EMBODIRSI: RECURSIVE SELF-IMPROVEMENT FOR DATA-EFFICIENT ROBOT ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting robot manipulation policies to new tasks and environments remains highly data-intensive, while the data needed for further improvement depends on the policy's current capabilities and failure modes. We introduce EmbodiRSI, an agentic system for recursive self-improvement (RSI) in a real-to-sim-to-real setting, where task-specific simulations are constructed from target deployment scenarios and used as low-cost environments for iterative policy improvement before transfer back to the physical world. EmbodiRSI uses policy execution feedback to guide subsequent experience acquisition and policy updates. Two complementary mechanisms close this loop: *Collaborative Error Correction* generates agent-assisted corrective trajectories from policy-reached states, while *Adaptive Data Collection* directs expert demonstration generation toward the current policy's weaknesses. The task-specific simulation serves as a reusable workspace for policy warm-up, repeatable evaluation, failure diagnosis, and targeted data generation across successive RSI rounds. Across three tabletop environments and 14 subtasks, EmbodiRSI increases scene-balanced autonomous simulation success from 50.4% to 83.5% over two RSI updates. With 400 adaptive simulated trajectories and only ten real-world refinement trajectories per subtask, EmbodiRSI achieves 83.1% scene-balanced autonomous real-world success, compared with 75.0% for adaptation using 200 real-world demonstrations per subtask. These results demonstrate that feedback-driven recursive improvement in deployment-specific simulations can enable data-efficient adaptation of embodied policies to physical environments.

1 INTRODUCTION

Recent advances in general-purpose robot manipulation policies have substantially expanded the range of tasks that a single robot can perform (Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025b), yet reliable deployment in new environments remains challenging. Deployment often reveals systematic failure modes that are insufficiently covered by the original training distribution. Importantly, these failure modes change as the policy improves: some weaknesses are resolved, while the remaining errors become concentrated in different states or configurations. Effective adaptation therefore requires more than a fixed training set. It calls for a learning system that repeatedly evaluates the current policy, identifies its remaining weaknesses, and acquires targeted experience for subsequent improvement, without relying on costly physical data collection and human intervention at every iteration.

The missing link is between *observing a failure* and *obtaining useful experience for subsequent improvement*. Within an episode, a missed grasp or displaced object can leave the robot in a state absent from nominal demonstrations. Across episodes, recurring failures reveal regions of the task distribution where additional experience is most needed. Dataset aggregation supervises learner-induced states (Ross et al., 2011); intervention and recovery methods turn execution errors into training experience (Hoque et al., 2024; Lin et al., 2025); and recent agentic systems coordinate collection, learning and execution (Li et al., 2026). These directions motivate a broader question:

Can a robot use feedback from its current policy to determine both how to recover from failures and what experience should be acquired for the next policy update?

We introduce **EmbodiRSI**, an agentic learning system for *recursive self-improvement* (RSI) within a *real-to-sim-to-real* (R2S2R) workflow (Figure 1). Starting from a target physical workspace,

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.

