollout of a policy co-trained on 40 real-world demos containing only white parts and 400 synthetic demos with part colors randomized.

policies from scratch using carefully-designed shaped rewards and curricula, whereas we bootstrap RL from BC pre-training, which enables RL to operate with simple sparse rewards for achieving the desired assembly.

b) Complementary combinations of behavior cloning and reinforcement learning: Various combinations of learning from demonstrations/behavior cloning and reinforcement learning have begun maturing into standard tools in the learning-based control development paradigm [27,29]. For instance, demonstrations are often used to support RL in overcoming exploration difficulty and improving sample efficiency [16,80,99]. RL can also act as a robustification operator to improve upon base BC behaviors [16,29], paralleling the RL fine-tuning paradigm that has powered much of the recent advancement in other areas like NLP [100] and vision [40]. Additionally, many successful robotics deployments [56]–[58] have been powered by the "teacher-student distillation" paradigm, wherein perception-based "student" policies are trained to clone behaviors produced by a state-based "teacher" policy, which is typically trained via RL in simulation. We demonstrate that our residual RL approach for fine-tuning modern diffusion policy architectures can allow each of these complementary ways to combine BC and RL to come together and enable precise manipulation directly from RGB images.

APPENDIX VII
EXTENDED LIMITATIONS AND FURTHER WORK

a) Real-world distillation: Our experiments have demonstrated the effectiveness of online learning versus offline or passive learning through behavior cloning. Still, we employ only offline learning in our teacher-student distillation phase for sim-to-real transfer, which will likely upper-bound the performance we can transfer to the real world. Combining our pipeline with techniques for online learning could improve performance significantly. However, at this point, there are significant challenges to overcome to make this practically applicable to the tasks studied herein.

The field is progressing rapidly, and we are excited to investigate how online learning in the real world can be made practical for a broader set of tasks with longer horizons and less obvious ways of performing automatic state resets in follow-up work. This effort further ties into a more general framework for pre-training and adaptation of robot systems where the deployed robot can continue learning and adapting "on the job" after deployment. These investigations complement the methods presented in this paper and are not in scope.

At the same time, our results indicate that making more capable systems only through increasing the collection of real-world demos may also be fundamentally limited unless online learning is introduced as a fine-tuning step in those systems.

b) Locality of online correction learning: Though effective, we re-emphasize that our residual online reinforcement learning framework has the fundamental limitation of being bound to the pre-trained policy and mainly performing locally corrective actions. This limitation is both a strength and a weakness. First, the strong pre-trained prior allows RL to perform

the tasks and improve, and having a frozen prior helps stabilize training and prevent collapse. At the same time, the degree to which online learning can generalize to states far from the training set is limited.

c) Limitations of simulators in contact-rich tasks: We have added an experiment for a task from the Factory [54] task suite that pushes the accuracy of the simulator more than with the original FurnitureBench [5] tasks. This new task has a clearance of 0.2mm for the insertion, which shows that the general BC + Residual RL framework also works well in this setting. We did not show, however, that this transfers to the real world, and it would likely be more challenging than in the original tasks for at least two reasons. First, with increased precision requirements, accurate calibration of physics parameters between the actual and simulated environment will likely matter more. Second, performing manipulation from vision when parts are smaller is more challenging.

APPENDIX VIII
WHY ACTION CHUNKING AND DIFFUSION POLICIES?

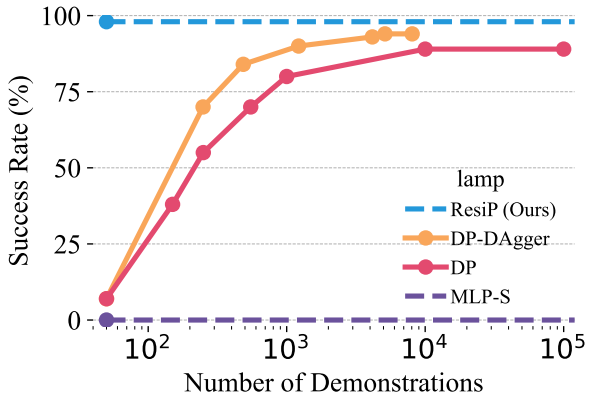
Simple feed-forward MLPs of modest size have shown impressive performance in many domains when trained with RL [16,56,57], and offer a natural starting point for RL fine-tuning after BC pre-training. However, the standard MLP policies trained to directly output single action control instead of a trajectory plan through an action chunk (MLP-S) fail across all tasks we consider. Therefore, we also trained MLP policies with action chunking (MLP-C). When we introduce chunking, MLP performance improves drastically, as shown in Tab. I. However, we also find that the more complex Diffusion Policy (DP) architecture generally outperforms MLPs, especially in tasks of intermediate difficulty. For example, an improvement from 10% success rate to 26% for the `one_leg` task on medium randomness makes subsequent fine-tuning far easier.

