ASYNCHRONOUS SIM-REAL CO-TRAINING FOR ON-POLICY ROBOT LEARNING WITH LIMITED REAL HARDWARE

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving an already capable robot policy through online interaction is challenging when physical hardware and interaction opportunities are limited. Simulation can supply additional experience, but differences in collection speed and availability across domains complicate batch composition and data freshness during joint online learning. We present Asynchronous Sim-Real Co-training (ASRC), a framework that uses continually collected simulated and real-world experience for PPO-style policy improvement. ASRC decouples collection from policy optimization, allowing independent interaction in each domain while learning proceeds. To coordinate their heterogeneous experience, an availability-aware scheduler adjusts the real-data fraction within a prescribed range, while latest-first selection and bounded-buffer eviction prioritize recent samples. The framework supports both CNN policy optimization and residual adaptation of $\pi_{0.5}$. Across three manipulation tasks on a single physical robot, both policy families achieve final real-world success rates above 90% on every task, outperforming the evaluated baselines. System experiments show up to $2.96\times$ higher update throughput than a synchronous counterpart, while ablations support the importance of balancing domain composition and prioritizing fresh experience.

1 INTRODUCTION

Imitation learning from demonstrations has enabled robot policies to perform a wide range of complex manipulation tasks (Chi et al., 2025; Zhao et al., 2023; Black et al., 2024; Kim et al., 2024; Team et al., 2024; Bjorck et al., 2025; Liu et al., 2025). Yet, limited demonstration coverage and compounding execution errors can drive a policy into states that are poorly represented in its training data, leading to failures (Ross et al., 2011; Liu et al., 2026). Collecting expert labels at policy-visited states and retraining offers one path to improvement (Ross et al., 2011), but requires additional data collection and substantial expert annotation. This motivates a practical question: *how can we build on an already capable policy to improve its real-world task success when physical hardware and interaction opportunities are limited?*

Online reinforcement learning (RL) offers a natural approach (Haarnoja et al., 2018; Fujimoto et al., 2018). Several real-world robotic RL methods use off-policy algorithms to improve sample efficiency by reusing past experience (Hu et al., 2023; Luo et al., 2025; Xu et al., 2026a; Wagenmaker et al., 2025; Wu et al., 2025; Xu et al., 2026b; Xiao et al., 2026). However, when fine-tuning a capable pretrained policy with off-policy actor–critic methods, an insufficiently trained action-value function can misguide policy updates and degrade existing capabilities (Uchendu et al., 2023; Yuan et al., 2025).

Instead of off-policy RL, we explore PPO-style on-policy optimization (Schulman et al., 2017) as an alternative approach to incremental policy improvement. Prior work has demonstrated the feasibility of fine-tuning pretrained robot policies with PPO (Ren et al., 2025; Zang et al., 2026a; Zhang et al., 2026; Chen et al., 2025). However, on-policy methods limit the reuse of experience from older policies, which can increase the amount of real-world interaction needed for policy improvement (Schulman et al., 2017). Collecting this experience on physical robots incurs substantial time and operational effort (Zang et al., 2026b).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Simulation offers a scalable source of additional interaction. Nevertheless, visual and physical discrepancies can impede transfer (Wei et al., 2025; Yin et al., 2025). To improve real-world performance, prior work has trained policies on mixed simulated and real demonstrations (Maddukuri et al., 2025; Lei et al., 2026b; Wei et al., 2025), combined simulated RL with supervision from real demonstrations (Shi et al., 2026), or used simulation-trained value functions and policies to guide subsequent real-world RL (Yin et al., 2025; Wu et al., 2025). These approaches either rely on fixed real-world demonstrations for co-training or separate simulation pretraining from subsequent real-world adaptation. This motivates studying online sim-real co-training, in which both environments continually provide experience for PPO-style improvement of a shared policy, combining the interaction scale of simulation with feedback from actual deployment.

In this work, we present **A**synchronous **S**im-**R**eal **C**o-training (ASRC), a framework for on-policy robot learning with limited real hardware. Unlike parallel simulation, real-world collection in our setting is constrained by robot availability, environment resets, and human feedback, resulting in unequal and varying data availability across domains. ASRC therefore decouples data collection from policy optimization, allowing simulated and real-world interaction to proceed independently with separate experience buffers. To assemble each update batch, an availability-aware scheduler adjusts the real-data fraction within a prescribed range, balancing the desired domain composition against the available samples. Meanwhile, latest-first selection prioritizes recent experience rather than continually consuming an accumulated backlog. The shared policy is optimized using a decoupled PPO objective (Hilton et al., 2022; Fu et al., 2026). We evaluate ASRC on three manipulation tasks using a single physical robot, with both CNN policies and residual adaptation of $\pi_{0.5}$ achieving final real-world success rates above 90% on every task, outperforming the evaluated baselines. System experiments further show that asynchronous execution translates additional simulation parallelism into higher training throughput, achieving up to $2.96\times$ the update throughput of a zero-staleness counterpart.

Our contributions are threefold: (1) We develop an online sim-real co-training framework that combines continually collected simulated and real-world experience for PPO-style policy improvement with limited physical hardware. (2) We combine availability-aware mixture scheduling with latest-first experience management to coordinate batch composition and data freshness under heterogeneous collection rates. (3) We evaluate CNN policies and $\pi_{0.5}$ residual adaptation on three real-world manipulation tasks, assessing policy performance, system throughput, and the effects of domain composition and sample freshness.

2 Related works

Real-world reinforcement learning Real-world RL directly optimizes policies through physical interaction. SERL (Luo et al., 2024) and HIL-SERL (Luo et al., 2025) demonstrate effective robot manipulation, with HIL-SERL incorporating corrective human interventions. Recent work extends RL to pretrained generative policies and vision-language-action models. DSRL (Wagenmaker et al., 2025) steers the latent noise of a frozen diffusion policy, but EXPO-FT reports limited gains on several real-world tasks (Dong et al., 2026). RLT (Xu et al., 2026a) trains a lightweight off-policy actor–critic using frozen VLA representations and action priors; its training involves human corrections and policy switching, adding operator effort. Similarly, EXPO-FT (Dong et al., 2026) uses human interventions for reliable, sample-efficient VLA adaptation. RL-100 (Lei et al., 2026a) combines imitation learning, iterative offline RL, and PPO-style online fine-tuning, but requires approximately 14 hours of real-world data collection per task on average. These approaches motivate reducing physical interaction costs and reliance on corrective interventions. Our method jointly learns from ongoing simulated and real-world experience to improve existing policies under limited physical interaction, without corrective human interventions during RL training.

Simulation for Real-World Robot Learning Simulation platforms provide scalable environments for robot interaction and data generation (Tao et al., 2024; Mittal et al., 2025; Zhu et al., 2020). To address visual and dynamics gaps, domain randomization varies visual and physical parameters during training (Tobin et al., 2017; Peng et al., 2018; Andrychowicz et al., 2020). Alternatively, real-to-sim-to-real approaches reconstruct scenes and objects to better match the target environment (Torne et al., 2024; Han et al., 2025). However, designing randomization ranges and closely matching simulation to reality can require substantial manual tuning. In contrast, supervised

learning on mixed simulated and real demonstrations can improve real-world performance despite substantial domain differences (Maddukuri et al., 2025). Within this paradigm, studies examine the effects of data scale, mixing ratios, and domain gaps (Wei et al., 2025), as well as cross-domain representation alignment (Cheng et al., 2026; Lei et al., 2026b). Beyond mixing demonstrations, other methods combine online simulation RL with pre-collected real-world data (Shi et al., 2026; Ma et al., 2026; Niu et al., 2022). However, their real-world training signals come from fixed datasets rather than continued physical interaction. To incorporate online real-world feedback, prior work uses simulation-learned value functions, policy priors, or interaction data to guide real-world RL (Yin et al., 2025; Wu et al., 2025; Xu et al., 2026b), or learns jointly from simulated and real environments (Di-Castro et al., 2021; Liu et al., 2023). In this work, we focus on online, on-policy sim-real learning with limited physical hardware. We coordinate cross-domain data availability and freshness using asynchronous collection, availability-aware domain mixing, and latest-first selection.

