Reverse to Advance: Teleoperation-Cost Effective Hard Policy Learning from Reversed Easy Tasks

Qiyuan Qiao* **Ge Yuan*** **Can Wang†** **Dong Xu**
The University of Hong Kong
qiaoqy@connect.hku.hk, gavinyuan97@gmail.com
{canwang,dongxu}@hku.hk

Abstract

High-quality teleoperation datasets are costly to collect, particularly for hard tasks. We observe that many tasks exhibit directional asymmetry: completing the forward hard task is difficult, whereas reversing it by relaxing or disrupting the environment is comparatively easy. This suggests that reversed easy-task trajectories can serve as a scalable supervision signal for the hard task, reducing the cost of manual demonstration collection. However, reversed data can be noisy, and directly training on it may yield suboptimal policies. To enable largely automated acquisition and effective use of reversed data, we propose a teleoperation-cost effective framework for hard policy learning via temporal reversal of easy tasks, consisting of three key components: a closed-loop data collection pipeline that alternates between hard-task and easy-task policies to autonomously reset the environment and generate diverse trajectories; a hierarchical data refinement pipeline that temporally inverts easy-task rollouts and filters low-quality motion using kinematic priors and a critic-guided advantage filter; and an iterative policy learning method that trains the hard-task policy using both initial reversed easy-task demonstrations and the filtered reversed data in a continuous online learning loop. By combining automated collection, hierarchical refinement, and iterative learning, our method enables scalable, reliable training of complex, high-precision manipulation tasks. Across two simulated benchmarks and real-robot experiments, we demonstrate that our method improves hard-task success rates with higher data efficiency and more stable training compared to reversal-based and reinforcement-learning baselines, without requiring extensive hard-task teleoperation.

1 Introduction

Learning complex robotic manipulation tasks remains a fundamental challenge in embodied AI. While Imitation Learning [12, 28], particularly through Diffusion Policies [6], has shown remarkable prowess in mapping high-dimensional observations to precise actions, its performance is heavily bottlenecked by the availability and quality of expert demonstrations. However, collecting sufficient demonstrations is an arduous process. This process demands unwavering concentration from human operators, as even a split-second lapse in focus can result in a failed trial [10]. Furthermore, the quality of human-generated data is inherently inconsistent; datasets are frequently plagued by suboptimal paths, unintended oscillations, and varying levels of expertise, which can introduce significant noise and hinder the policy's ability to converge on a robust solution. These limitations are particularly exacerbated in hard tasks characterized by high-precision requirements and complex contact dynamics.

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

We observe that in many manipulation scenarios, there exists a fundamental asymmetry in task difficulty: solving a task, such as inserting a peg into a hole, stacking delicate blocks, or assembling a complex mechanism, is significantly harder than unsolving it, such as extraction, de-stacking, or disassembly. While the forward hard task often requires navigating a narrow state-space funnel with high-precision contact, the reverse easy task typically involves expanding into free space, making it far more amenable to both human teleoperation and automated execution. This observation provides a key insight: *Can we leverage the abundant, easily collected demonstrations from these reverse tasks to bootstrap the learning of the original hard task?* If a robot can master the easy reverse task, the resulting trajectories, when temporally inverted, could theoretically serve as a rich source of supervision for the hard task. This approach promises to drastically reduce the human labor required by substituting high-effort expert demonstrations with low-effort, scalable data from the task's inverse. However, simply reversing these trajectories is often insufficient, as they lack the corrective behaviors needed to handle the precision requirements and covariate shift inherent in the hard-task domain.

To the best of our knowledge, existing approaches rarely exploit this forward–reverse task asymmetry. Prior work either relies on direct Reinforcement Learning from scratch [13, 5, 27] or on standard data augmentation with the reversed data [2]. However, these methods still fail to account for the inherent noise and suboptimal behaviors present in reversed trajectories. These observations motivate the design of a framework for learning complex manipulation policies under teleoperation cost constraints via simple-task temporal reversal. To achieve this, several key challenges must be addressed: 1) **Automatic data collection:** How can we construct a largely automated pipeline for acquiring reversed-task demonstrations, thereby minimizing the human teleoperation effort? 2) **Data filtering:** How can we identify and extract the most valuable, high-quality trajectories from the noisy reversed data to serve as reliable supervision for the original hard task? 3) **Progressive learning:** Is it possible to integrate automated data collection and filtering into an online learning loop, where the policy continuously improves by leveraging filtered data, allowing the system to progressively approach optimal performance on the hard task?

To address these challenges, we propose **Auto-E2H** (**Auto**mated **E**asy-**to**-**H**ard Data Acquisition and Refinement), a teleoperation-cost effective framework for hard policy learning via temporal reversal. To efficiently collect high-quality data with minimal human intervention, we first design a **closed-loop data collection pipeline** that alternates between hard-task and easy-task policies. This mechanism enables the policies to reset each other, generating diverse trajectories while significantly reducing the need for teleoperation. To filter noise in the collected data, we further design a **hierarchical data refinement pipeline**, where easy-task rollouts are temporally reversed and processed through kinematic filtering to remove slow or static segments, followed by critic-guided advantage filtering to retain only high-value trajectory segments. We further propose an **iterative policy learning method**, where the hard-task policy is trained using both the initial reversed easy-task demonstrations and the filtered reversed data, forming a continuous online learning loop that progressively improves performance. By combining automated collection, hierarchical refinement, and iterative policy learning, our framework ensures that the hard-task policy is trained on informative, diverse, and reliable trajectories, enabling scalable learning of complex, high-precision manipulation tasks with minimal human supervision.

2 Related Works

Imitation Learning in Robotics. Imitation Learning [12, 28] aims to train policies by leveraging demonstrations from expert behavior. Classic frameworks include Behavioral Cloning [8, 23, 9, 17, 20, 24, 1] and more recent methods based on generative modeling, such as diffusion policies [6], which have shown strong capabilities in mapping high-dimensional observations to precise actions. However, the success is heavily dependent on the quality and quantity of demonstrations. Complex and high-precision manipulation tasks often require substantial human effort to collect consistent, expert-level trajectories, and suboptimal or noisy demonstrations can significantly hinder policy performance [4]. To address this challenge, we propose a teleoperation-efficient framework that uses easily collected reversed-task trajectories to aid policy learning of the corresponding forward, complex tasks, reducing reliance on large-scale high-quality expert data.

Data Collection and Refinement in Teleoperation. Collecting high-quality demonstrations remains a major bottleneck for imitation learning. Recent work has focused on improving efficiency and data utility. Some approaches use portable or universal teleoperation systems to capture rich,

multi-task demonstrations across robots and environments, reducing repeated human effort [7, 25]. Others enhance learning from limited data via augmentation, human-in-the-loop refinement, or pre-trained multimodal models [16, 21, 26, 3]. Compositional and structured strategies exploit environment variations and scaling laws to improve generalization and maximize the value of each demonstration [10, 15]. Recent works have considered how to refine the collected data. For example, GR-RL [14] adopts a learned task progress model to identify low-quality transitions in human demonstrations, discarding segments that do not contribute positively to task progress before applying offline and online reinforcement learning. TR-DRL [13] exploits dynamics-consistent filters to identify fully reversible transitions, augment trajectories by reversing them, and apply potential-based reward shaping to partially reversible components. In contrast, we propose a hierarchical data refinement method that combines kinematic filtering and critic-guided advantage evaluation. Unlike prior work, our approach removes low-quality or static segments and retains only those contributing positively to task success, enabling more effective use of reversed-task trajectories.

Forward–Reverse Task Asymmetry. In many manipulation scenarios, there exists a natural asymmetry in task difficulty: the forward task often involves precise, constrained interactions, whereas the corresponding reversed task is typically easier. Leveraging this asymmetry has the potential to reduce human teleoperation effort by using easily collected reversed-task trajectories to supervise learning of the harder forward tasks. Existing works have explored temporal or time-reversal symmetries to exploit this structure in RL [11, 27, 5, 2, 13] and self-learning [18]. TR-DRL [13], for example, identifies fully reversible transitions using a dynamics-consistent filter, augments these transitions by reversal, and applies potential-based reward shaping to partially reversible components, thereby improving sample efficiency in both single- and multi-task RL benchmarks. However, such methods primarily focus on enforcing physical reversibility or shaping rewards based on object states, and they do not explicitly address the combination of data collection, noise filtering, and policy learning from reversed trajectories. In contrast, our framework integrates automated collection, hierarchical refinement, and iterative policy learning to leverage reversed-task data effectively, enabling scalable learning of complex forward tasks with minimal human intervention.

3 Method

Problem Formulation. Consider a finite-horizon manipulation task of interest, denoted by $\mathcal{T}_h$, which exhibits a high degree of difficulty due to complex dynamics. We define its corresponding *reverse-direction task* $\mathcal{T}_e$, which is assumed to be easier to solve. Both tasks share a common observation space $\mathcal{S}$, action space $\mathcal{A}$, and horizon H. Formally, let $\tau_e = \{(\mathbf{s}_t, \mathbf{a}_t)\}_{t=0}^{H-1}$ denote a successful trajectory of $\mathcal{T}_e$. We assume the transformed trajectory $\tilde{\tau}_h = \mathrm{Reverse}(\tau_e)$ contains state-action segments that provide meaningful supervision for learning a policy on $\mathcal{T}_h$. Our goal is to learn a policy π_h for the hard task $\mathcal{T}_h$ using a limited initial hard-direction dataset $\mathcal{D}_h$ obtained by reversing easy-task demonstrations, together with additional easy-task data $\mathcal{D}_e$:

$$\pi_h^* = \arg\max_{\pi_h} \mathbb{E}_{\tau_h \in \mathcal{D}_h, \tau_e \in \mathcal{D}_e} \Big[R(\tau_h, \tilde{\tau}_h) \Big| \tilde{\tau}_h = \mathrm{Reverse}(\tau_e) \Big]. \tag{1}$$

where $R(\cdot)$ denotes the success rate of the deployment of the hard task $\mathcal{T}_h$.