In one case, `lamp` on low randomness, MLP-C outperformed DP. In qualitative evaluations, we find that DP has smoother and faster actions, which is generally beneficial. Still, it seems to hurt performance in this case, as it tends to retract before the gripper fully grasps the lamp base. We also find that all methods struggle with the most challenging tasks, on which MLP-C and DP both achieve less than 5% success rate, indicating that there is still room for improvement in BC methods. The `peg-in-hole` task, despite its relatively short horizon of ∼100 timesteps, proved particularly challenging for BC methods. This task involves a ∼0.2 mm tolerance insertion, resulting in a 5% success rate. This poor performance on a short yet precise task lends credence to the hypothesis that BC methods are ill-equipped to handle high-precision requirements.

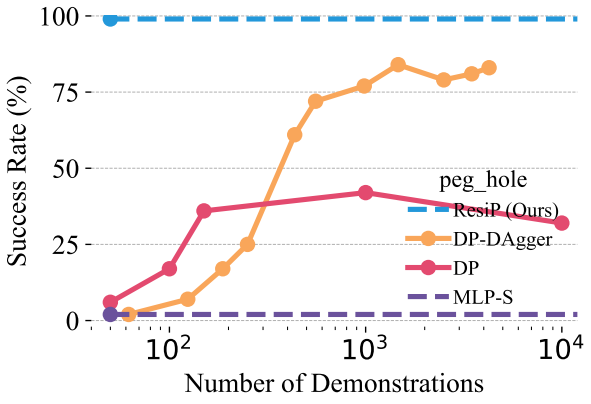
APPENDIX IX
FURTHER ANALYSIS OF OFFLINE VERSUS ONLINE LEARNING

A. Distillation scaling analysis

The scaling analyses in Fig. 25 show the same trends as in Fig. 1 (right) and Fig. 6 for the tasks `lamp` and `peg-in-hole`. We believe that this data point suggests that pure offline learning from demonstrations may not be sufficient for policies to learn robust and reactive policies, pointing towards the necessity for techniques like RL to reach a high level of robustness and reliability.



(a) Scaling analysis for the `lamp` task. This task appears significantly more conducive to offline learning than the other tasks tested.



(b) Success rates in exploration phase of training for `one_leg`, low randomness.

Fig. 25. We run similar scaling analyses as in Fig. 1 (right) and Fig. 6 for `lamp` and `peg-in-hole`. The general findings of interactive learning are that it is more efficient and has higher asymptotical performance. However, the difference appears to be much smaller for `lamp` and bigger for `peg-in-hole`. What drives these differences are left for future work.

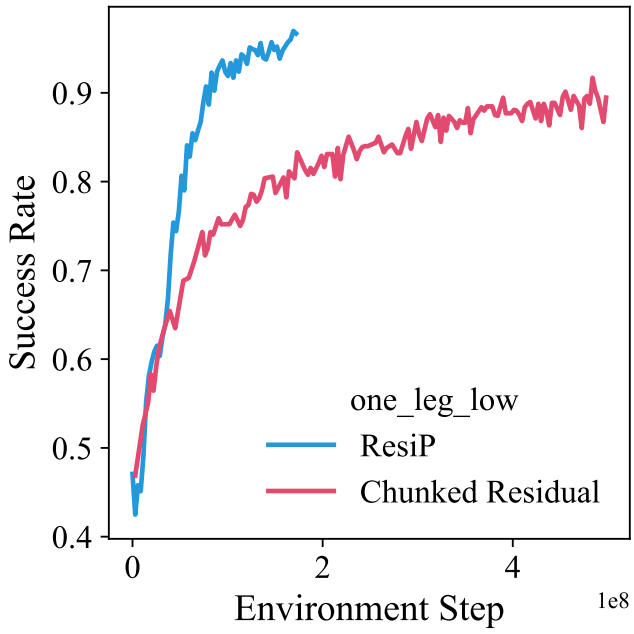


Fig. 26. When learning a residual correction term that corrects the whole chunk at a time online, we find that the learning is significantly slower (measured in environment steps) and saturates a lower asymptotic level.