3 PRELIMINARIES

3.1 SIM-REAL LEARNING SETUP

Following prior work (Maddukuri et al., 2025; Shi et al., 2026), we consider robot manipulation tasks with paired simulated and real-world environments. We denote a task by $\mathcal{T} = \langle \mathcal{T}_{\text{sim}}, \mathcal{T}_{\text{real}} \rangle$ and model each domain as a language-conditioned, finite-horizon partially observable Markov decision process (POMDP):

$$\mathcal{T}_d = \langle \mathcal{S}_d, \mathcal{A}, \mathcal{O}_d, \mathcal{L}, P_d, \Omega, r, \mu_d, \gamma, H \rangle, \qquad d \in \{\text{sim}, \text{real}\}. \tag{1}$$

Here, $\mathcal{S}_d$ and $\mathcal{O}_d$ are the domain-specific state and observation spaces, $\mathcal{A}$ is the shared robot action space, and $\mathcal{L}$ is the space of language instructions. The two domains have corresponding task semantics. We use Ω to denote a shared observation-construction rule, whose restriction to $\mathcal{S}_d$ takes values in $\mathcal{O}_d$. The domains share the task instruction $\ell \in \mathcal{L}$, reward function $r(s, a, s', \ell)$, discount factor γ, and episode horizon H, while their transition dynamics P_d and initial-state distributions μ_d may differ. For evaluation, we define a success predicate $C_d(s, \ell) \in \{0, 1\}$ for each domain.

A shared policy π_θ interacts with either domain as follows. Starting from $s_0 \sim \mu_d(\cdot \mid \ell)$, it receives the current observation $o_t = \Omega(s_t)$ and samples $a_t \sim \pi_\theta(\cdot \mid o_t, \ell)$. The environment executes a_t, transitions to $s_{t+1} \sim P_d(\cdot \mid s_t, a_t)$, and returns reward $r(s_t, a_t, s_{t+1}, \ell)$. Let $p_d^{\pi_\theta}(\tau \mid \ell)$ denote the resulting trajectory distribution. Our ultimate objective is to maximize the probability of completing the real-world task within at most H action steps:

$$J_{\text{real}}^{\text{succ}}(\theta; \ell) = \mathbb{E}_{\tau \sim p_{\text{real}}^{\pi_\theta}(\cdot \mid \ell)} \left[\mathbf{1}\{\exists\, t \in \{0, \ldots, H\} :\ C_{\text{real}}(s_t, \ell) = 1\} \right]. \tag{2}$$

3.2 DECOUPLED PPO OBJECTIVE

We build on Proximal Policy Optimization (PPO) (Schulman et al., 2017), an on-policy reinforcement learning algorithm that updates a policy using a clipped surrogate objective. However, in our setting, collection and optimization proceed asynchronously, so a training batch may contain samples generated by different historical policy versions. Using these behavior policies as proximal references couples the update constraint to collection delays, rather than to the learner's recent policy.

To separate these roles, we adopt the decoupled PPO objective (Hilton et al., 2022; Fu et al., 2026). Let $\pi_{\text{behav},i}$ denote the policy that generated sample i, and let π_{prox} be a recent learner policy held fixed during a training iteration. Define $\rho_i(\theta) = \frac{\pi_\theta(a_i|o_i,\ell_i)}{\pi_{\text{prox}}(a_i|o_i,\ell_i)}$ and $w_i = \frac{\pi_{\text{prox}}(a_i|o_i,\ell_i)}{\pi_{\text{behav},i}(a_i|o_i,\ell_i)}$. The asynchronous update maximizes

$$L_{\text{dec}}(\theta) = \widehat{\mathbb{E}}_i \left[w_i \min \left(\rho_i(\theta)\hat{A}_i,\ \text{clip}(\rho_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i \right) \right]. \tag{3}$$

Here, $\hat{A}_i$ is an estimated advantage, ϵ is the clipping threshold, and $\widehat{\mathbb{E}}_i$ denotes the empirical average over batch samples. The weight w_i accounts for the action-probability mismatch between the behavior and proximal policies, while clipping ρ_i constrains updates relative to the recent reference. When all behavior policies coincide with π_{prox}, $w_i = 1$ and the objective reduces to standard PPO. This objective provides the optimization backbone of ASRC.

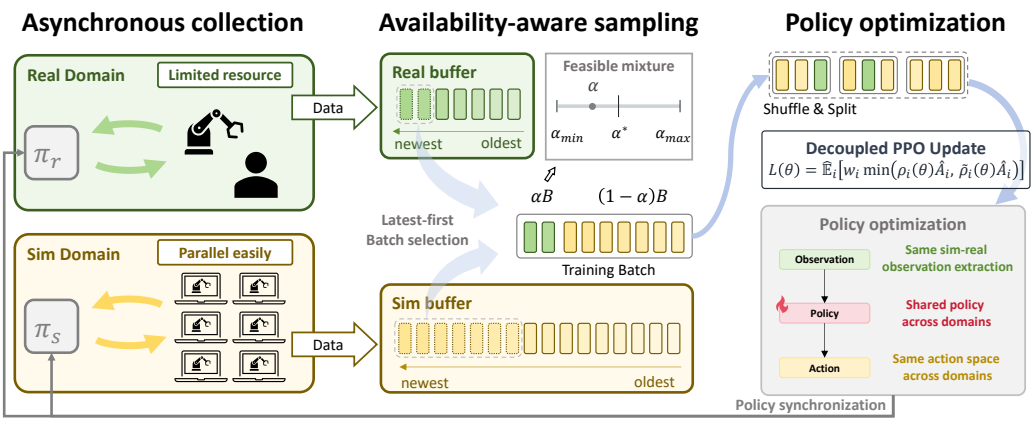

Figure 1: **Overview of ASRC.** Simulated and real-world rollouts are collected separately and combined through availability-aware mixture scheduling and latest-first selection for decoupled PPO updates. Updated policy parameters are synchronized to both collection branches.

4 METHOD

In this section, we present **A**synchronous **S**im-**R**eal **C**o-training (ASRC), a framework for on-policy robot learning with limited real hardware. As illustrated in Figure 1, ASRC overlaps policy optimization with independent data collection from simulated and real-world environments (Section 4.1). Domain-separated buffers feed an availability-aware scheduler that adjusts the sim-real composition of each training batch while prioritizing recent experience (Section 4.2). The resulting batches drive decoupled PPO updates for both direct policy optimization and residual adaptation of pretrained vision-language-action models (Section 4.3).

4.1 DOMAIN-SEPARATED ASYNCHRONOUS COLLECTION

Simulation supports parallel data collection, whereas physical interaction is constrained by the number of available robots and interrupted by resets, object rearrangement, and human feedback. These differences create unequal and time-varying collection rates, so requiring both domains to complete collection together can leave the faster branch idle. ASRC therefore allows simulated and real-world collection to proceed independently and overlap with policy optimization.