Overview. To address this challenge, we introduce **Auto-E2H**, a time-reversal-based self-improving framework for hard-task policy learning, as shown in Figure 1. First, to enable efficient, scalable data collection and support online self-improvement, we expect to execute the hard-task and easy-task policies in an alternating closed loop, allowing each policy to reset the environment for the other and alleviating repeated manual interventions. To implement this, we propose a closed-loop demonstration collection method (**Section 3.1**) that ensures continuous and diverse trajectory generation for both tasks. Second, the reversed trajectory $\mathrm{Reverse}(\tau_e)$ can be very noisy for learning the hard-task policy π_h. Thus, we propose a hierarchical refinement process (**Section 3.2**) to obtain high-quality candidate segments, ensuring that the trajectories $\tilde{\tau}_h = \mathrm{Reverse}(\tau_e)$ provide meaningful supervision for hard-task policy learning. Third, to iteratively improve the hard-task policy π_h, we propose an iterative policy learning algorithm (**Section 3.3**) that updates π_h using both the initial reversed easy-task demonstrations and the refined reversed trajectories.

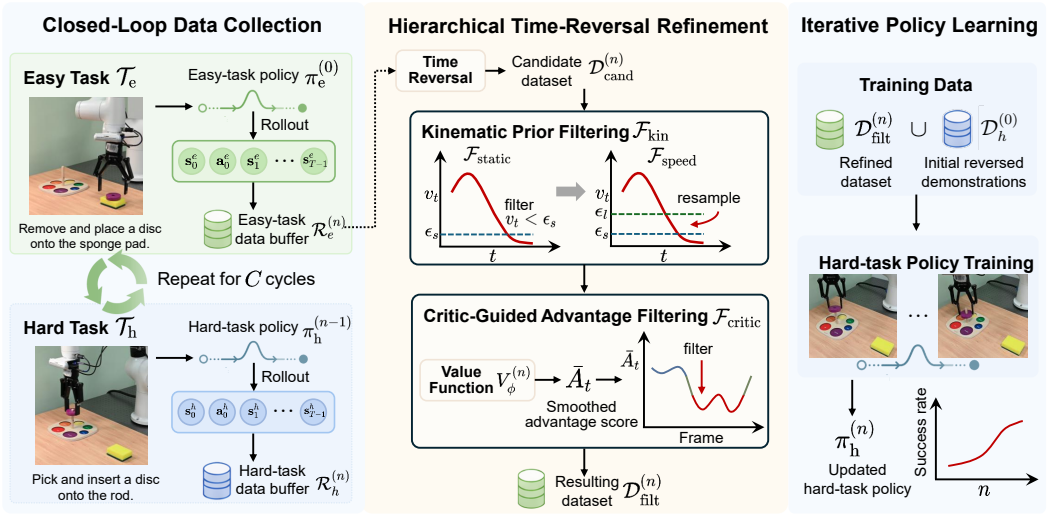

Figure 1: **Framework.** Our framework consists of three key components. First, the Closed-Loop Data Collection stage alternates between executing the easy-task policy $\pi_e^{(0)}$ and the evolving hard-task policy $\pi_h^{(n-1)}$ over multiple cycles, generating diverse trajectory data buffers for each task. In the Hierarchical Time-Reversal Refinement step, easy-task trajectories are reversed and undergo kinematic filtering to remove slow or static segments, followed by critic-guided advantage filtering to prune low-progress segments. Finally, Iterative Policy Learning updates $\pi_h^{(n)}$ using the fixed initial dataset $\mathcal{D}_h^{(0)}$ and the current refined dataset $\mathcal{D}_{\text{filt}}^{(n)}$.

3.1 Closed-Loop Data Collection

Since our goal is to enable online self-improvement of the policy, data collection must be efficient, scalable, and minimally reliant on human intervention. By leveraging replay rollouts from the action policy, Auto-E2H substantially reduces the cost of human teleoperation, while the stochastic denoising process inherent to diffusion policies further promotes behavioral diversity in the collected data. However, a key challenge is the need for frequent environment resets: after executing the hard task, the system must return to a valid initial configuration for the next trial. Auto-E2H resolves this issue through an autonomous closed-loop collection strategy, where the hard-task and easy-task policies are executed in an alternating manner to reset the environment for each other, alleviating the need for repeated human intervention.

Specifically, at training iteration n, we perform C collection cycles. In each cycle $c \in \{1, \ldots, C\}$, the process consists of two stages: 1) **Hard-task rollout**: we execute the current hard-task policy $\pi_h^{(n-1)}$ on $\mathcal{T}_h$ for up to H steps, yielding a trajectory $\tau_h^{(c)}$; 2) **Easy-task rollout**: starting from the terminal state of $\tau_h^{(c)}$, we execute the fixed easy-task policy $\pi_e^{(0)}$ on $\mathcal{T}_e$ for up to H steps, yielding a trajectory $\tau_e^{(c)}$. The terminal state of $\tau_e^{(c)}$ then serves as the initial state for the next hard-task rollout, forming a closed interaction loop. After an external reset, this process can repeat without manual intervention while automatic resets succeed. If a reset fails, collection pauses, the arm returns home, and an operator restores the scene before collection resumes. We index the hard-task and easy-task trajectory buffers by the collection iteration n:

$$\mathcal{R}_h^{(n)} = \{\tau_h^{(c)}\}_{c=1}^{C}, \quad \mathcal{R}_e^{(n)} = \{\tau_e^{(c)}\}_{c=1}^{C}. \tag{2}$$

All trajectories in $\mathcal{R}_h^{(n)}$, including failures, are used for critic training. Only successful trajectories from $\mathcal{R}_e^{(n)}$ are used to construct candidate reversed data for policy learning.

This closed-loop mechanism expands the initial-state distribution for the hard-task policy, as the easy-task policy produces diverse terminal states. After reversal and transformation, these states yield varied candidate starting configurations for the hard task. While this diversity is beneficial, it also introduces noise, making a refinement stage necessary to filter out low-quality segments.

3.2 Hierarchical Data Refinement

To optimize the collected data, we perform a three-step hierarchical refinement: data preprocessing, which reverses the trajectory; kinematic filtering, which removes slow or static segments and regularizes motion; and critic-guided advantage filtering, which prunes low-progress segments to align with the hard-task goal.

Data Preprocessing. For each successful easy-task rollout $\tau_e = \{(\mathbf{s}_t, \mathbf{a}_t)\}_{t=0}^{T-1}$ in the buffer $\mathcal{R}_e^{(n)}$, we construct a candidate hard-task trajectory by reversing the state sequence:

$$\tilde{\tau}_h = \{\mathbf{s}_{T-1-t}\}_{t=0}^{T-1}. \tag{3}$$

Next, the action sequence is reconstructed from adjacent reversed end-effector poses and the gripper state rather than directly reusing the original actions:

$$\mathbf{a}'_t = h(\mathbf{x}_t, \mathbf{x}_{t+1}, g_t), \quad t = 0, \ldots, T-2, \tag{4}$$

where $\mathbf{x}_t = [x, y, z, q_w, q_x, q_y, q_z]$ is the reversed end-effector pose in the robot frame and $g_t \in \{0, 1\}$ is the reversed gripper signal (1: open; 0: closed). The function $h(\cdot)$ maps adjacent reversed poses and the gripper signal to the policy's 8-D absolute-pose and gripper-command representation. The last frame is discarded because it has no next-pose target. Directly reversing the original actions misaligns the pose targets, while negating absolute positions or quaternions does not invert motion. This reconstructs trajectory geometry rather than contact forces. After these processing steps, we obtain the candidate dataset $\mathcal{D}_{\text{cand}}^{(n)}$ for the training iteration n:

$$\mathcal{D}_{\text{cand}}^{(n)} = \left\{ (\tilde{\tau}_h^{(c)}, \mathbf{a}'^{(c)}) \;\middle|\; 1 \le c \le C,\ \tau_e^{(c)} \text{ succeeds} \right\}. \tag{5}$$

The reversed trajectories in $\mathcal{D}_{\text{cand}}^{(n)}$ are diverse but noisy, and training directly on this data would degrade policy quality. Therefore, we apply Kinematic Prior Filtering followed by Critic-Guided Advantage Filtering:

$$\mathcal{D}_{\text{filt}}^{(n)} = \mathcal{F}_{\text{critic}}\big(\mathcal{F}_{\text{kin}}(\mathcal{D}_{\text{cand}}^{(n)})\big). \tag{6}$$

Kinematic Prior Filtering. The kinematic filter is defined as:

$$\mathcal{F}_{\text{kin}} = \mathcal{F}_{\text{speed}} \circ \mathcal{F}_{\text{static}}. \tag{7}$$

The purpose of the static filter $\mathcal{F}_{\text{static}}$ is to address segments of the trajectory where the motion is unnaturally slow. Such segments are statistically distinguishable through near-static or abnormally low-velocity kinematic profiles caused by inference latency, communication delay, or ambiguity-induced oscillations during action-policy rollouts. To handle this, we first calculate the instantaneous displacement v_t of the end-effector between time steps t and $t+1$:

$$v_t = \|\mathbf{x}_{t+1} - \mathbf{x}_t\|_2. \tag{8}$$

We then eliminate contiguous intervals where the frame-to-frame displacement remains below a threshold ϵ_s, and the total spatial drift does not exceed ϵ_d over more than L_{s} consecutive steps. This effectively removes hovering and frozen segments that contribute little to no supervisory signal. The speed filter $\mathcal{F}_{\text{speed}}$ targets segments whose velocity lies between the static threshold ϵ_s and a low-speed threshold ϵ_l. For these segments, where the velocity is higher than ϵ_s but still low enough to warrant adjustment, we resample the motion to accelerate it toward a target step size ΔH. This acceleration regularizes the action distribution, helping to ensure a more consistent and dynamic trajectory. Frames where the velocity exceeds the low-speed threshold ϵ_l are left unaltered, as they are considered to have sufficiently natural motion speeds. After these processing steps, we obtain a kinematically cleaned dataset $\mathcal{D}_{\text{kin}}^{(n)}$.