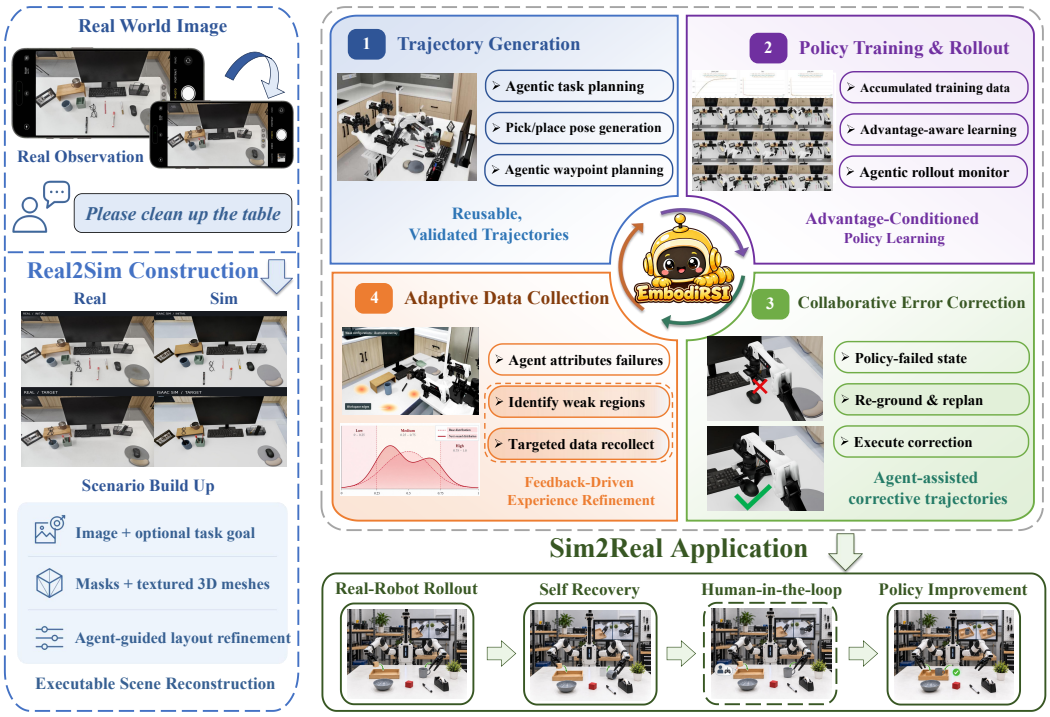


Figure 1: **Overview of EmbodiRSI.** Real-world observations and task goals guide the construction of an executable simulation environment and the generation of validated demonstrations. Within this environment, policy rollout feedback guides Collaborative Error Correction and Adaptive Data Collection. The resulting experience updates the policy for the next round, forming a recursive self-improvement loop. A small real-world human-in-the-loop (HIL) dataset then refines the policy for autonomous deployment.

EmbodiRSI constructs a task-specific simulation in which the policy can be repeatedly evaluated and improved before transferring back to the real world. The current policy generates rollouts, execution outcomes determine what experience should be acquired next, and the resulting experience is used to update the policy. The updated policy is then evaluated again, producing new feedback for the next round. In this way, the experience distribution evolves together with the policy rather than being fixed in advance.

Two complementary mechanisms close this RSI loop. *Collaborative Error Correction* operates within an episode: when policy execution fails, an agent guides recovery from the intermediate state actually reached by the policy, generating corrective trajectories without resetting the task. *Adaptive Data Collection* operates across episodes: the agent aggregates rollout outcomes, identifies recurring failure modes, and redirects subsequent expert demonstration generation toward configurations where the current policy remains weak. Together, these mechanisms transform execution feedback into targeted training experience for subsequent policy updates. The updated policy is then rolled out again to reveal its remaining limitations.

Task-specific simulation provides the workspace in which this recursive improvement can be performed efficiently and repeatedly. Given observations of the target deployment environment and task specifications, EmbodiRSI reconstructs and organizes a corresponding virtual workspace, while a trajectory agent generates validated expert demonstrations for policy initialization and subsequent adaptive collection. The same environment supports repeatable evaluation, failure diagnosis, error correction, and targeted data generation. Geometric reasoning, collision checking, and motion-planning tools are used to validate generated trajectories and recovery actions. After iterative improvement in simulation, only a small number of human-in-the-loop (HIL) trajectories are collected in the physical workspace to refine the policy for autonomous real-world execution.

Our evaluation follows this R2S2R progression from recursive improvement in task-specific simulation to deployment in the corresponding physical environments. Across three tabletop environments and 14 subtasks, EmbodiRSI increases scene-balanced autonomous simulation success from 50.4% to 83.5% over two RSI updates (R0 to R2). To isolate the benefit of feedback-driven experience acquisition, we further compare adaptive and static collection in two environments at matched retained-trajectory counts. With a cumulative budget of 400 simulated trajectories per subtask, adaptive collection improves scene-balanced simulation success by 20.6 percentage points and pre-refinement real-world success by 12.9 points over static collection. In the full three-environment evaluation, EmbodiRSI achieves 83.1% scene-balanced autonomous real-world success using 400 adaptive simulated trajectories and only ten real-world refinement trajectories per subtask, compared with 75.0% for adaptation using 200 real-world demonstrations per subtask.

Our main contributions are as follows.

- We propose EmbodiRSI, an agentic recursive self-improvement system within an R2S2R workflow, where policy execution feedback iteratively guides experience acquisition and policy updates in task-specific simulation environments.
- We develop two complementary mechanisms to close the RSI loop: *Adaptive Data Collection*, which directs expert demonstrations toward the current policy's weaknesses, and *Collaborative Error Correction*, which generates corrective trajectories from policy-reached failure states through agent-guided recovery.
- We demonstrate progressive policy improvement and data-efficient real-world adaptation across diverse manipulation tasks. EmbodiRSI achieves higher real-world success with only a small amount of physical refinement data than adaptation using substantially more real-world demonstrations.