B. Interactive distillation with DAgger

DAgger [1] can learn from scratch significantly more efficiently than pure BC measured in both gradient steps and samples. We have added the DAgger performance to the scaling plot, shown in orange in Fig. 25. Consider the scaling plot in Fig. 1 (right) as an example. In ∼10k gradient steps, DAgger surpasses BC from 50 human demos trained with ∼100k steps. After 10k steps, it has around 800 rollouts in the aggregated dataset. After around 20k gradient steps, it seems to surpass the best-performing BC distillation runs using more than 10k rollouts and 500k gradient steps, at which point it has ∼1.5k demonstrations in the replay buffer. This result highlights the effectiveness of online and interactive learning as opposed to learning purely passively from an offline dataset. Furthermore, it highlights that the expert we query is an effective teacher. It also highlights that for interactive learning to be effective, one needs to have a teacher ready to be queried as learning progresses.

APPENDIX X
RESIDUAL RL ABLATIONS

A. Effect of fully versus partially closed-loop policies

One differentiating factor of our residual model from some prior work is that the base and residual models make predictions at different frequencies, i.e., every 8 timesteps for the base model and every timestep for the residual model. Making predictions with the most up-to-date information is likely an easier prediction problem, and we expect this to work better than the "standard" setup of letting the residual correct the full output of the base model. When training a residual model that corrects a whole chunk at a time but otherwise uses the same hyperparameters, we observe that training is less sample efficient and performance saturates at a lower success rate. In particular, the chunked residual policy reaches ∼85% success rate in about 250 million environment steps, while the one-step residual needs about 75 million. Lars: Add some more details about the training procedure.

To further probe the difference between fully closed-loop policies and those using chunking, we evaluate the policies with perturbations added to the parts in the environment throughout the episode. In particular, at each timestep, 1% of parts across the environments will have a random force applied to them. The forces are sampled from the same distribution as the initial part randomization distribution.

See Fig. 9 and Tab. XIV for results. We generally see that the partially open-loop policies have a bigger drop in performance when perturbations are introduced, around 20 percentage points compared to 12 for the one-step residual model.

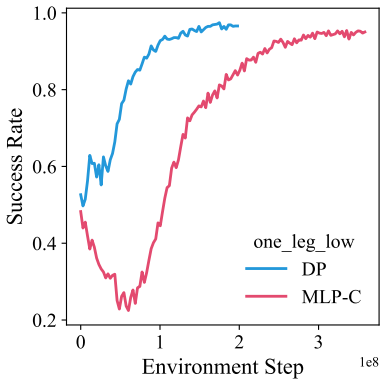
Model	No Perturb	W/ Perturb	Drop in SR
Standard RPPO	98%	86%	12 pp
Chunked RPPO	92%	73%	19 pp
Chunked pre-trained BC	52%	32%	20 pp
DAgger chunked student DP	90%	68%	24 pp

TABLE XIV. Success rates with/without perturbations for different models. SR = Success Rate, pp = percentage points.

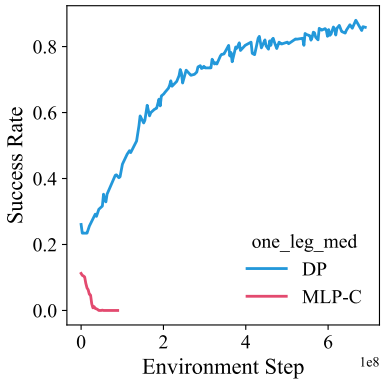
B. Residual base policy ablation

a) MLP as base: To further tease apart what part of the diffusion policy that provides the most important performance increase, the action chunking or the denoising diffusion process, we run the same residual PPO run for the `one_leg` task as before, but with the best-performing BC MLP model in place of the diffusion policy. The results are shown in Fig. 27a and Fig. 27b. The resulting training dynamics are intriguing. Despite the initial success rate of the base model being close to that of the diffusion model, the success rate drops markedly when exploration noise is introduced. This is especially visible in the training performance in plot 2 below. We also notice that the evaluation performance drops as the residual model explores and learns more. However, the residual is eventually able to find actions that the MLP responds better to and, in the end, converges to a similar performance as the diffusion-based runs. In the more challenging task with higher initial state randomness, the same initial dynamic plays out, but the training performance drops to zero, causing the learning to collapse. We conclude that any base model achieving a high enough initial success rate can be plugged into our framework (and, based on our BC experiments, a base model with chunking is likely to outperform one without chunking) but that the expressivity and robustness to input noise offered by diffusion de-noising also contributes to downstream performance benefits during residual RL.