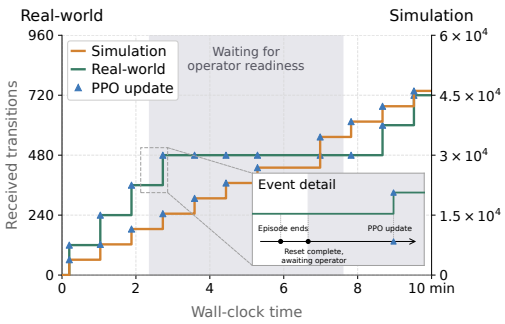

Figure 2: **Independent simulated and real-world interaction.** Cumulative received transitions in a ten-minute co-training trace are shown on separate real and simulation scales. Counts are logged after each PPO update.

Concretely, ASRC maintains a separate environment-interaction loop and a rollout replica of the shared policy π_θ for each domain. The two branches independently generate fixed-length rollout segments and send them to domain-specific buffers $\mathcal{B}_{\text{sim}}$ and $\mathcal{B}_{\text{real}}$. Each segment retains the necessary information for the decoupled PPO update in Equation 3. A shared learner draws on both buffers while collection proceeds, and updated policy parameters are distributed to both rollout branches.

Case study. We select a ten-minute segment from a co-training run and record the cumulative number of transitions received by each domain buffer at every PPO update (Figure 2). Real-world collection pauses for approximately five minutes while awaiting operator readiness. During this pause, simulation continues producing data and the learner continues updating the policy. The inset also illustrates a shorter timing mismatch: after an episode ends, the robot completes its reset

before the next PPO update is logged. A synchronous collect-then-update schedule can require a ready robot to wait for policy optimization before collecting another rollout. By decoupling collection from optimization, ASRC instead allows the next real-world rollout to begin after operator confirmation, without imposing such an update barrier.

4.2 AVAILABILITY-AWARE MIXTURE SCHEDULING

Given independently collected experience in the two domain-specific buffers, we next determine how to assemble a batch for each policy update. This requires deciding both how much experience to draw from each domain and which available segments to select within that domain. To address these two decisions, we combine *availability-aware domain mixing* with *latest-first selection*.

Availability-aware domain mixing. Choosing an appropriate mixing ratio is important for balancing simulated and real-world data (Maddukuri et al., 2025). We therefore prescribe a target real-data fraction $\alpha^\star$ to control the relative contribution of each domain. However, fluctuations in real-world collection can make enforcing this exact ratio incur additional waiting. To accommodate this variability, we introduce an allowed range $[\alpha_{\min}, \alpha_{\max}]$ containing $\alpha^\star$, so that each update can adapt its mixture to available data while remaining within prescribed bounds.

Concretely, each update uses a batch of B equal-length rollout segments, comprising b_{real} real-world segments and $b_{\text{sim}} = B - b_{\text{real}}$ simulated segments. Let n_{real} and n_{sim} denote the available segment counts in the two buffers. We define the feasible real-world counts and select the one closest to the target:

$$\mathcal{F} = \left\{ b \in \mathbb{Z}_{\geq 0} : B - n_{\text{sim}} \leq b \leq n_{\text{real}},\ \alpha_{\min} \leq \tfrac{b}{B} \leq \alpha_{\max} \right\}, \quad b_{\text{real}} \in \arg\min_{b \in \mathcal{F}} \left| \tfrac{b}{B} - \alpha^\star \right|. \quad (4)$$

If $\mathcal{F}$ is empty, the learner waits until enough data arrive to form a feasible batch.

Latest-first selection. Once the domain counts are determined, we select which segments contribute to the update. A natural choice is a first-in, first-out (FIFO) rule, which consumes the earliest arrivals first. However, this can leave the learner training on accumulated experience from earlier behavior policies. We instead select the b_d most recently received segments from each buffer $\mathcal{B}_d$, where $d \in \{\text{sim}, \text{real}\}$. Selected segments are removed from the buffer and are not sampled again for subsequent update batches.

To bound the retention of unused experience, each domain buffer has a capacity C_d. If the buffer exceeds its capacity, the oldest segments are discarded until the capacity is restored. Together with latest-first selection, this rule favors recent experience rather than working through the entire backlog. With availability-aware mixing, each update batch remains close to the prescribed sim-real mixture, subject to data availability, while prioritizing recent experience within each domain. After selection, experience from both domains is combined into a mixed batch for the PPO updates.

4.3 POLICY OPTIMIZATION AND VLA ADAPTATION

We optimize the shared policy on the selected sim-real experience using the decoupled PPO objective in Equation 3. Behavior-policy likelihoods are recorded during collection, while proximal likelihoods are computed under the current learner policy before each update and held fixed throughout it. Samples from both domains thus share the same proximal reference.

We further consider online adaptation of pretrained vision-language-action models. To make such post-training lightweight, we freeze the VLA and train a residual policy to correct its base actions. While the VLA predicts action chunks, the residual policy corrects one action at each control step (Xiao et al., 2026). The residual policy conditions on the current observation o_t and the remaining cached action sequence with its validity mask, denoted by c_t. For the motion dimensions, the residual and the resulting pre-clipping command are

$$\delta a_t \sim \pi_\theta^{\text{res}}(\cdot \mid o_t, c_t), \qquad \tilde{a}_t = a_t^{\text{base}} + \boldsymbol{\lambda} \odot \delta a_t, \quad (5)$$

where a_t^{base} contains the current step's base motion command, $\boldsymbol{\lambda}$ specifies a correction scale for each motion dimension, and $\odot$ denotes element-wise multiplication. The combined command is clipped to the action bounds before execution, with gripper control handled separately when present. We train the residual policy with the same sim-real collection and scheduling pipeline.

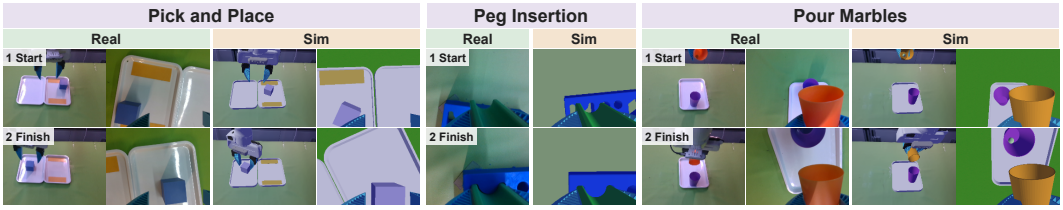

Figure 3: **Real-world and simulated task environments.** Within each domain, *Pick and Place* and *Pour Marbles* show external and wrist views (left to right), while *Peg Insertion* shows the wrist view.

5 EXPERIMENTS

Our experiments investigate (1) whether ASRC improves real-world policy performance with limited online physical interaction, (2) how asynchronous execution translates heterogeneous data collection into training throughput, and (3) how domain composition and sample freshness affect learning. We first evaluate performance for CNN policies and $\pi_{0.5}$ residual adaptation (Section 5.2). To examine system efficiency separately from policy performance, we then measure training throughput under different simulation parallelism levels and mixture ranges, with a synchronous baseline as a reference (Section 5.3). Finally, we vary the real-data mixture and compare latest-first selection with FIFO to assess their effects on learning (Section 5.4).

5.1 EXPERIMENTAL SETUP

Environmental setting. As illustrated in Figure 3, we study three manipulation tasks: *Pick and Place*, *Peg Insertion*, and *Pour Marbles*. Real-world experiments use a single Franka Research 3 robot, with corresponding simulated environments built in ManiSkill (Tao et al., 2024). All tasks use sparse success rewards. Real-world success is checked automatically for Peg Insertion and labeled manually otherwise, without action corrections; simulation uses corresponding task-success checks. More details are provided in Appendix A.