Critic-Guided Advantage Filtering. Kinematic cleanup removes obviously degenerate frames but does not certify that the remaining segments make actual progress on the hard task. We address this with Critic-Guided Advantage Filtering $\mathcal{F}_{\text{critic}}$, where a critic trained on the *current* hard-task rollouts is used to prune disadvantage segments from the kinematically cleaned dataset $\mathcal{D}_{\text{kin}}^{(n)}$.

At each iteration n, we train a value function $V_\phi^{(n)}(\mathbf{s})$ from scratch on all hard-task rollouts in $\mathcal{R}_h^{(n)}$. With success indicator $y \in \{0, 1\}$, episode length $T \le H$, per-step reward -1, terminal

success reward H, and discount γ=1, the raw return is $G_t^{\text{raw}} = -(T-t) + Hy$. We normalize it as $\bar{G}_t = (G_t^{\text{raw}} - H)/(2H)$ and minimize

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_{\tau \sim \mathcal{R}_h^{(n)},\, t}\left[\left(V_\phi(\mathbf{s}_t) - \bar{G}_t\right)^2\right], \qquad \bar{G}_t = -\frac{T-t}{2H} - \frac{1-y}{2}. \tag{9}$$

Even when all rollouts fail with $T = H$, $\bar{G}_t = -1 + t/(2H)$ remains nonconstant. At cold start this provides temporal-ordering supervision, rather than a calibrated success probability. We retrain the critic from scratch at every iteration so that it tracks the state distribution and failure modes of the current hard-task policy rather than a stale mixture of earlier policies.

For each reversed trajectory, we optionally truncate the suffix after it enters a high-value region to suppress irrelevant post-contact retreat phases. Then, with $\bar{r} = -1/(2H)$, we compute a per-step progress score using temporal-difference advantages:

$$\delta_t = \bar{r} + \gamma V_{t+1} - V_t, \quad A_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}. \tag{10}$$

Frames with the smoothed advantage score $\bar{A}_t < \alpha_{\text{drop}}$ are marked for removal, and we ensure minimum run lengths for both retained and dropped segments to avoid fragmentation. The resulting dataset is $\mathcal{D}_{\text{filt}}^{(n)}$. This approach ensures that the critic only vetoes segments that do not align with the task's progress, reducing the risk of critic errors impacting the training process.

3.3 Iterative Policy Learning

Algorithm 1 Iterative policy learning with alternating hard-task and easy-task rollouts. The easy-task policy remains fixed; each hard-task policy update uses the initial demonstrations and current filtered data.

Require: Easy-task demonstrations $\mathcal{D}_e^{(0)}$ and their reversed hard-direction dataset $\mathcal{D}_h^{(0)}$; iterations N; cycles C

Train $\pi_h^{(0)} \leftarrow$ TrainPolicy($\mathcal{D}_h^{(0)}$) and $\pi_e^{(0)} \leftarrow$ TrainPolicy($\mathcal{D}_e^{(0)}$)
for $n = 1, 2, \ldots, N$ **do**
  Collect hard-task and easy-task rollouts $\mathcal{R}_h^{(n)}$ and $\mathcal{R}_e^{(n)}$ using $\pi_h^{(n-1)}$ and $\pi_e^{(0)}$ (Eq. 2)
  Invert and align successful rollouts from $\mathcal{R}_e^{(n)}$ to form $\mathcal{D}_{\text{cand}}^{(n)}$ (Eq. 5)
  Apply kinematic refinement: $\mathcal{D}_{\text{kin}}^{(n)} \leftarrow \mathcal{F}_{\text{kin}}(\mathcal{D}_{\text{cand}}^{(n)})$ (Eq. 7)
  Train value function $V_\phi^{(n)}$ from scratch on all hard-task rollouts $\mathcal{R}_h^{(n)}$ (Eq. 9)
  Filter with the critic: $\mathcal{D}_{\text{filt}}^{(n)} \leftarrow \mathcal{F}_{\text{critic}}(\mathcal{D}_{\text{kin}}^{(n)} \mid V_\phi^{(n)})$ (Eq. 10)
  Update $\pi_h^{(n)} \leftarrow$ TrainPolicy($\mathcal{D}_h^{(0)} \cup \mathcal{D}_{\text{filt}}^{(n)}; \pi_h^{(n-1)}$)
end for
return $\pi_h^{(N)}$

We further propose an Iterative Policy Learning algorithm to progressively improve the hard-task policy using initial reversed easy-task demonstrations and refined easy-task rollouts (Algorithm 1). Initially, the policies are trained on $\mathcal{D}_h^{(0)}$ and $\mathcal{D}_e^{(0)}$. At iteration n, policies $\pi_h^{(n-1)}$ and $\pi_e^{(0)}$ collect buffers $\mathcal{R}_h^{(n)}$ and $\mathcal{R}_e^{(n)}$. Successful easy-task trajectories are inverted and aligned to form $\mathcal{D}_{\text{cand}}^{(n)}$ (Equation 5), then refined by $\mathcal{F}_{\text{kin}}$ (Equation 7).

The critic $V_\phi^{(n)}$ is trained on all current hard-task rollouts $\mathcal{R}_h^{(n)}$, including failures, using Equation 9. Critic-guided advantage filtering then produces $\mathcal{D}_{\text{filt}}^{(n)}$ using Equation 10. We warm-start $\pi_h^{(n)}$ from $\pi_h^{(n-1)}$ and rebuild its training set at each iteration as $\mathcal{D}_h^{(0)} \cup \mathcal{D}_{\text{filt}}^{(n)}$: the fixed initial reversed demonstrations and the current iteration's accepted data. The easy-task policy $\pi_e^{(0)}$ remains fixed throughout.

4 Experiments

Experimental Setup. We evaluate our method on four Isaac Lab, two Robosuite[29], and four real-world task pairs, each consisting of a target hard task Task $\mathcal{H}$ and an easier reverse-direction task

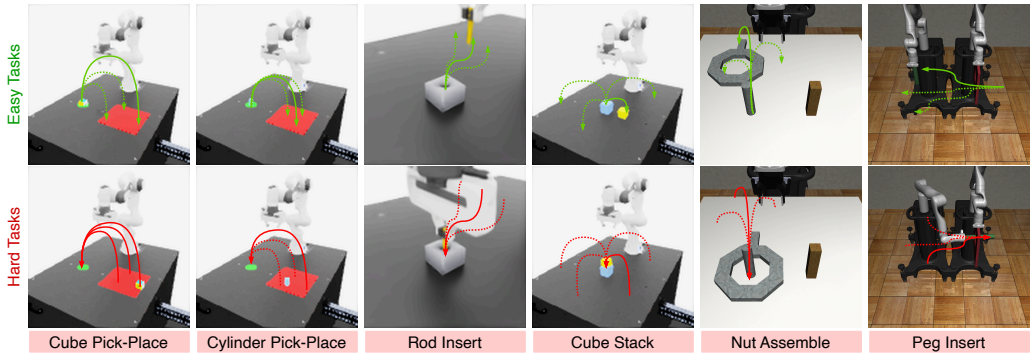

Figure 2: **Simulation benchmark task setups.** The first four pairs are Isaac Lab tasks spanning object relocation, contact-rich insertion, and multi-object stacking The last two pairs are Robosuite nut assembly/disassembly and peg insertion/removal. Within each pair, Task $\mathcal{H}$ and Task $\mathcal{E}$ share the same robot, observations, and action space but have opposite success directions.

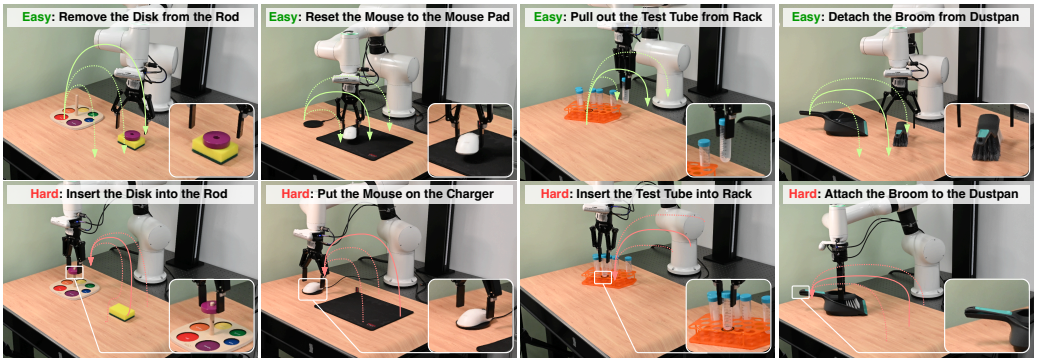

Figure 3: **Real-world robot task setups.** The four panels show near-completion photographs of the Disk, Mouse, Test Tube, and Brush tasks used in our real-world evaluation. In each task pair, Task $\mathcal{H}$ and Task $\mathcal{E}$ share a geometric path; their contact dynamics need not be exactly reversible.

Task $\mathcal{E}$, as shown in Figure 2 and Figure 3. We report hard-task success rates over 40 evaluation trials per Isaac Lab task, 50 per Robosuite task, and 20 per real-world task. We use a single set of filtering thresholds across simulation and real-world environments, without per-task tuning.

We compare our method with two RL-based approaches, TR-DRL [13], which combines temporal reversal with RL for policy learning, and RECAP [19], an offline RL algorithm that learns from the rollouts of a hard policy, as well as a direct hard-demo baseline Direct-$\mathcal{H}$. To ensure a fairer comparison, we also compare with TR-DRL-DP, which uses dynamics-based reversal within the same Diffusion Policy collection and imitation-learning loop as Auto-E2H. For the Diffusion Policy comparisons using reversed easy-task data, baselines share the same initial demonstrations, backbone, and hyperparameters as Auto-E2H. The hard-demo BC comparison instead uses 50 directly collected hard-task demonstrations; the Robosuite TR-DRL comparison uses its original SAC implementation. RECAP differs only by adding a one-dimensional indicator to the Diffusion Policy conditioning input to identify rollout source.