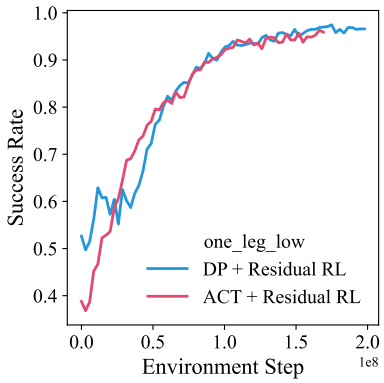
b) ACT as base: To see if the robustness to noise and suitedness for residual learning is unique to the diffusion-type model, we also implement and test using Action-Chunked Transformer (ACT) [6] as the base model for the `one_leg` task at low randomness. With some tuning, we find that the ACT model can achieve comparable performance as the diffusion model, though slightly lower, in pre-training. In the fine-tuning phase, however, it functions as well and stably as the diffusion model base, as shown in Fig. 27c. This suggests that the residual RL framework is suitable for a wide range of fine-tuning applications and may be applied to fine-tune even larger and possibly multi-task models.



(a) Evaluation success rates for RL training for `one_leg`, low randomness for Diffusion and MLP base policy.



(b) Evaluation success rates for RL training for `one_leg`, medium randomness for Diffusion and MLP base policy.



(c) Evaluation success rates for RL training for `one_leg`, low randomness for Diffusion and ACT base policy.

Fig. 27. We compare the diffusion-based residual RL training performance with the best-performing MLP as the base model in (a) and (b). As we can see, despite having similar pertaining performance (for low randomness), the MLP-based residual model performs poorly compared to the diffusion-based one. On a higher randomness setting, it fails to complete the task. We compare with using the ACT [6] as the base policy in (c). The pre-training performance is slightly worse, but performance quickly catches up during online learning, indicating that the ACT model is also well-suited for residual learning.

C. Residual action scaling parameter ablation

A design choice we make is the parameter $\alpha = 0.1$. The parameter choice is somewhat arbitrary and was informed by some intuitions about the task. For example, since the residual model intends to make local corrections, we want to imbue it with that inductive bias. In the normalized action space, the workspace is constrained to [-1, 1], and letting a $\sigma = 1$ for the residual Gaussian model correspond to [-0.1, 0.1] on the macro scale seemed reasonable.

We have tested more values of the parameter $\alpha \in \{0.01, 0.05, 0.2, 1.0\}$, but kept the resulting exploration noise on the macro scale fixed (i.e., scaled with the value of α, so $\alpha_1\sigma_1 = \alpha_2\sigma_2$). The result, shown in the figure below, shows a remarkable robustness to this parameter, and all cases have very similar performance.

We note a couple of observations. First, $\alpha = 0.2$ seems to perform slightly better than our original $\alpha = 0.1$. Second, different levels of α also result in very different magnitudes of activations at the last layer, which impacts losses. This experiment shows that the resulting performance does not differ significantly, but we suspect it could make training less stable in harder settings.

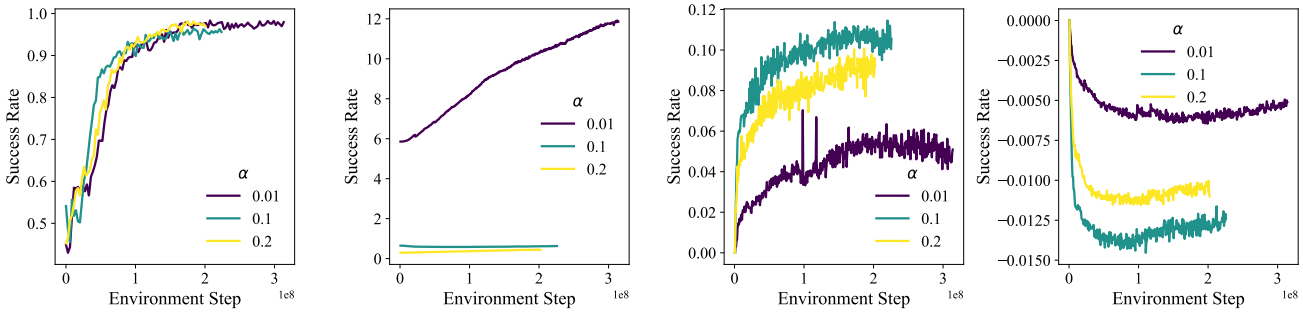


Fig. 28. We test different values of the residual action scaling parameter α and test it for values $\alpha \in \{0.01, 0.1, 0.2\}$ while adjusting the exploration noise to be such that the macro-level exploration is the same initially. We find that for success rates in this task, the value is not crucial but does cause training dynamics to change, particularly the residual model output norms and policy loss magnitude.