2 Related Work

Simulation-based Scalable Data Generation. RoboGen and GenSim generate environments or tasks (Wang et al., 2023; 2024); MimicGen, SkillMimicGen, DexMimicGen and DemoGen expand source demonstrations (Mandlekar et al., 2023; Garrett et al., 2024; Jiang et al., 2025b; Xue et al., 2025). RoboTwin uses generative digital twins (Mu et al., 2025) and RoboTwin 2.0 scales bimanual data generation with domain randomization (Chen et al., 2025). SAGE and V-CAGE incorporate agentic generation and verification (Xia et al., 2026; Liu et al., 2026). Deployment-specific Real2Sim2Real systems connect reconstructed workspaces to policy learning and transfer (Torne et al., 2024; Fang et al., 2025; Han et al., 2025; Dan et al., 2025; Patel et al., 2025). EmbodiRSI uses these capabilities as infrastructure for repeated policy improvement: the workspace remains available after initial training, while rollout feedback revises what experience is generated within it.

Robot Learning via Real2Sim2Real. Real2Sim2Real reduces real-world data requirements by reconstructing or generating simulation from real observations, learning policies or expanding robot experience in simulation, and transferring the resulting policies or behaviors back to real robots. Representative systems include RialTo (Torne et al., 2024), RL-GSBridge (Wu et al., 2025), RoboGSim (Li et al., 2024), Re^{3}Sim (Han et al., 2025), RoboSimGS (Zhao et al., 2026), ReBot (Fang et al., 2025), and X-SIM (Dan et al., 2025). Closely related real-to-sim and photorealistic digital-twin systems, such as IKER (Patel et al., 2025), Scalable Real2Sim (Pfaff et al., 2025), RoboPearls (Tao et al., 2025), GSWorld (Jiang et al., 2025a), and Real2Sim-Eval (Zhang et al., 2025), demonstrate the value of simulation as a bridge between real observations and robot execution. EmbodiRSI further extends this paradigm with an agent-driven closed-loop that unifies scene construction, trajectory generation, policy training, rollout evaluation, failure recovery, and data refinement.

Recursive Self-Improvement for Robot Learning. Recent work on recursive robot self-improvement explores how policies can be iteratively refined using experience generated by their own evolving behavior. SOAR (Zhou et al., 2024) uses foundation models to guide autonomous data collection and learning from unlabeled experience, while RECAP (Intelligence et al., 2025a) combines demonstrations, autonomous experience, and corrective feedback through advantage conditioning. VERITAS (Zhang & Shah, 2026) uses visual verification to filter autonomous rollouts for subsequent policy improvement, and RoboClaw (Li et al., 2026) integrates autonomous data collection, learning,

and execution into a unified iterative loop. RISE (Yang et al., 2026) further improves policies through imagined rollouts and advantage estimation in a learned world model. EmbodiRSI follows this direction by coupling within-episode recovery with across-episode recollection, so that experience generated by each policy iteration informs subsequent updates. Appendix F provides further context.

3 METHOD: A SYSTEM FOR RECURSIVE POLICY SELF-IMPROVEMENT

EmbodiRSI couples policy optimization to the acquisition of its next training experience. At round k, let π_k be the current scene-specific policy, $\mathcal{D}_k$ the accumulated simulation dataset and $p_k(c)$ the distribution of demonstration-collection configurations. A configuration c specifies a source object, target anchor, instruction, grasp and waypoint program, and admissible reset ranges. One policy covers all subtasks in a scene. Sections 3.1 and 3.2 describe the supporting infrastructure; Section 3.3 develops the RSI loop.

3.1 AN EXECUTABLE ENVIRONMENT FOR RSI

Given a tabletop image and, optionally, a target image or language goal, EmbodiRSI constructs an initial scene $\mathcal{S}_0$ and an organized target scene $\mathcal{S}_{\text{org}}$ through a fully automated, agent-orchestrated pipeline. The scene agent identifies objects and invokes SAM 3 (Carion et al., 2026) to obtain instance masks and SAM 3D (Chen et al., 2026) to reconstruct the corresponding textured meshes. It then applies Agent-Guided Adaptive Collision Rectification to resolve interpenetration while preserving support and containment relations. To construct the organized target scene, the agent establishes coarse task relations and refines object poses using rendered views and geometric checks.