Baselines. We compare ASRC with three groups of methods: (1) *Demonstration-based learning*: real-only BC/SFT and sim-real BC/SFT (Maddukuri et al., 2025). (2) *Simulation-based RL*: zero-shot deployment of a simulation-trained policy, and RL-Co (Shi et al., 2026), which combines simulation RL with supervised learning on real demonstrations. (3) *Real-world RL*: real-only PPO and RLPD (Ball et al., 2023). Inspired by Wu et al. (2025) and Xu et al. (2026b), our RLPD implementation combines demonstrations from both domains with online real-world experience.

Training setup. We instantiate ASRC with ResNet-based CNN policies and residual adaptation of $\pi_{0.5}$ (Intelligence et al., 2025). To initialize task-specific policies, we collect real demonstrations through SpaceMouse teleoperation and generate simulated demonstrations with motion planning. For each task, CNN-based online real-world RL methods share the same BC-initialized actor. Experiments are implemented using RLinf (Yu et al., 2026). More details are provided in Appendix B.

Evaluation protocol. We evaluate policies on the real robot with frozen model parameters and no human action corrections. Each checkpoint is evaluated over 36 trials for *Pick and Place* and *Pour Marbles*, and 40 trials for *Peg Insertion*. Methods are evaluated under the same initial-condition distributions.

5.2 MAIN RESULTS

Real-world task success. We first examine whether ASRC improves policy performance through continued real-world interaction. As shown in Figure 4, both CNN policies and $\pi_{0.5}$ residual policies achieve final success rates above 90% on all three tasks, outperforming the evaluated baselines, with two CNN experiments reaching 100%. In contrast, removing simulated interaction and training with real-only PPO yields smaller gains or even degradation from the initial policy under comparable real-data budgets. RLPD shows particularly sharp initial drops on two tasks. Although it recovers on some tasks, it remains below ASRC within the same real-data budget. We also compare against demonstration-based learning and simulation-based RL. Among these fixed reference policies, zero-shot simulation RL has the lowest success rate in every task-policy setting, highlighting

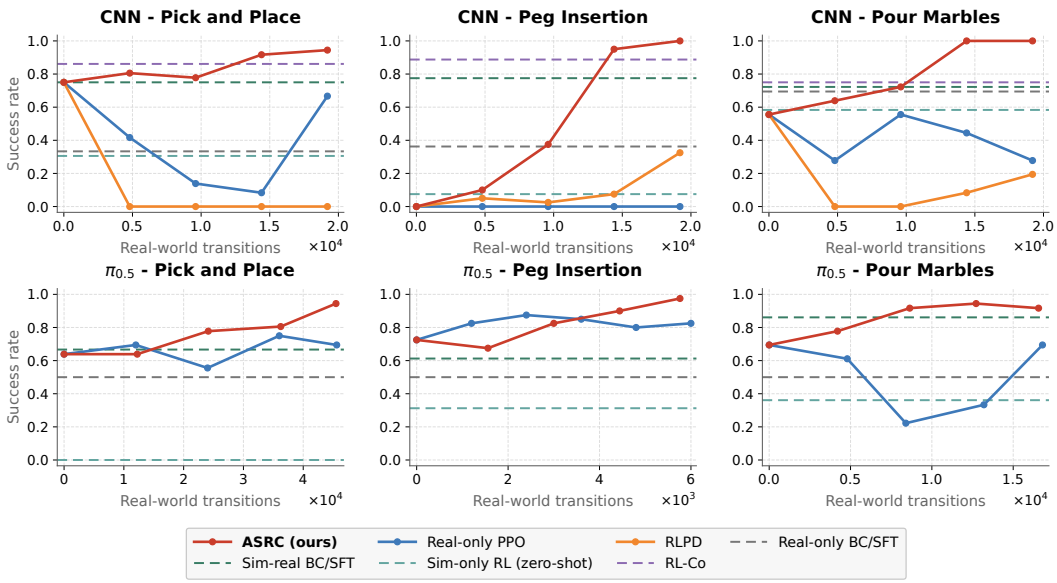

Figure 4: **Real-world task success.** CNN policies (top) and $\pi_{0.5}$ residual adaptation (bottom). Online learning curves show success at evaluated checkpoints against the real-world transition budget. Dashed lines denote fixed reference policies.

the importance of incorporating real-world data. With the same real-demonstration set, adding simulated demonstrations improves BC/SFT performance, while RL-Co achieves further gains through simulation RL. Nevertheless, these methods without continued real-world interaction remain below ASRC, supporting the value of learning from ongoing real-world feedback alongside simulated experience.

Task-execution behavior. Success rates measure whether a policy completes a task but do not fully characterize how it does so. To examine behavioral changes after training, we therefore compare 15 successful executions for Sim-real BC, RL-Co, and ASRC with CNN policies. Figure 5 shows measured end-effector trajectories on *Pour Marbles*, together with image overlays sampled at equal time intervals. Sim-real BC exhibits pronounced path bends and locally slow motion in the illustrated executions. Although RL-Co improves task success through simulation RL while retaining real-demonstration supervision, these patterns remain visible. In contrast, ASRC exhibits more direct approaches and more continuous motion in the illustrated examples.

Table 1: **Successful-trial execution time.** Mean $\pm$ standard deviation in seconds over 15 successful CNN trials per method and task.

Method	Pick and Place	Pour Marbles
Sim-real BC	17.44 ± 2.99	14.11 ± 4.57
RL-Co	14.07 ± 1.98	12.96 ± 3.09
ASRC (ours)	$\mathbf{11.23 \pm 1.29}$	$\mathbf{5.75 \pm 0.22}$

To quantify the execution efficiency associated with these behavioral differences, we report execution times in Table 1. On *Pick and Place*, ASRC reduces mean execution time by approximately 36% relative to Sim-real BC and 20% relative to RL-Co. On *Pour Marbles*, the corresponding reductions are 59% and 56%, consistent with the qualitative observations. ASRC also exhibits a smaller standard deviation in execution time on both tasks. Together, these results show that ASRC achieves not only higher task success but also shorter and more consistent execution times on successful trials.

5.3 System Efficiency

We next study how simulation parallelism, domain composition, and constraints on data staleness affect training throughput. We use CNN policies on *Peg Insertion*, keeping hardware resources and training batch size fixed. We vary the target real-data fraction $\alpha^\star \in \{0.05, 0.1, 0.2\}$ with an allowed deviation of ± 0.05, using 8, 16, or 32 simulation environments. For comparison, we run Sync

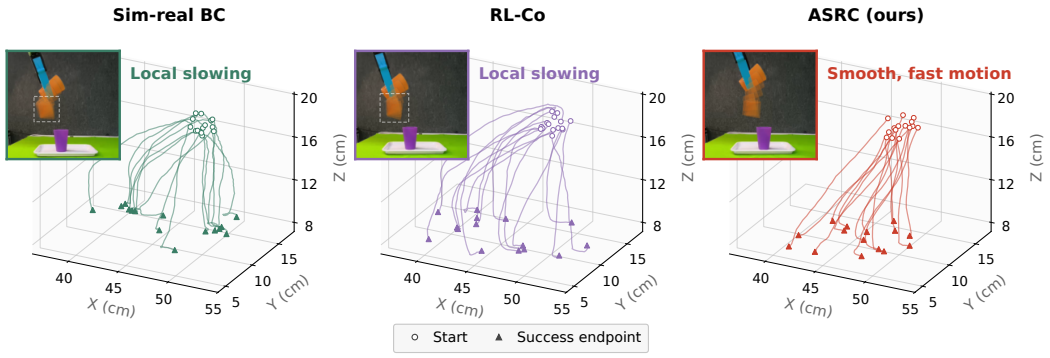

Figure 5: **Execution behavior on Pour Marbles.** Measured end-effector paths from 15 successful CNN-policy trials per method. Circles and triangles mark initial and success-feedback positions. Insets show representative video segments with equal-time image overlays.