4.1 Main Results

Overall performance. Across all three suites, Auto-E2H achieves monotonic gains over six self-improvement iterations on simulation and four iterations in the real world (Figures 4 and 5). Pooled across tasks within each suite, Task $\mathcal{H}$ success rises from 33.1% to 96.3% on Isaac Lab (160 trials per iteration), from 0.0% to 85.0% on Robosuite (100 trials), and from 40.0% to 70.0% in the real world (80 trials), with non-overlapping pooled 95% Wilson intervals between the initial and final policies in every suite. The learned Task $\mathcal{E}$ policy also serves as an automatic scene-reset module during real-world rollouts, succeeding in 80%, 95%, 82.5%, and 80% of cycles on Disk, Mouse, Test

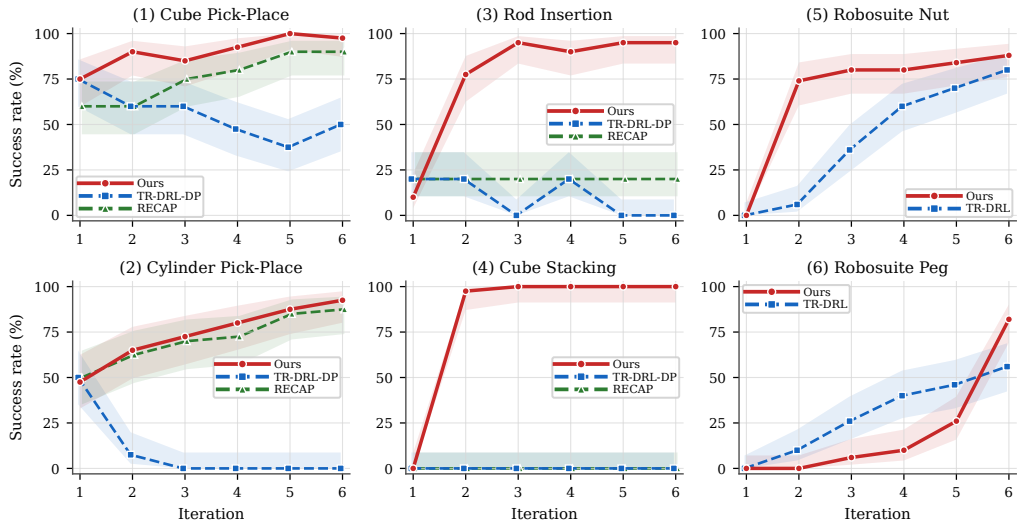

Figure 4: **Isaac Lab and Robosuite success curves.** We report four Isaac Lab tasks (1–4) and two Robosuite tasks (5–6). Each subplot reports Task $\mathcal{H}$ success rate over self-improvement iterations.

Tube, and Brush (84.4% average), so manual intervention is needed only for the remaining failed reset cycles. Detailed comparisons against TR-DRL-DP, RECAP, and Direct-$\mathcal{H}$ are reported in the corresponding subsections.

Isaac Lab benchmarks. In Figure 4, we compare Auto-E2H with TR-DRL-DP and RECAP. Over the six self-improvement iterations, the four-task Isaac Lab average rises monotonically from 33.1% to 96.3%, with per-task success ranging from 92.5% to 100.0% at the final iteration. This exceeds TR-DRL-DP and RECAP by 83.8 and 46.9 percentage points, respectively, and the gap is reflected in non-overlapping pooled 95% Wilson intervals at the final iteration.

Robosuite benchmarks. In Figure 4, we also evaluate Auto-E2H on two Robosuite task pairs with reversible dynamics and asymmetric difficulty: nut assembly/disassembly and peg insertion/removal. We directly compare with TR-DRL rather than TR-DRL-DP, as the environments follow the released TR-DRL settings. Starting from 0% success at iteration 1, Auto-E2H rises to 85.0% two-task average at iteration 6 (88.0% nut, 82.0% peg), exceeding TR-DRL by 17.0 percentage points. Compared with this online RL technique, the supervised policy-update signal of Auto-E2H produces stronger hard-task policies while remaining compatible with stable real-world rollouts.

Real-world robot experiments. In Figure 5, we apply the same iterative protocol to four real-world manipulation tasks: Disk, Mouse, Test Tube, and Brush. Over four self-improvement iterations, the four-task average increases from 40.0% (iteration 1) to 70.0% (iteration 4), a 30.0 percentage-point gain over Auto-E2H's own initial policy trained on the 50 reverse demonstrations. At the final iteration, Auto-E2H exceeds the TR-DRL-DP and RECAP four-task averages by 45.0 and 52.5 percentage points, respectively, with TR-DRL-DP and RECAP collapsing on three of the four real-world tasks.

Filtering behavior comparison. Figure 6 visualizes the filtering decisions on a representative temporally reversed rollout from the real-world task suite. Auto-E2H applies a hierarchical filtering process: static segments with minimal motion are removed, low-speed segments are resampled to regularize and accelerate motion, and segments with low critic value are pruned to exclude low-progress intervals. This allows Auto-E2H to adapt the retained data to Task $\mathcal{H}$ behavior patterns and identify intervals where task progress is poor; for example, the critic value begins to drop around 12 s, corresponding to a misalignment between the wooden disk and the target rod that could lead to a skewed insertion failure. By contrast, TR-DRL's dynamic filter mainly rejects segments with gripper-object contact or fast gripper motion that fall outside the model's prediction range, while retaining many static or hesitant frames with little environmental interaction; this mismatch explains its weaker improvement signal.

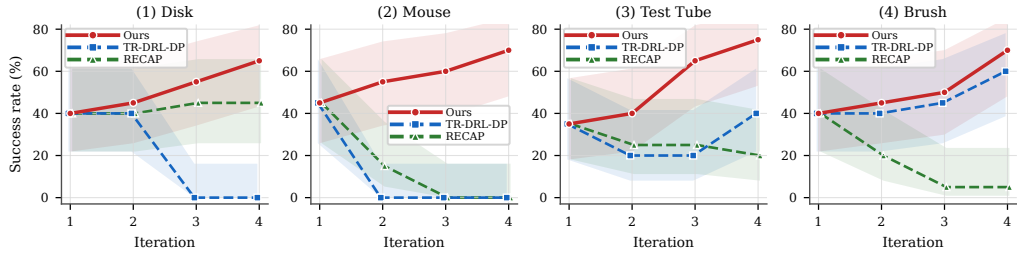

Figure 5: **Real-world robot success curves.** Each subplot reports Task $\mathcal{H}$ success rate over self-improvement iterations.

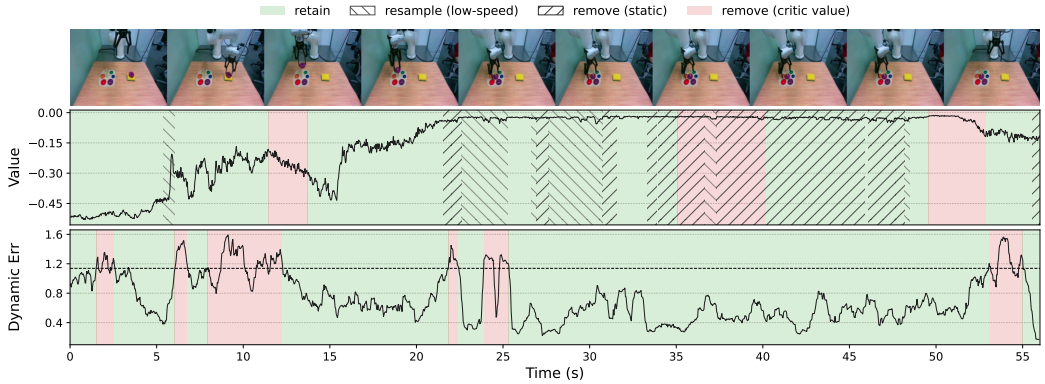

Figure 6: **Filtering comparison on a temporally reversed Disk environment rollout.** The top row shows uniformly sampled fixed-camera frames from the episode. The middle and bottom rows compare the filter scores used by our method and the baseline: green regions indicate retained timesteps, forward-hatched regions indicate static removal, backward-hatched regions indicate low-speed resampling, and red regions indicate critic- or dynamics-based removal.

4.2 Ablation Studies

We ablate Auto-E2H on four Isaac Lab tasks by removing the kinematic filter (w/o kin.), removing the critic-guided filter (w/o critic), or replacing reversed Task $\mathcal{E}$ data with successful Task $\mathcal{H}$ rollouts (Direct-$\mathcal{H}$). Figure 7 shows that the full pipeline is the most stable across tasks. At the final iteration, the full Auto-E2H reaches 96.3% average success, compared with 28.1% without kinematic filtering, 44.4% without critic-guided filtering, and 16.3% with direct Task $\mathcal{H}$ self-training. The components are complementary: kinematic filtering removes static or low-speed artifacts, the critic selects progress-making reversed segments, and reverse-direction data avoids the sparsity of early hard-task successes.

4.3 Data Collection Strategies

Auto-E2H also reduces the human effort required to collect useful real-world data. We compare collecting 50 reversed easy-task demonstrations, used by our method, with directly collecting 50 hard-task demonstrations, using the same Diffusion Policy backbone and training setup as the iteration-1 Auto-E2H model. Table 1 reports the average active recording time per demonstration, computed from the raw dataset captured at 30 Hz. The "Ours easy demos" column shows the success rate of Auto-E2H after the final iteration (Figure 5), while the "Direct hard demos" column corresponds to a policy trained from 50 hard-task demonstrations with the same model architecture and hyperparameters. Across all four tasks, Auto-E2H not only reduces collection time by 50–70% per demonstration but also improves final task success by 10–20 percentage points.

Across the four tasks, Auto-E2H reduces total operator time by about 60% compared with hard-demo BC, including failed-reset recovery.