This scene is the *practice environment* for RSI: it supports repeatable resets, exposes object states for training-time diagnosis and permits recovery from policy-failed states. It is retained throughout learning, rather than used only to produce a seed dataset. Scene construction, geometric operators and organization examples are detailed in Appendices B and C.

3.2 A REUSABLE SOURCE OF TRAINING EXPERIENCE

A trajectory agent compiles the scene pair into executable task configurations and validated demonstrations. It proposes object-space routes, projects them through GraspGen (Murali et al., 2025) candidates, and validates complete robot programs with cuRobo (Sundaralingam et al., 2026). Admission requires swept-object collision checking, multi-view review, sequential motion feasibility and execution-time task checks. Infeasible candidates are repaired and revalidated or rejected.

Stored waypoints follow source or target anchors. After a reset, they are re-grounded to measured object poses and replanned from the current robot state. This reusable interface connects agent decisions to executable supervision for initial collection, recovery and targeted recollection. Appendix B.2 details compilation and execution.

3.3 AGENT-GUIDED RECURSIVE POLICY SELF-IMPROVEMENT

Initial demonstrations may omit both imperfect learner-induced states and configurations that remain difficult after training. EmbodiRSI addresses these gaps at two timescales (Figure 2): recovery supplies corrective supervision *within* an episode, while rollout diagnosis refines the collection distribution *across* episodes. Accumulated experience updates the policy, closing the loop summarized in Algorithm 1.

3.3.1 AGENTIC ROLLOUT MONITOR

During a rollout $\tau^{\pi_k} = \{(o_t, x_t, a_t)\}_{t=1}^{T}$, a monitor agent combines a temporal image window, action history and simulator object/robot state x_t:

$$y_t = \mathcal{A}_{\text{RSI}}(o_{t-w:t}, x_t, a_{1:t}, l), \qquad y_t \in \{\texttt{continue}, \texttt{recover}, \texttt{complete}\}. \tag{1}$$

Visual evidence identifies missed grasps, slippage and incorrect placement; programmatic checks measure target-object error and gripper state. Completion requires both a visual decision and geometric success. Recovery is gated by confidence, cooldown, attempt budget and an object-holding

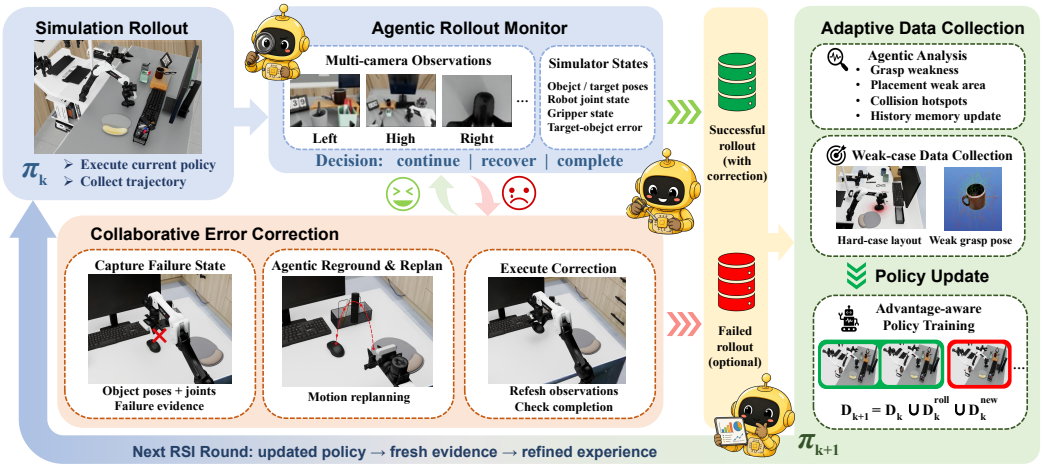


Figure 2: **The recursive self-improvement loop in EmbodiRSI.** The rollout monitor combines multi-camera observations and simulator states to trigger recovery from the current state without resetting. Failure analysis guides targeted data collection. Successful rollouts, including those completed after correction, and newly generated demonstrations are accumulated to update the policy for the next round. External recovery is used only during training; evaluation is autonomous.

test, avoiding interruption of an ongoing valid transport. Decisions are logged with configuration identifiers, scene states and failure evidence. Simulator object poses support these training tools but are not inputs to the learned policy.