Table 2: **Training throughput on Peg Insertion with CNN policies.** Throughput counts training transitions and complete learner rounds. Parentheses report speedup over Sync within each $\alpha^\star$ group. Bold marks the best value and speedup for each metric in each group.

$\alpha^\star$	Mode	# Sim envs	Actual real ratio (%)	Training-data throughput (transitions/s)		Update throughput (rounds/min)
—	—	—	—	Real-world	Simulation	—
0.05	Sync	32	5.56	1.72 (1.00×)	29.32 (1.00×)	0.431 (1.00×)
	Async	8	5.56	1.97 (1.14×)	33.52 (1.14×)	0.493 (1.14×)
	Async	16	5.56	3.22 (1.87×)	54.73 (1.87×)	0.805 (1.87×)
	Async	32	4.76	**4.37** (**2.54**×)	**87.45** (**2.98**×)	**1.275** (**2.96**×)
0.1	Sync	32	8.73	3.23 (1.00×)	33.82 (1.00×)	0.515 (1.00×)
	Async	8	11.11	4.18 (1.29×)	33.45 (0.99×)	0.523 (1.02×)
	Async	16	7.41	**4.55** (**1.41**×)	56.84 (1.68×)	0.853 (1.66×)
	Async	32	5.56	4.38 (1.35×)	**74.45** (**2.20**×)	**1.095** (**2.13**×)
0.2	Sync	32	16.67	3.69 (1.00×)	18.45 (1.00×)	0.308 (1.00×)
	Async	8	16.67	4.36 (1.18×)	21.81 (1.18×)	0.363 (1.18×)
	Async	16	16.67	**4.45** (**1.21**×)	**22.26** (**1.21**×)	**0.371** (**1.21**×)
	Async	32	16.67	4.26 (1.16×)	21.32 (1.16×)	0.355 (1.16×)

with 32 simulation environments at each target fraction. This baseline retains the same collection pipeline but enforces zero staleness, admitting only data collected with the current learner policy version. Table 2 reports real and simulation training-data throughput and update throughput. All metrics use the total wall-clock duration of the measurement window.

Simulation parallelism and domain composition. Increasing the number of simulation environments improves training throughput. For example, at $\alpha^\star = 0.05$, increasing the environment count from 8 to 32 yields roughly 2–3× higher throughput across all three metrics. However, these gains remain constrained by real-world data availability and the allowed mixture range. At $\alpha^\star = 0.1$, additional simulation environments increase throughput while reducing the actual real-data fraction. At $\alpha^\star = 0.2$, increasing the environment count beyond 8 provides little further benefit. This indicates that once simulation supply is sufficient, further parallelism offers limited gains when real-world data availability constrains the mixture. With 16 simulation environments, increasing $\alpha^\star$ from 0.1 to 0.2 leaves real-data throughput nearly unchanged while substantially reducing update throughput. Thus, a higher target fraction can primarily reduce the amount of simulated experience used per unit time rather than accelerate the use of real-world data.

Benefits of asynchronous execution. Beyond resource availability and mixture constraints, requiring all training samples to come from the current policy version also limits data utilization and

update throughput. With the same target fraction and 32 simulation environments, Async achieves higher update throughput than Sync at all three target fractions. This comparison suggests that relaxing the zero-staleness constraint allows independently collected experience to remain usable across policy versions, reducing the data discard and waiting associated with strict version filtering.

5.4 Ablations

Finally, we examine how domain composition and sample freshness affect learning, assessing the availability-aware domain mixing and latest-first selection introduced in Section 4.2. We follow the CNN experimental setup in Section 5.2 on *Pick and Place*, varying only the mixture configuration or sample-selection rule. To compare learning under similar real-data budgets, Figure 6 plots real-world success rates at checkpoints throughout training against the cumulative online real-world transitions used for updates.

Real-data mixture. With an allowed deviation of ± 0.05, we vary the target real-data fraction $\alpha^\star \in \{0.05, 0.1, 0.25, 0.5\}$. As shown in Figure 6 (left), $\alpha^\star = 0.1$ achieves the highest final success rate. Reducing the target to 0.05 leads to less stable performance. This suggests that further reducing the real-data share does not necessarily sustain improvements from co-training. Conversely, larger real-data fractions yield lower final performance under comparable budgets. In particular, $\alpha^\star = 0.5$ reaches only 77.8% after approximately 2×10^4 real-world transitions, a gain of 2.8 percentage points over the initial 75%. Under a fixed batch size, a larger real-data fraction permits fewer updates and less simulated experience for the same real-data budget. These results suggest that an overly large fraction can also limit the real-data efficiency gains from simulation.

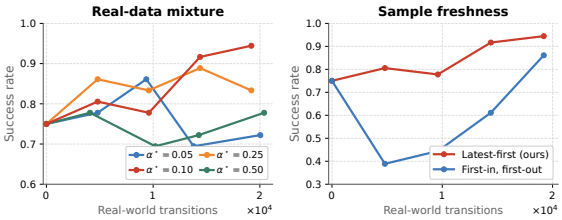

Figure 6: **Effects of domain composition and sample freshness.** Real-world success on Pick and Place with CNN policies, plotted against cumulative online real-world transitions used for updates.

Sample freshness. To assess latest-first selection, we replace it with the first-in, first-out (FIFO) rule described in Section 4.2, which prioritizes the earliest arrivals in each buffer. Figure 6 (right) shows a sharp initial performance drop under FIFO. Although performance subsequently recovers, its final success rate remains below that of latest-first selection. In contrast, latest-first maintains performance above the initial policy at all evaluated checkpoints and shows a smaller intermediate decline. This comparison supports prioritizing recent experience to improve learning stability and final performance in this setting.

6 Conclusions

We presented ASRC, an asynchronous sim-real co-training framework that improves existing robot policies through continued interaction with limited real hardware. Its design coordinates independent collection, domain composition, and experience freshness so that simulated and real-world data can jointly support PPO-style policy improvement. Experiments on three manipulation tasks show higher real-world success for both CNN policies and $\pi_{0.5}$ residual adaptation, shorter and more consistent successful executions, and higher training throughput under asynchronous execution. Together, these results support using ongoing simulated experience alongside real-world feedback to improve deployed policies, extending the role of simulation beyond pretraining or co-training with fixed real-world data.

Limitations. Our experiments start from BC/SFT-initialized policies and require corresponding simulated tasks with aligned observation and control interfaces, leaving the costs of demonstration collection and simulator construction unresolved. Real-world learning also relies on human resets and success labels. Asynchronous execution accommodates these interruptions but does not eliminate operator effort or provide fully autonomous training.

AI USE STATEMENT

We used generative AI tools to assist with writing and research execution. For writing, these tools helped refine author-written drafts. For research execution, they assisted with method implementation and the preparation of experimental and plotting scripts, following detailed plans and specifications provided by the authors. The authors reviewed and revised the AI-assisted text and validated the resulting code and experimental outputs. The authors assume full responsibility for the accuracy and integrity of the manuscript, implementation, and reported results.