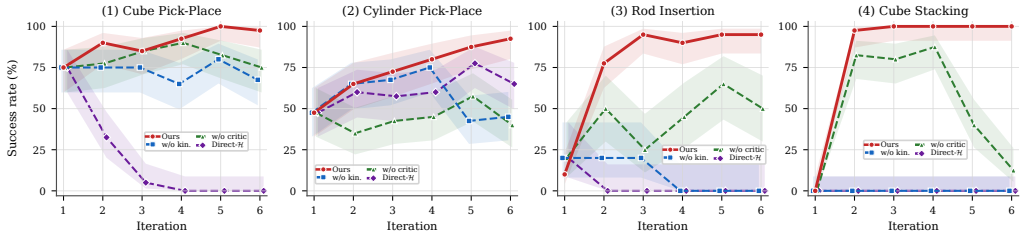

Figure 7: **Ablation success curves.** We conduct ablation studies on the Isaac Lab environments. Each subplot reports Task $\mathcal{H}$ success rate over self-improvement iterations.

Table 1: **Data collection strategy comparison.** Each task uses 50 demonstrations. Time reduction is $1 - t_{\text{ours}}/t_{\text{direct}}$; success gain is in percentage points (pp).

	Ours easy demos		Direct hard demos		Ours advantage	
Scene	Time/demo	Success	Time/demo	Success	Time reduction	Success gain
Disk	**19.5 s**	**65%**	41.5 s	50%	**53.1%**	**+15 pp**
Mouse	**24.1 s**	**70%**	54.7 s	60%	**55.9%**	**+10 pp**
Test Tube	**19.6 s**	**75%**	39.1 s	55%	**50.0%**	**+20 pp**
Brush	**19.0 s**	**70%**	62.8 s	60%	**69.7%**	**+10 pp**

5 Conclusion

We introduced Auto-E2H, a teleoperation-cost efficient framework for learning hard manipulation tasks by leveraging temporal reversal of easier tasks. By exploiting the inherent asymmetry between hard and easy tasks, Auto-E2H not only reduces dependence on costly expert demonstrations but also leverages reversed trajectories for rich supervision. To achieve this, the framework integrates three complementary components: closed-loop data collection, which autonomously generates diverse trajectories; hierarchical data refinement with kinematic and critic-guided filtering to retain high-quality segments; and iterative policy learning to progressively improve the hard-task policy. Experimental results across four Isaac Lab tasks, two Robosuite tasks, and four real-world manipulation tasks show that Auto-E2H consistently outperforms strong baselines. These results highlight that temporally reversed, easy-to-hard data can serve as scalable supervision, enabling efficient and reliable learning of complex manipulation policies with minimal human intervention.

Limitations and Discussion. Despite its advantages, Auto-E2H has limitations and is primarily suitable for directionally asymmetric tasks, where the reverse task is easier but still provides useful geometric supervision. It cannot fully handle general hard-task learning because reversed rollouts are inherently noisy rather than exact demonstrations, as contact, friction, gravity, and grasp/release events can break temporal reversibility. Pose-based reversal remains useful under moderate contact mismatch but degrades under strong direction-dependent resistance; force-aware or dynamics-aware control is needed beyond this regime. Dynamics-based reversal is further constrained because reliable world and inverse-dynamics models are difficult to learn in high-dimensional, contact-rich manipulation. Auto-E2H partially mitigates these issues through kinematic and critic-guided filtering, and closed-loop collection naturally diversifies easy-task rollouts via states induced by the evolving hard policy. Future work could extend its applicability by integrating stronger dynamics-based reversal, task-progress critics, and explicit diversity objectives for easy-task trajectories.

Acknowledgments and Disclosures

This work is supported in part by the Research Grants Council (RGC) of the Hong Kong SAR under the General Research Fund (17203023), the Collaborative Research Fund (C5052-23G), the NSFC/RGC Collaborative Research Scheme (CRS_HKU703/24) and HKU Shanghai Intelligent Computing Research Center. The research work described in this paper was conducted in the JC STEM Lab of Multimedia and Machine Learning funded by The Hong Kong Jockey Club Charities Trust. We thank Selina Wang for discussions on the early idea of incremental data collection.

The authors declare no other competing interests relevant to this work.

References

[1] Shuanghao Bai, Wanqi Zhou, Pengxiang Ding, Wei Zhao, Donglin Wang, and Badong Chen. Rethinking latent redundancy in behavior cloning: An information bottleneck approach for robot manipulation. In *International Conference on Machine Learning*, pages 2560–2580. PMLR, 2025.

[2] Brett Barkley, Amy Zhang, and David Fridovich-Keil. An investigation of time reversal symmetry in reinforcement learning. In *6th Annual Learning for Dynamics & Control Conference*, pages 68–79. PMLR, 2024.

[3] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4788–4795. IEEE, 2024.

[4] Shang-Fu Chen, Co Yong, and Shao-Hua Sun. Restoring noisy demonstration for imitation learning with diffusion models. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[5] Peng Cheng, Xianyuan Zhan, Wenjia Zhang, Youfang Lin, Han Wang, Li Jiang, et al. Look beneath the surface: Exploiting fundamental symmetry for sample-efficient offline RL. *Advances in Neural Information Processing Systems*, 36:7612–7631, 2023.

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[7] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*. Robotics: Science and Systems Foundation, 2024.

[8] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[9] Dylan J Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. *Advances in Neural Information Processing Systems*, 37: 120602–120666, 2024.

[10] Jensen Gao, Annie Xie, Ted Xiao, Chelsea Finn, and Dorsa Sadigh. Efficient data collection for robotic manipulation via compositional generalization. In *RSS 2024 Workshop: Data Generation for Robotics*, 2025.

[11] Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. In *International Conference on Learning Representations*, 2019.

[12] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[13] Yunpeng Jiang, Jianshu Hu, Paul Weng, and Yutong Ban. Time reversal symmetry for efficient robotic manipulations in deep reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[14] Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, et al. GR-RL: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025.

[15] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2024.

[16] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

[17] Shaunak A Mehta, Yusuf Umut Ciftci, Balamurugan Ramachandran, Somil Bansal, and Dylan P Losey. Stable-BC: Controlling covariate shift with stable behavior cloning. *IEEE Robotics and Automation Letters*, 10(2):1952–1959, 2025.

[18] Suraj Nair, Mohammad Babaeizadeh, Chelsea Finn, Sergey Levine, and Vikash Kumar. TRASS: Time reversal as self-supervision. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 115–121. IEEE, 2020.

[19] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[20] Fumihiro Sasaki and Ryota Yamashina. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2020.

[21] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[22] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

[23] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4950–4957, 2018.

[24] Boyang Wang, Nikhil Sridhar, Chao Feng, Mark Van der Merwe, Adam Fishman, Nima Fazeli, and Jeong Joon Park. This&that: Language-gesture controlled video generation for robot planning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12842–12849. IEEE, 2025.

[25] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. In *RSS 2024 Workshop: Data Generation for Robotics*, 2024.

[26] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[27] Xiangtong Yao, Zhenshan Bing, Genghang Zhuang, Kejia Chen, Hongkuan Zhou, Kai Huang, and Alois Knoll. Learning from symmetry: Meta-reinforcement learning with symmetrical behaviors and language instructions. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5574–5581. IEEE, 2023.

[28] Maryam Zare, Parham M Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 54(12):7173–7186, 2024.

[29] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A Implementation Details

All learned policies use an image-conditioned Diffusion Policy [6] with DDIM sampling [22]. The policy observes two history steps from a table camera, a wrist camera, and a 15-dimensional state containing end-effector pose, object pose, and gripper state. In the real-robot experiments, the seven object-pose entries are fixed to zero during both training and inference. The policy uses only the two RGB images, the 7-D end-effector pose, and the 1-D gripper state; it does not require object-pose estimates or privileged object state. The action is an 8-dimensional end-effector goal pose plus gripper command. Data are recorded at 30 Hz, and images are cropped to 128×128 during policy training. The visual backbone uses a ResNet-18 image encoder. The action horizon and executed action chunk both contain sixteen steps. The diffusion model uses 100 training noise steps and ten DDIM denoising steps at inference. The conditioning-dropout probability is 0.3.

Policy training uses Adam with learning rate 2×10^{-4}, seed 42, and batch size 256 per GPU. The step count is scaled to preserve the same total sample budget under different GPU counts. On the four NVIDIA A100 GPUs used in our experiments, each initial Task $\mathcal{E}$ and Task $\mathcal{H}$ policy is trained for 4000 optimizer steps, and each incremental Task $\mathcal{H}$ update adds another 4000 optimizer steps. At iteration n, the hard-task policy is warm-started from $\pi_h^{(n-1)}$ using $\mathcal{D}_h^{(0)} \cup \mathcal{D}_{\text{filt}}^{(n)}$: the fixed initial reversed demonstrations and the current iteration's accepted data, sampled with a 1:1 weighting. The Task $\mathcal{E}$ policy is trained once and then kept fixed.

The critic is trained at each iteration from the current Task $\mathcal{H}$ rollout data with Monte Carlo return labels and an 80/20 train/validation split. It uses the same two-step visual/state observation format, a ResNet-18 image encoder initialized from the current action policy, an MLP with hidden dimensions $(512, 512, 256, 256)$, MSE value loss, Adam learning rate 5×10^{-4}, weight decay 10^{-4}, gradient clipping at 1.0, color jitter augmentation, dropout 0.15, crop size 112×112, batch size 256, and 500 training steps.

Filtering parameters. We use one fixed set of filtering parameters across all simulation and real-world environments. The static filter uses end-effector speed threshold $\epsilon_s = 0.04\,\mathrm{m/s}$, spatial-drift threshold $\epsilon_d = 0.05$ m, and minimum static length $L_{\mathrm{s}} = 16$ frames. The speed filter uses low-speed threshold $\epsilon_l = 0.06\,\mathrm{m/s}$ and resamples selected low-speed segments toward target step size $\Delta H \approx 3.3 \times 10^{-3}$ m at 30 Hz, equivalent to a target speed of $0.1\,\mathrm{m/s}$. For critic-guided filtering, we use $\gamma = 1.0$, $\lambda = 0$, smoothing window 31, set the advantage drop threshold α_{drop} to the 30th percentile of smoothed advantage scores, and use minimum keep/drop length 30.