3.3.2 Collaborative Error Correction

Recovery must solve the task from the state the policy actually reached, not replay a clean demonstration from its initial state. At a recovery trigger t_f, a nominal grasp is transported to the current object pose:

$$T_g^{\text{cur}} = T_o^{\text{cur}} (T_o^0)^{-1} T_g^0. \tag{2}$$

Here, T_o^{cur} and T_o^0 denote the current and initial poses of object o, respectively. Candidate configurations are probed in deterministic order without advancing the simulator. Entry motion is planned from the *current joint state*, and targets are re-grounded to the current scene. The correction can provide an entry-to-grasp motion or continue the remaining manipulation program.

Diagnosis and planning pause simulation advancement. After a correction, the system refreshes observations, checks completion and may return control to the policy under the recovery gates. We log recovery requests, executed corrections and final episode outcomes separately. Successful continuations provide action supervision from imperfect states that nominal demonstrations may omit. Recovery supplies corrective training experience from policy-failed states, while performance evaluation uses the learned policy alone.

3.3.3 Adaptive Data Collection

Agent-guided data collection adapts the collection distribution to the current policy's diagnosed weaknesses. After each collection round, the agent attributes final failures to the source-object reset, target-anchor reset or neither; stored attributions can be reused without additional VLM calls. For reset subject s with a non-degenerate, configuration-specific range $[\ell^s, h^s]$, we normalize its position to a common grid and compute the attributed-failure rate:

$$u^s(x,y) = \left(\frac{x - \ell_x^s}{h_x^s - \ell_x^s}, \frac{y - \ell_y^s}{h_y^s - \ell_y^s} \right), \qquad \widehat{r}_{k,b}^s = \frac{U_{k,b}^s}{N_{k,b}^s}. \tag{3}$$

Here, $N_{k,b}^s$ counts completed monitored episodes with valid subject positions in cell b, whereas $U_{k,b}^s$ counts only final failures with supported attribution to s. Cell selection requires sufficient observations

Algorithm 1 EmbodiRSI: recursive policy self-improvement

Require: Deployment observations and goals, base distribution p_0, round/data budgets
1: Construct the simulator; compile validated task configurations
2: Collect initial demonstrations $\mathcal{D}_0$ and train π_0
3: **for** each scheduled RSI round k **do**
4: Roll out π_k; diagnose execution and request eligible state-conditioned recovery
5: Retain rollout-derived episodes $\mathcal{D}_k^{\text{roll}}$, counting each episode once
6: Attribute final failures; form p_{k+1} by Eq. (4)
7: Generate and validate new demonstrations $\mathcal{D}_k^{\text{new}}$ under p_{k+1}
8: Accumulate $\mathcal{D}_{k+1}$; annotate and train π_{k+1} by Eqs. (5)–(7)
9: **end for**
10: Refine the selected policy with a small real-world human-in-the-loop (HIL) dataset
11: Evaluate independently with the learned policy alone

and attributed failures, together with a rate exceeding the subject-level global rate. Episodes that succeed after recovery do not contribute to the numerator; these statistics describe assisted collection rather than unassisted test performance.

Selected cells are mapped back to world coordinates to restrict the corresponding reset ranges, including yaw when sufficiently supported. Configurations for subsequent demonstration generation are sampled from

$$p_{k+1}(c) = (1-\alpha)p_0(c) + \alpha p_{\text{hard},k}(c), \qquad 0 \le \alpha \le 1. \tag{4}$$

Here, $p_{\text{hard},k}$ concentrates sampling on diagnosed weak configurations, while the base component preserves access to the original range for $\alpha < 1$. The mixture controls configuration sampling but does not determine retained-trajectory proportions, which also depend on execution-based acceptance. This refinement applies only to training collection; test resets remain unchanged. Failure concentration provides a heuristic for prioritizing subsequent data acquisition (Appendix A.3).