REPRODUCIBILITY STATEMENT

We provide details to facilitate reproduction of our results. Appendix A describes the real-world and simulation setups, task definitions, observation and action interfaces, and success criteria. Appendix B details demonstration collection, policy architectures, initialization, and the implementations and training configurations of ASRC and the baselines.

REFERENCES

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, et al. π_{RL}: Online RL fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

Shuo Cheng, Liqian Ma, Zhenyang Chen, Ajay Mandlekar, Caelan Garrett, and Danfei Xu. Generalizable domain adaptation for sim-and-real policy co-training. *Advances in Neural Information Processing Systems*, 38:11905–11933, 2026.

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*, 2024. doi: 10.15607/RSS.2024.XX.045.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Shirli Di-Castro, Dotan Di Castro, and Shie Mannor. Sim and real: Better together. *Advances in Neural Information Processing Systems*, 34:6868–6880, 2021.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models, 2026. URL `https://arxiv.org/abs/2605.25477`.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *Advances in Neural Information Processing Systems*, 38:36256–36282, 2026.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. Pmlr, 2018.

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

Xiaoshen Han, Junqiu Yu, Minghuan Liu, Yilun Chen, Xiaoyang Lyu, Yang Tian, Bolun Wang, Weinan Zhang, and Jiangmiao Pang. Re ^{3} sim: Generating high-fidelity simulation data via 3d-photorealistic real-to-sim for robotic manipulation. *arXiv preprint arXiv:2502.08645*, 2025.

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. *Advances in Neural Information Processing Systems*, 35:17086–17098, 2022.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi05: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. Performant robotic manipulation with real-world reinforcement learning. *Science Robotics*, 11(116):eaed6267, 2026a. doi: 10.1126/scirobotics.aed6267. URL `https://www.science.org/doi/abs/10.1126/scirobotics.aed6267`.

Yu Lei, Minghuan Liu, Abhiram Maddukuri, Zhenyu Jiang, and Yuke Zhu. A mechanistic analysis of sim-and-real co-training in generative robot policies. *arXiv preprint arXiv:2604.13645*, 2026b.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *Advances in Neural Information Processing Systems*, 38:97121–97151, 2026.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Wenxing Liu, Hanlin Niu, Wei Pan, Guido Herrmann, and Joaquin Carrasco. Sim-and-real reinforcement learning for manipulation: A consensus-based approach. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3911–3917. IEEE, 2023.

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

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.

Siyu Ma, Yuqi Liang, Chang Yu, Yunuo Chen, Hao Su, Yixin Zhu, Yin Yang, and Chenfanfu Jiang. Taccorl: Integrating tactile feedback into vla via simulation. *arXiv preprint arXiv:2606.11743*, 2026.

Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, et al. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. *arXiv preprint arXiv:2503.24361*, 2025.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Haoyi Niu, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming Hu, Xianyuan Zhan, et al. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. *Advances in Neural Information Processing Systems*, 35:36599–36612, 2022.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Liangzhi Shi, Shuaihang Chen, Feng Gao, Yinuo Chen, Kang Chen, Tonghe Zhang, Hongzhi Zang, Jiakai Zhou, Weinan Zhang, Chao Yu, et al. Beyond imitation: Reinforcement learning-based sim-real co-training for vla models. *arXiv preprint arXiv:2602.12628*, 2026.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*, 2024.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. In *International Conference on Machine Learning*, pp. 34556–34583. PMLR, 2023.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

Adam Wei, Abhinav Agarwal, Boyuan Chen, Rohan Bosworth, Nicholas Pfaff, and Russ Tedrake. Empirical analysis of sim-and-real cotraining of diffusion policies for planar pushing from pixels. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5625–5632. IEEE, 2025.

Mingdong Wu, Lehong Wu, Yizhuo Wu, Weiyao Huang, Hongwei Fan, Zheyuan Hu, Haoran Geng, Jinzhou Li, Jiahe Ying, Long Yang, et al. Simlauncher: Launching sample-efficient real-world robotic reinforcement learning via simulation pre-training. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7933–7940. IEEE, 2025.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Jim Fan, Guanya Shi, et al. Self-improving vision-language-action models with data generation via residual rl. In *International Conference on Learning Representations*, volume 2026, pp. 13209–13236, 2026.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. Rl token: Bootstrapping online rl with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026a.

Qinwen Xu, Jiaming Liu, Rui Zhou, Shaojun Shi, Nuowei Han, Zhuoyang Liu, Chenyang Gu, Shuo Gu, Yang Yue, Gao Huang, et al. Twinrl: Digital twin-driven reinforcement learning for real-world robotic manipulation. *arXiv preprint arXiv:2602.09023*, 2026b.

Patrick Yin, Tyler Westenbroek, Ching-An Cheng, Andrey Kolobov, and Abhishek Gupta. Rapidly adapting policies to the real-world via simulation-guided fine-tuning. In *International Conference on Learning Representations*, volume 2025, pp. 92257–92278, 2025.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pp. 829–846, 2026.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Zhang Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations*, volume 2025, pp. 28129–28164, 2025.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Peihong Wang, Hua Yuan, Yixian Zhang, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. RLux-VLA: A Unified and Efficient Framework for Reinforcement Learning of Vision-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026a. doi: 10.15607/RSS.2026.XXII.089.

Hongzhi Zang, Shu'ang Yu, Hao Lin, Tianxing Zhou, Zefang Huang, Zhen Guo, Xin Xu, Yuze Sheng, Jiakai Zhou, Shizhe Zhang, Feng Gao, Wenhao Tang, Yufeng Yue, Quanlu Zhang, Xinlei Chen, Chao Yu, and Yu Wang. USER: A Unified and Extensible System for Online Real-World Policy Learning in Embodied AI. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026b. doi: 10.15607/RSS.2026.XXII.037.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 106282–106319, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

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

A Environment settings

A.1 Real-world setups

Figure 7 shows our real-world setup, consisting of a tabletop workspace, a Franka Research 3 robot equipped with a UMI gripper (Chi et al., 2024), and two RGB cameras, one fixed externally and one mounted on the wrist. We use only the cameras' RGB channels as visual input, without depth. The control interface is seven-dimensional, comprising three end-effector translation increments, three rotation increments, and one gripper command. The environment accumulates the pose increments onto the previous target end-effector pose and sends the resulting absolute pose to the robot controller. Gripper actuation is handled separately.

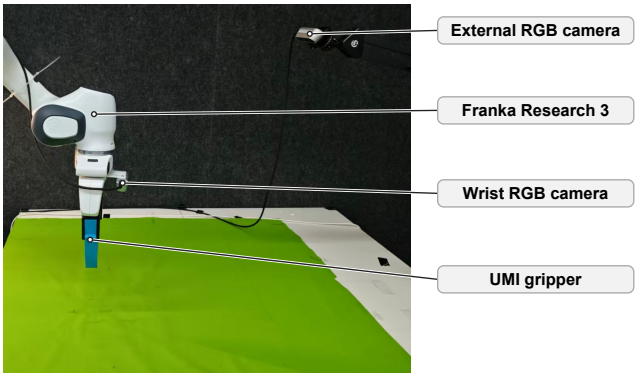

Figure 7: **Real-world setup.** The robot, UMI gripper, and external and wrist-mounted RGB cameras above the tabletop workspace.

A.2 Simulation setups

Simulation platform. We build our environments in ManiSkill (Tao et al., 2024), using SAPIEN-based rigid-body simulation. We first measure the real table's length, width, and height, together with the robot base position and orientation relative to the table. We then model a table with matching dimensions and mount the simulated robot at the corresponding pose. The robot is based on a Franka Panda URDF with its gripper replaced by a UMI model. This construction aligns the simulated and real-world setups at the task-layout level.