TR-DRL-DP. For the Isaac Lab comparison labeled TR-DRL-DP, we do not run the original SAC-based TR-DRL algorithm. Instead, we keep the same closed-loop rollout collection and Diffusion Policy imitation-learning loop as our method, and replace our refinement-and-reversal module with a dynamics-model pipeline adapted from TR-DRL. At each iteration, hard-task rollouts are encoded into visual-state features and used to train a forward dynamics model and an inverse dynamics model for 2000 steps each. For every adjacent pair in an easy-task rollout, the inverse model predicts the reverse action from the later observation to the earlier observation, and the forward model checks that action by reconstructing the earlier feature/state. The resulting cycle-consistency error is smoothed and thresholded at the 80th percentile for that iteration; transitions above the threshold are discarded, and retained runs shorter than 16 frames are removed. The accepted easy-task segments are then temporally reversed. This reversal is not implemented by simply reusing the recorded Task $\mathcal{E}$ actions in reverse order: for a reversed step from original frame t to frame $t-1$, the action label is predicted as $\mathrm{IDM}(o_t, o_{t-1})$, with the gripper dimension copied from the recorded gripper state. The original RGB observations are kept, and the resulting hard-task-direction data are mixed with the original demonstrations to finetune the Diffusion Policy. Thus this baseline isolates a TR-DRL-DP inside our imitation-learning setting, while the Robosuite TR-DRL comparison uses the original SAC-based implementation.

Policy and critic training were run on a four-GPU NVIDIA A100 server. One Isaac Lab policy-update iteration, including rollout collection, filtering, critic training, and policy finetuning, typically took 1.5–3 hours depending on the task.

B Evaluation Protocol and Statistical Reporting

We evaluate Auto-E2H on four Isaac Lab task pairs, two Robosuite task pairs, and four real-world manipulation tasks. The reported success curves are computed from success counts rather than smoothed training metrics. The initial human-demonstration budgets are 40 easy-direction demonstrations for each Isaac Lab task pair, 10 demonstrations per Robosuite task pair, and 50 demonstrations per real-world task. After initialization, no additional human demonstrations are collected. Each self-improvement iteration instead adds policy rollouts: 40 cyclic Task $\mathcal{H}$/Task $\mathcal{E}$ rollout pairs per Isaac Lab task, 50 rollout episodes per Robosuite task, and 20 hard-task rollout episodes per real-world task. Successful Task $\mathcal{E}$ rollouts provide candidate reversed data, while all Task $\mathcal{H}$ rollouts are used for critic training. The rollout horizon is 3000 for cube pick-place, cylinder pick-place, and cube stacking, and 200 for rod insertion.

For the benchmark curves, each Isaac Lab task uses 40 evaluation trials, and each Robosuite task uses 50 trials. For real-world experiments, each success rate is computed over 20 trials. The shaded regions in the benchmark, real-world, and ablation success-curve figures show Wilson binomial confidence intervals ($z = 1.96$), computed from the success-over-trial counts used by each plotted point.

Pooled suite-level confidence intervals. To complement the per-task curves, Table 2 reports pooled Task $\mathcal{H}$ success for Auto-E2H at the initial and final self-improvement iterations of each suite, with 95% Wilson binomial confidence intervals computed from the pooled successes-over-trials counts. The initial- and final-iteration intervals do not overlap in any of the three suites.

Table 2: Pooled Auto-E2H Task $\mathcal{H}$ success and 95% Wilson binomial confidence intervals at the initial and final self-improvement iterations. Here k/n denotes successful trials over total evaluation trials; counts are summed across all tasks within each suite.

Suite	Initial (k/n)	Initial 95% CI	Final (k/n)	Final 95% CI
Isaac Lab (4 tasks)	53/160 (33.1%)	[26.3%, 40.7%]	154/160 (96.3%)	[92.1%, 98.3%]
Robosuite (2 tasks)	0/100 (0.0%)	[0.0%, 3.7%]	85/100 (85.0%)	[76.7%, 90.7%]
Real-world (4 tasks)	32/80 (40.0%)	[30.0%, 51.0%]	56/80 (70.0%)	[59.2%, 78.9%]

C Filter-Comparison Visualizations

Figure 8–17 provide per-scene examples of the filtering decisions used during easy-to-hard data conversion. Each visualization shows sampled rollout frames, the critic value used by our filter, and the dynamics error used by the TR-DRL-DP. Green regions indicate retained transitions, red regions indicate removed transitions, and the hatched regions mark segments removed as static or resampled due to low-speed motion. The scenes are ordered as four real-world tasks, four Isaac Lab tasks, and two Robosuite tasks.

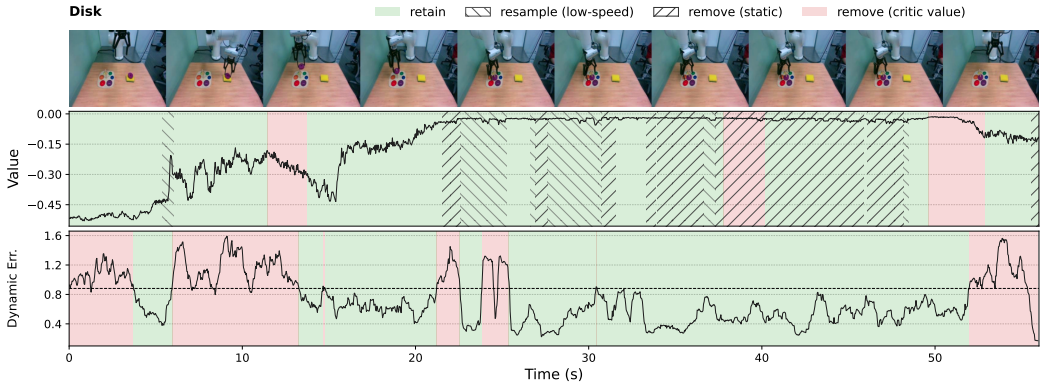

Figure 8: Filter-comparison visualization for Real Disk.

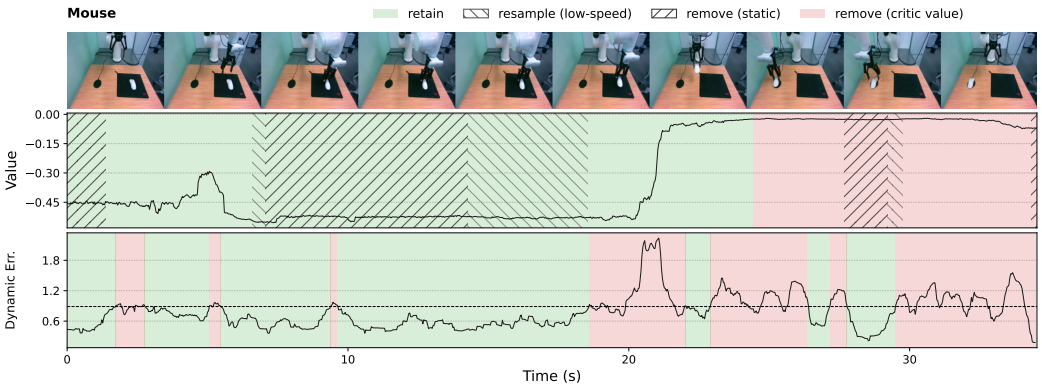

Figure 9: Filter-comparison visualization for Real Mouse.

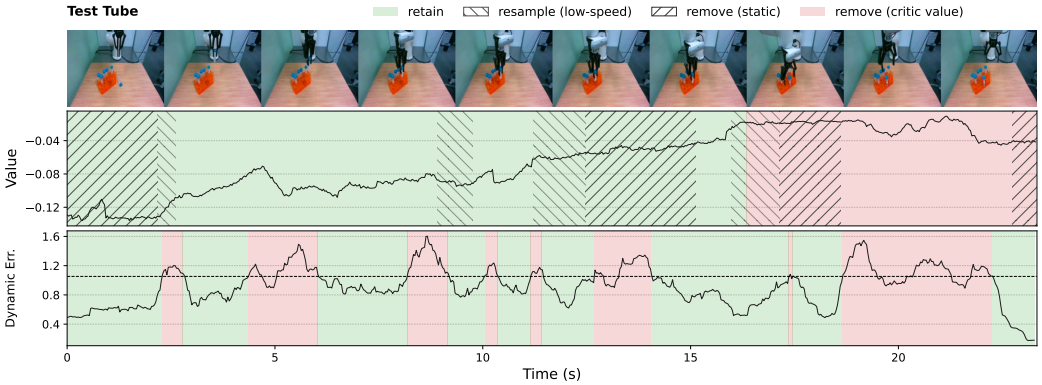

Figure 10: Filter-comparison visualization for Real Test Tube.

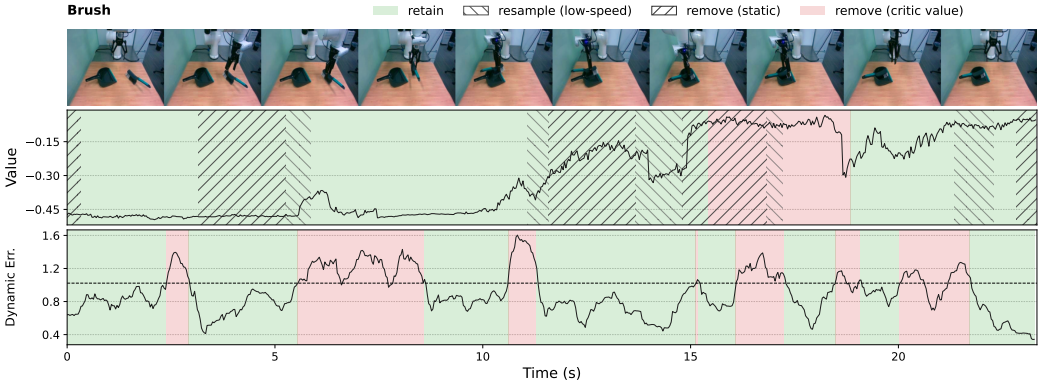

Figure 11: Filter-comparison visualization for Real Brush.

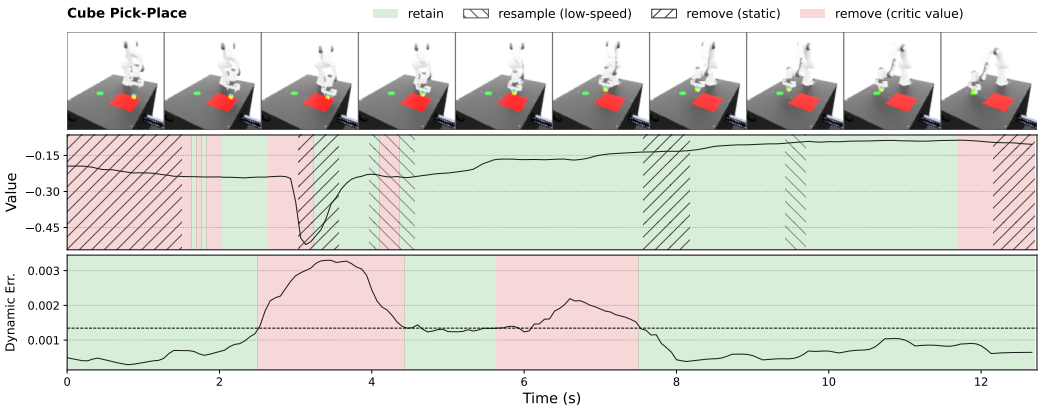

Figure 12: Filter-comparison visualization for Isaac Cube Pick Place.

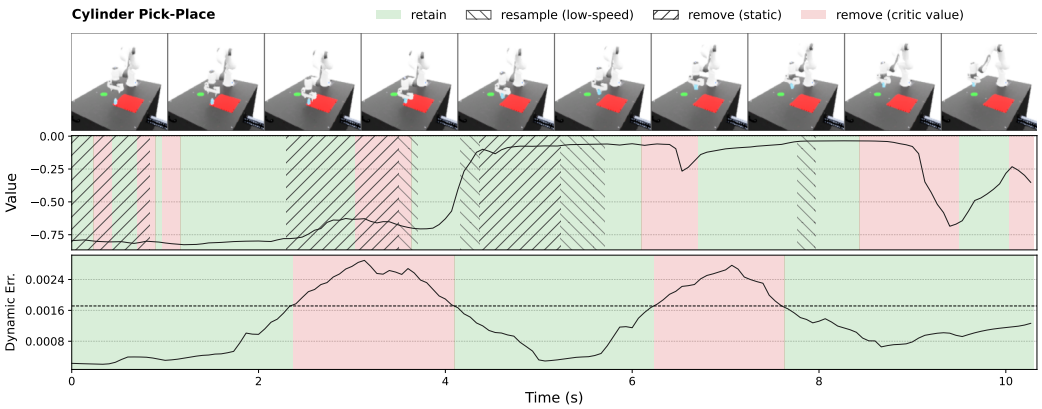

Figure 13: Filter-comparison visualization for Isaac Cylinder Pick Place.

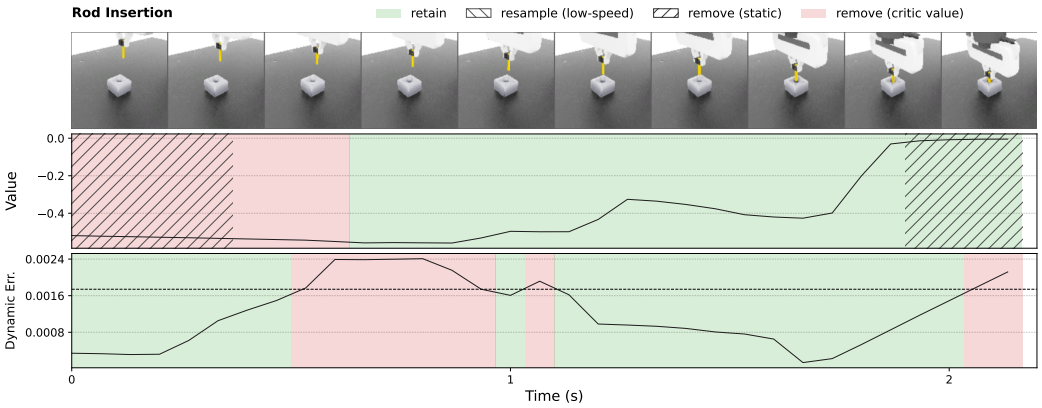

Figure 14: Filter-comparison visualization for Isaac Rod Insertion.

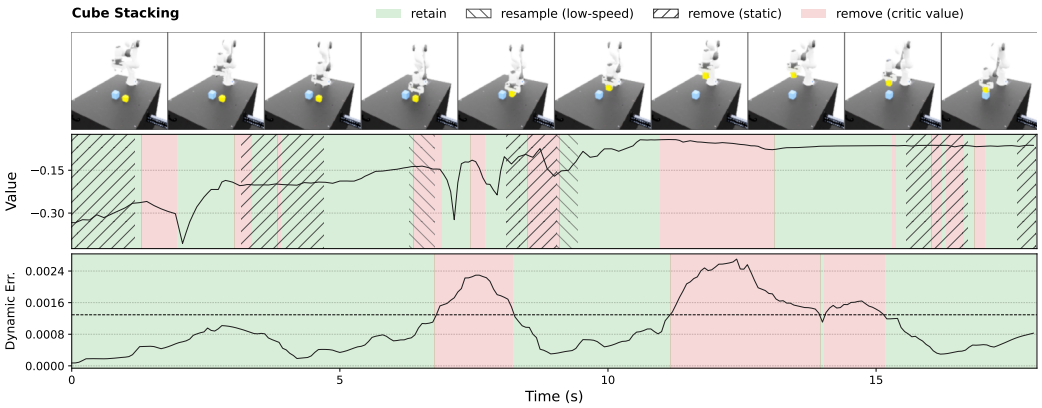

Figure 15: Filter-comparison visualization for Isaac Cube Stack.

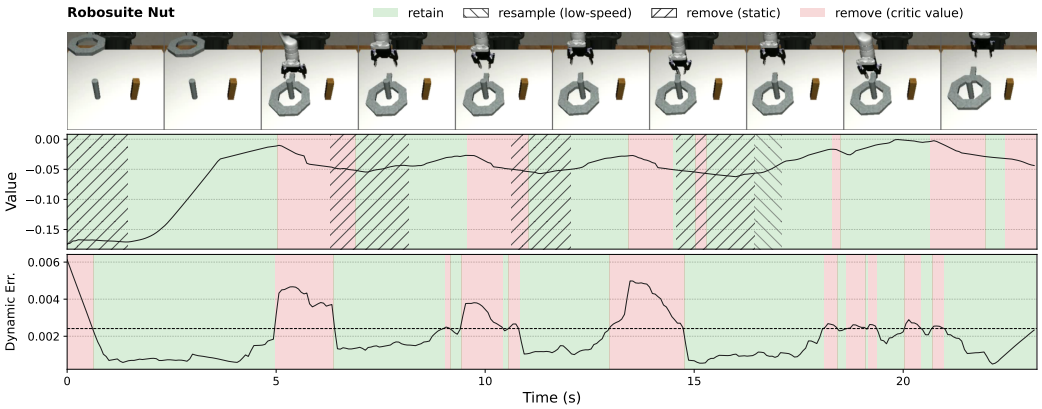

Figure 16: Filter-comparison visualization for Robosuite Nut.

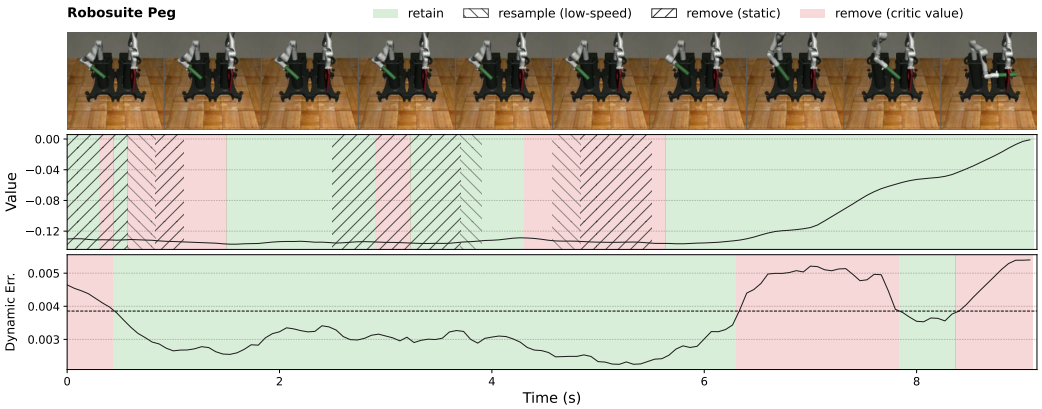

Figure 17: Filter-comparison visualization for Robosuite Peg.

D Critic Analysis

We study critic filtering and update choices on four Isaac Lab tasks: cube pick-place, cylinder pick-place, rod insertion, and cube stacking. The comparisons use the same Diffusion Policy backbone and training budget, six policy iterations, and 40 evaluation trials per reported success rate.

Cold-start supervision. For success indicator $y \in \{0, 1\}$, rollout length $T \leq H$, horizon H, and frame index t, the critic target is $\bar{G}_t = -(T-t)/(2H) - (1-y)/2$. It lies in $[-0.5, 0)$ for successful rollouts and $[-1, -0.5)$ for failed rollouts. If every rollout fails at the maximum horizon, neighboring frames still differ by $1/(2H)$ in their targets. This provides temporal-ordering supervision, but does not by itself establish that the critic is a calibrated success predictor or a reliable task-progress measure in every state. Action reconstruction and kinematic filtering also contribute to cold start before critic filtering is applied.