Visual alignment. We use a simple visual alignment procedure rather than a complex reconstruction pipeline. We first measure the approximate camera locations in the real workspace for coarse calibration. We then record real camera images at five distinct robot joint configurations and reproduce the same configurations in simulation. We refine calibration by manually adjusting the simulated camera poses to approximately align the rendered robot with the real images across all five configurations. For the external-camera background, we capture a photograph of the real workspace and composite the segmented simulated robot and task objects onto this image. This provides coarse background alignment without reproducing subtle state-dependent changes in background appearance. The wrist-camera view uses the simulated rendering directly. Objects are represented by meshes that approximately match their shapes and colors, without reconstruction to reproduce their exact appearance.

Physical setup. Precisely matching simulated and real-world control dynamics would require extensive system identification. We do not perform such identification, instead retaining ManiSkill's default Panda joint stiffness and damping, set to 1000 and 100, respectively. Physics and control frequencies are 500 Hz and 10 Hz, respectively. As shown in Section 5.2, ASRC benefits from simulated interaction even with this simplified physical setup and uncalibrated controller gains.

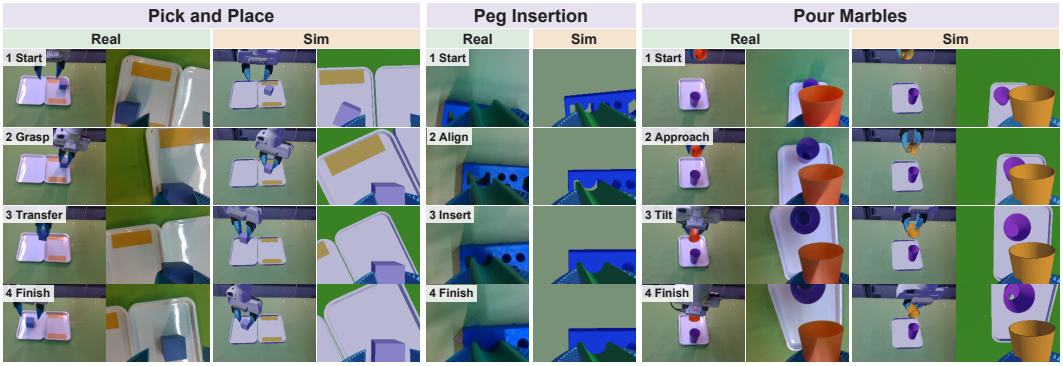

Figure 8: Real-world and simulated task environments at four successive demonstration stages (top to bottom). Within each domain, Pick and Place and Pour Marbles show external and wrist views (left to right), while Peg Insertion shows the wrist view.

A.3 Task definitions

Figure 8 illustrates the three tasks from their initial states to completion. Each episode is limited to 120 environment steps. We describe each task's goal, initialization, and failure condition below.

Pick and Place. The goal of this task is to grasp a cube from a source tray and release it inside a target tray. Both trays occupy fixed positions on the table. The cube center is sampled uniformly within ± 85 mm and ± 52.5 mm along the two horizontal axes relative to the source tray center, with yaw sampled from $[-\pi, \pi]$. The robot starts with an open gripper, and its initial end-effector pose is perturbed from a reference pose by up to 15 mm along each horizontal axis, 10 mm vertically, and 0.35 rad in yaw. An episode fails if the cube is not released inside the target tray within the 120-step limit.

Peg Insertion. The goal of this task is to insert a U-shaped peg attached to the robot's end effector into a matching hole in a board, without a preceding grasping stage. The initial robot pose is defined relative to a calibrated fully inserted target pose. The peg starts 100 mm above this target along the insertion axis, with independent uniform offsets of up to 50 mm along each axis of the insertion plane and a rotation of up to 10° around the insertion axis. An episode fails if insertion is not completed within the 120-step limit.

Pour Marbles. The goal of this task is to pour a single marble from a cup attached to the robot's end effector into a receiving cup placed in a tray. The receiving cup is initially upright, with its center sampled uniformly within ± 80 mm and ± 47.5 mm along the tray's horizontal axes. The pouring cup initially contains one marble. The robot's initial end-effector pose is perturbed from a reference pose by up to 15 mm along each horizontal axis, 10 mm vertically, and 0.1 rad in yaw. An episode fails if the marble is not transferred into the receiving cup within the 120-step limit.

A.4 Observation

Visual input. Both real-world and simulated cameras produce RGB images at 640×480 pixels. We add equal black padding above and below each image to obtain a 640×640 square, then resize it to 224×224. *Pick and Place* and *Pour Marbles* use both external and wrist-camera views, whereas *Peg Insertion* uses only the wrist-camera view. To reduce training cost, CNN models further resize the images to 128×128 before feeding them into the network.

State input. In addition to images, each task provides a state input. For *Pick and Place* and *Pour Marbles*, this is a 14-dimensional vector comprising seven joint angles, the tool center point (TCP) position and three Euler angles, and a gripper-state scalar. For *Peg Insertion*, we first measure the TCP pose at full insertion in the real-world setup and use it as the target pose T_{target}. We compute the relative pose $T_{\text{rel}} = T_{\text{target}}^{-1} T_{\text{TCP}}$ and use its translation and rotation-vector components as the

six-dimensional state. These state inputs are available during interaction in both real-world and simulated environments.

A.5 ACTION

As described in Appendix A.1, we use delta end-effector control. For safety, we impose task-specific workspace and orientation bounds. If the accumulated target end-effector pose exceeds these bounds, it is clipped to the allowed range. The same bounds apply in simulation and the real world. We also scale model actions to task-specific ranges. Per-step translation increments are limited to 10 mm per axis for *Pick and Place* and *Pour Marbles*, and 2 mm for *Peg Insertion*, which requires finer motion. Rotation increments are limited to 0.05 rad per component for *Pick and Place* and *Pour Marbles*, and 0.02 rad for *Peg Insertion*.

Inference schedules differ across models. CNN policies predict one action at a time and perform the next inference after receiving a new observation from the environment. In contrast, $\pi_{0.5}$ predicts action chunks of length 10 for *Pick and Place* and *Pour Marbles*, and length 8 for *Peg Insertion*. During residual adaptation, $\pi_{0.5}$ generates a complete action chunk at once, and the CNN residual policy corrects each action sequentially using the latest observation at every control step.

A.6 SUCCESS CRITERIA AND REWARD

Success criteria. Success is defined as completing the task described in Appendix A.3 within 120 environment steps. In simulation, we evaluate success using the full environment state, enabling automated checks across many parallel environments. Real-world *Peg Insertion* also uses an automated check. We first record the TCP pose at successful insertion and mark an episode as successful when the current TCP pose falls within a specified tolerance of this target. For *Pick and Place* and *Pour Marbles*, an operator observes whether the task has been completed as specified and provides success feedback through the keyboard. The environment marks the episode as successful upon receiving this feedback. Despite the safety bounds, real-world execution can still involve unsafe behavior, such as collisions or damage to task objects. We therefore allow the operator to signal such behavior through the keyboard, which marks the episode as failed and terminates it early. Simulated episodes instead continue until success or the step limit.

Reward. All tasks use sparse rewards. A successful transition receives an environment reward of 1.0 and terminates the episode, while all other transitions receive 0. Early termination due to unsafe real-world behavior incurs no additional penalty and also receives a reward of 0.

B IMPLEMENTATION DETAILS

B.1 DATA COLLECTION

We collect real-world demonstrations through SpaceMouse teleoperation, with initial object placements approximately matching the randomization ranges defined for each environment. In simulation, we generate demonstrations through motion planning by manually decomposing each task into key actions and interpolating between them to form a complete trajectory. Table 3 reports the total number of demonstrations collected for the three tasks.

Table 3: **Demonstration dataset sizes.** Number of demonstration trajectories collected for each task and domain.

Domain	Pick and Place	Peg Insertion	Pour Marbles
Simulation	500	500	500
Real-world	250	150	100

B.2 POLICY ARCHITECTURE

We use ResNet-10 to encode each 128×128 image into a 256-dimensional feature and project the state input into a 64-dimensional feature. For residual policies, we encode the remaining cached actions and their validity mask into a 64-dimensional feature. These features are concatenated and passed through the actor network to predict an action or residual correction. A value network takes the same combined features and outputs a scalar state value. For RLPD, the critic instead concatenates the observation features with the action and processes them through a network to estimate the Q-value.

B.3 REAL-WORLD RL INITIALIZATION

Before real-world RL training, we prepare a capable policy to provide a shared initialization for ASRC, real-only PPO, and RLPD. For CNN policies on *Pick and Place*, we perform BC using 450 simulated and 225 real-world demonstration trajectories. The remaining 50 simulated and 25 real-world trajectories form a held-out validation set used to select the best checkpoint. For *Peg Insertion* and *Pour Marbles*, we use only simulated demonstrations for BC to examine learning with less reliance on real-world data. Each task uses 450 simulated trajectories for training and the remaining 50 for validation-based checkpoint selection.

We follow the same domain-selection and validation strategy when fine-tuning $\pi_{0.5}$. For *Pick and Place*, we use demonstrations from both domains with held-out validation data. For *Peg Insertion* and *Pour Marbles*, both training and validation use only simulated demonstrations. Unlike the standalone $\pi_{0.5}$ baselines, the base policies used to initialize real-world RL receive no state input. They predict action chunks from external and wrist-camera images for *Pick and Place* and *Pour Marbles*, and from wrist-camera images alone for *Peg Insertion*.

B.4 IMPLEMENTATIONS

Demonstration-based baselines. We train CNN policies with BC and $\pi_{0.5}$ with SFT on the collected demonstrations. Real-only baselines use only real-world data, whereas sim-real baselines use both domains. CNN baselines retain the same inputs and network architecture as ASRC. For $\pi_{0.5}$ baselines, we add state inputs to match the information available to ASRC's residual policy.

Simulation-only RL. For CNN policies, we initialize with simulation-only BC and then train with PPO in simulation. For $\pi_{0.5}$, we initialize with simulation-only SFT and use the same residual architecture as ASRC, optimizing the residual policy with PPO while keeping the base model frozen. Both policy families receive extended simulation training before real-world evaluation.

RL-Co. Following the two-stage procedure of RL-Co (Shi et al., 2026), we use the same initial model as ASRC from the first stage and optimize simulation PPO with an auxiliary supervised loss on real training episodes. The auxiliary-loss weight is 0.2. The second stage uses the real-world demonstrations collected as described in Appendix B.1 for supervision.

RLPD. Following the RLinf-USER implementation (Zang et al., 2026b), we use an off-policy actor–critic with an ensemble of ten Q heads and two randomly sampled heads for target estimation. Across all three tasks, the actor starts from the same BC policy as ASRC. We additionally preload a prior-data buffer with real-world and simulated trajectories, and sample each training batch equally from this buffer and the online replay buffer. Table 4 lists the training settings and prior-data counts.

Real-only PPO. We use the same task-specific initialization and a configuration similar to ASRC, but remove simulation collection. To allow more frequent updates, we keep the optimizer batch size unchanged and reduce the collection interval from 4,320 to 240 transitions.

ASRC. We collect simulated and real-world experience asynchronously and use availability-aware domain mixing with latest-first selection to construct batches for decoupled PPO updates. The same framework supports CNN policies and $\pi_{0.5}$ residual adaptation. Table 5 summarizes the task-specific training configurations.

Table 4: **RLPD training configurations.** The learning rate applies to the actor, critics, and entropy temperature. Prior-data counts refer to trajectories loaded before online training.

Parameter	Pick and Place	Peg Insertion	Pour Marbles
Discount factor γ	0.98	0.98	0.98
Target update rate τ	0.005	0.005	0.005
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}
Batch size	256	256	256
Initial temperature α	0.01	0.002	0.002
Initial log standard deviation	-0.5	0	0
Target entropy	-3	-3	-3
Real-world prior trajectories	20	23	20
Simulation prior trajectories	50	50	50

Table 5: **ASRC training configurations.** Task-specific settings for CNN policies and $\pi_{0.5}$ residual adaptation.

Model	Parameter	Pick and Place	Peg Insertion	Pour Marbles
CNN	Actor learning rate	10^{-5}	10^{-5}	10^{-5}
	Value learning rate	10^{-4}	10^{-4}	10^{-4}
	Optimization epochs	2	2	2
	Entropy coefficient	0.002	0.002	0
	Simulation environments	32	32	32
$\pi_{0.5}$ residual	Actor learning rate	2×10^{-5}	2×10^{-5}	10^{-4}
	Value learning rate	2×10^{-4}	2×10^{-4}	2×10^{-4}
	Optimization epochs	4	4	4
	Entropy coefficient	0	0	0
	Simulation environments	16	32	16

B.5 SIMULATION CURVE

Figure 9 reports simulation success rates during the same ASRC co-training runs used for the real-world results in Figure 4. We plot the raw per-iteration values without smoothing. Across all three tasks and both policy families, simulation success improves alongside real-world performance during co-training.

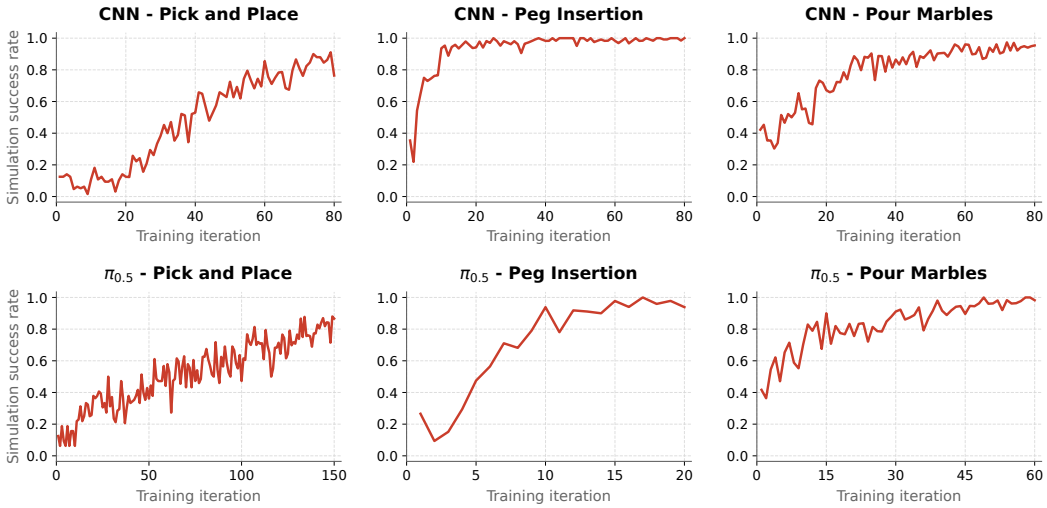

Figure 9: **Simulation success during ASRC co-training.** Raw simulation success rates for CNN policies (top) and $\pi_{0.5}$ residual adaptation (bottom), recorded during online sim-real training.