Early filtering and critic updates. Table 3 reports two controlled comparisons on the four Isaac Lab tasks: bypassing critic filtering only in iterations 1–2, and fine-tuning the critic from the previous iteration rather than retraining it. The early-filter comparison keeps reversal, kinematic filtering, training budget, evaluation starts, and seed fixed. Removing the early critic filter reduces iteration-2 success from 77.5% to 50% on rod insertion and from 100% to 82.5% on stacking. After filtering is restored, the final differences are only 0–2.5 percentage points across tasks. Retraining is generally slightly better in intermediate iterations, while final results are comparable; we retain retraining as the default.

Table 3: Critic-filter and update ablations. Entries are hard-task success rates (%), each over 40 evaluation trials. "No early filter" bypasses critic filtering only in iterations 1–2; "Warm-start critic" fine-tunes the preceding critic.

Task	Variant	Iter. 1	2	3	4	5	6
Cube pick-place	Full method	75	95	85	92.5	100	97.5
	No early filter	75	82.5	85	92.5	100	100
	Warm-start critic	77.5	82.5	87.5	90	95	100
Cylinder pick-place	Full method	45	65	75	80	80	85
	No early filter	40	65	70	77.5	80	82.5
	Warm-start critic	42.5	65	72.5	75	82.5	82.5
Rod insertion	Full method	10	77.5	95	90	95	95
	No early filter	10	50	87.5	90	95	95
	Warm-start critic	10	67.5	90	90	92.5	95
Cube stacking	Full method	0	100	100	100	100	100
	No early filter	0	82.5	100	100	100	100
	Warm-start critic	0	92.5	100	100	100	100

E Training-Seed Variability

We measure the training variability of Auto-E2H on cube pick-place, cylinder pick-place, rod insertion, and cube stacking in Isaac Lab. Table 4 reports three independent training seeds (42, 0, and 1), with 40 evaluation trials per entry, across six policy iterations. Final success is 97.5–100% for cube pick-place, 82.5–92.5% for cylinder pick-place, 95–97.5% for rod insertion, and 100% for cube stacking. These runs are reported separately from the original benchmark curves. Differences across training seeds measure training variability; a Wilson interval for one policy's evaluation trials instead describes binomial evaluation uncertainty, rather than variability across training seeds.

Table 4: Training-seed variability on Isaac Lab. Entries are hard-task success rates (%), with 40 evaluation trials each.

Task	Training seed	Iter. 1	2	3	4	5	6
Cube pick-place	42	75	95	85	92.5	100	97.5
	0	62.5	75	87.5	92.5	100	100
	1	67.5	85	87.5	95	95	100
Cylinder pick-place	42	45	65	75	80	80	85
	0	40	62.5	65	75	80	82.5
	1	50	70	77.5	77.5	85	92.5
Rod insertion	42	10	77.5	95	90	95	95
	0	12.5	80	82.5	90	95	95
	1	2.5	70	87.5	90	92.5	97.5
Cube stacking	42	0	100	100	100	100	100
	0	2.5	97.5	100	100	100	100
	1	2.5	100	100	100	100	100

F Data Collection and Training Costs

Each Auto-E2H run uses 50 easy-task demonstrations and 60 autonomous collection episodes; hard-demo BC uses 50 hard-task demonstrations. Failed resets require 21–45 s of operator recovery per task over the full run. Including recovery, Auto-E2H reduces total operator time by about 60% across the four real-world tasks (Table 5).

Table 5: Per-task time costs in seconds. Operator time includes demonstration collection and failed-reset recovery. Training time sums initial policy training, critic training, and policy updates. Autonomous rollout time is for Auto-E2H.

	Operator time		Auto rollout	Training time	
Task	Auto-E2H	Hard-demo BC	Auto-E2H	Auto-E2H	Hard-demo BC
Disk	997	2077	2472	39151	35048
Mouse	1251	2734	1909	40328	37926
Test Tube	1022	1957	1365	39104	35109
Brush	971	3141	1030	36176	39148

G Boundary of Pose-Based Reversal under Contact Resistance

Auto-E2H reconstructs motion geometry rather than interaction dynamics. Contact wrenches depend on friction, compliance, contact mode, and loading direction, and generally cannot be recovered by reversing their order or sign. The action space contains no force command and the policy receives no force-torque input. The following controlled experiments quantify the applicability boundary in the tested tasks; their force values are not universal thresholds for other assemblies or controllers.

G.1 Controlled Simulation

We fix geometry, demonstrations, and controller in Isaac Sim rod insertion and apply a simulated friction force opposite to instantaneous rod motion, with direction updated as the rod moves rather than constrained to its axis. Reconstructed actions are first replayed on 100 paired initial states (seed 42), without policy training. Extraction succeeds throughout, while reversed insertion declines sharply above 2 N (Table 6). Extraction produces a tension-like load that tends to preserve alignment, whereas insertion produces a compression-like load that can amplify angular error and rod–gripper slip. Every replay failure above 2 N involves more than 5 mm of slip. The complete learning pipeline follows the same trend (Table 7). At up to 2 N, iteration-6 success reaches 92.5–95%; at 4 and 6 N it falls to 17.5% and 0%.

Table 6: Direct replay under simulated contact resistance, without policy training. Entries are successful trials out of 100 paired initial states.

Simulated friction force (N)	Extraction	Reversed replay
0	100/100	100/100
0.5	100/100	100/100
1	100/100	100/100
2	100/100	97/100
4	100/100	21/100
6	100/100	17/100

Table 7: Complete Auto-E2H learning under simulated resistance. Entries are hard-task success rates (%), each over 40 evaluation trials.

Resistance (N)	Iter. 1	2	3	4	5	6
0	10	77.5	95	90	95	95
0.5	5	67.5	90	90	92.5	95
1	7.5	72.5	85	87.5	92.5	92.5
2	5	65	87.5	90	87.5	92.5
4	0	5	12.5	15	12.5	17.5
6	0	0	0	0	2.5	0

G.2 Real-Robot Disk Insertion

The hard direction grasps a disk and inserts it over a vertical rod. The untaped rod diameter is 13.91 mm and the disk-hole diameter is 16.14 mm. We progressively increase contact resistance by wrapping the rod in 0.1-mm-thick paper masking tape; each full wrap increases diameter by approximately 0.2 mm, with realized diameters measured after wrapping. The wrist force-torque sensor is used for analysis, not as a policy input. For successful insertions, the peak-force statistic is the mean of the 30 largest force magnitudes among frames with nonzero vertical motion. It is an empirical proxy for contact resistance, not an estimate of the friction coefficient.

Table 8 reports success over 20 evaluation trials per entry. At 0–6 wraps (3.4–3.6 N), reversed data remains useful, reaching 60–70% final success. At 7 wraps (10.1 N), performance degrades substantially, although additional reversed data improves success from 20% to 30%. At 8 wraps no insertion succeeds and the robot triggers its safety stop. The load then reflects compression from a jammed fit rather than sliding resistance, and no successful trajectory exists from which to compute the statistic, so it is reported as N/A. These results motivate force-aware or dynamics-aware extensions for strongly direction-dependent contact.

Table 8: Real-robot contact boundary. The peak-force statistic is computed only from successful insertions. Iteration columns report success rates (%), each over 20 trials.

Peak force (N)	Tape wraps	Rod diameter (mm)	Iter. 1	2	3	4
3.4	0	13.91	40	45	55	65
3.5	3	14.51	40	45	60	70
3.6	6	15.09	35	45	55	60
10.1	7	15.46	20	20	25	30
N/A	8	15.78	0	0	0	0

H Multi-Robot Operator-Load Estimate

We estimate a possible concurrent-deployment workload from the measured single-robot audit; this is not a multi-robot deployment experiment. Assume each of N robots reproduces the single-robot rollout time T_{auto} and intervention time T_{int}, and all robots collect concurrently, without credit for staggered scheduling or operator-free fine-tuning intervals. The measured $(T_{\mathrm{auto}}, T_{\mathrm{int}})$ pairs, in seconds, are $(2472, 21)$ for Disk, $(1909, 45)$ for Mouse, $(1365, 42)$ for Test Tube, and $(1030, 23)$ for Brush. The implied average utilization of one shared operator is

$$\rho_N = N \frac{T_{\mathrm{int}}}{T_{\mathrm{auto}}}. \tag{11}$$

For ten robots, this gives 8.5% for Disk, 23.6% for Mouse, 30.8% for Test Tube, and 22.3% for Brush. These averages do not bound simultaneous alerts or waiting times. A proposed implementation can stagger rollout and fine-tuning cycles and queue remaining alerts, pausing only the robot awaiting recovery while the others continue collecting. This scheduling proposal may reduce peak contention, but its benefit is not measured by the single-robot experiment.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and introduction clearly describe the contributions, including automated data collection, hierarchical refinement, and iterative policy learning for hard manipulation tasks. Experimental results support these claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses limitations: Auto-E2H is mainly effective for directionally asymmetric tasks; dynamics-based reversal is limited in high-dimensional, contact-rich tasks.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results or proofs; it focuses on Robotic manipulations and experimental evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient details to reproduce the experiments, including task setups, demonstration budgets, Diffusion Policy architecture, filtering thresholds, and the iterative training procedure.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Core algorithm code is provided in the supplementary material.

   Guidelines:

   - The answer [Yes] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and test details, including data splits, hyperparameters, and optimizer settings, are clearly specified or referenced in appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main success curves report 95% Wilson binomial confidence intervals with evaluation trial counts (Appendix B). Appendix E separately reports three training seeds for the four Isaac Lab tasks; these measure training variability rather than binomial evaluation uncertainty.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The type of compute (GPU), memory, and iteration counts are described, enabling reproduction of training runs.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to NeurIPS Code of Ethics; no ethical violations are involved.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive impacts (reducing human teleoperation) and potential negative impacts (limitations in high-precision or contact-rich environments).

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The work does not involve high-risk models or datasets requiring special safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the original publications for Isaac Lab, Robosuite, and the baseline methods.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The supplementary code is documented in a README.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or human subject experiments are conducted.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are involved; IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: No LLM is used in the core methodology; if used at all, it is only for polishing purposes.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.