3.3.4 Policy Updating and Recursion

Let $\mathcal{D}_k^{\text{roll}}$ contain retained rollout-derived episodes, which may include both policy actions and corrections, and let $\mathcal{D}_k^{\text{new}}$ contain validated demonstrations generated under p_{k+1}. Each saved episode is counted once. The next dataset is

$$\mathcal{D}_{k+1} = \mathcal{D}_k \cup \mathcal{D}_k^{\text{roll}} \cup \mathcal{D}_k^{\text{new}}. \tag{5}$$

The policy receives head- and wrist-camera RGB images, a 14-dimensional robot state (six arm-joint positions and one gripper value per arm) and an instruction. Following RECAP (Intelligence et al., 2025a), a task-conditioned value model annotates the accumulated experience:

$$\widehat{A}_t^{(n)} = R_t^{(n)} + V_\phi(o_{t+n}, l) - V_\phi(o_t, l), \qquad z_t = \mathbb{I}[\widehat{A}_t^{(n)} > \delta]. \tag{6}$$

The implementation sets $R_t^{(n)} = 0$, $n = 50$, so the binary condition depends on the value-estimate change. With $\widetilde{\mathcal{D}}_{k+1}$ denoting the annotated data, the policy update minimizes

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{(o_t,l,a_t,z_t)\sim\widetilde{\mathcal{D}}_{k+1}} \left[\ell_{\text{BC}}\big(\pi_\theta(\cdot \mid o_t, l, z_t), a_t\big)\right]. \tag{7}$$

Recorded actions remain the supervision targets, while advantage labels are provided in text form as condition for action prediction. This formulation integrates RSI-generated experience into policy learning. The updated policy π_{k+1} then generates the next round's rollouts, which guide the refinement of both corrective supervision and subsequent data collection. The number of rounds and retained-data quotas are fixed in advance; feedback guides experience acquisition within this schedule.

Coordination and real-world adaptation. At the outer level, an OpenClaw-based supervisor coordinates the complete RSI workflow through a Model Context Protocol (MCP) interface. (Appendix A.4). After simulation RSI, ten retained real-world human-in-the-loop (HIL) trajectories per subtask refine the selected policy with the same learning formulation. Operators correct residual execution errors, such as unstable grasps and inaccurate placement. Final simulation and physical tests are independent of training collection and disable agent takeover, planner recovery and human correction. A test-time reattempt is therefore behavior learned by the policy itself.



Figure 3: **Paired physical and simulated experimental settings.** Columns from left to right show Office Table (S1), Kitchen Table (S2), and Household Table (S3). The top row shows the physical workspaces, and the bottom row shows their corresponding simulated environments. Each column presents the same scene in the real world and simulation.

4 EXPERIMENTS

Experimental settings. We use a 14-DoF dual-arm ALOHA (Zhao et al., 2023) with two wrist cameras and one head camera, matching the simulation observations. The paired workspaces comprise Office Table (S1, six subtasks), Kitchen Table (S2, four) and Household Table (S3, four) (Figure 3). Each training setting uses one policy per scene. R0 is initialized from the pretrained $\pi_{0.5}$ checkpoint Intelligence et al. (2025b). Each subsequent RSI round continues training from the preceding round's policy using the accumulated dataset. The value model uses the vision–language backbone initialized from the same checkpoint, augmented with a value prediction head. We build our agent system based on GPT-5.5 and use Isaac Sim as the simulation platform. Subtasks share simulation/physical goal definitions (Table C.1) and are evaluated individually, not as complete tidying sequences.

Training budgets and evaluation. R0/R1/R2 use 50/200/400 cumulative *retained simulated training trajectories per subtask*. R0 is the seed policy; R1 and R2 follow collection-and-training updates. +10 adds ten real-world human-in-the-loop (HIL) refinement trajectories per subtask. All tests are policy-only, without external recovery. We run ten physical trials per task and setting; simulation test counts vary by task–checkpoint pair. Scene-balanced means weight scenes equally. Appendix D provides allocation and statistical details.

4.1 DOES THE POLICY IMPROVE ACROSS RSI ROUNDS?

Scene-balanced autonomous simulation success increases from 50.4% at R0 to 70.7% at R1 and 83.5% at R2 (Figure 4). All 14 subtasks improve from R0 to R2, with substantial gains on initially challenging tasks. For example, black-pen placement in S1 improves from 14.0% to 76.5%, whereas carrot placement in S2 starts near ceiling, leaving less room for improvement. These results demonstrate that EmbodiRSI's evaluated RSI rounds yield progressively stronger autonomous policies.

Qualitative observations are consistent with EmbodiRSI's two feedback paths. Early policies often continued nominal transport after missed grasps, whereas later policies grasped more reliably and sometimes reattempted unsuccessful interactions, with some reattempts leading to task completion. Further details are provided in Appendix E.

4.2 DOES FEEDBACK HELP AT MATCHED TRAJECTORY COUNTS?

We compare EmbodiRSI with Static collection in S1 and S3, the two more difficult scenes at initialization. Both settings start from the same 50 demonstrations per subtask and use the same

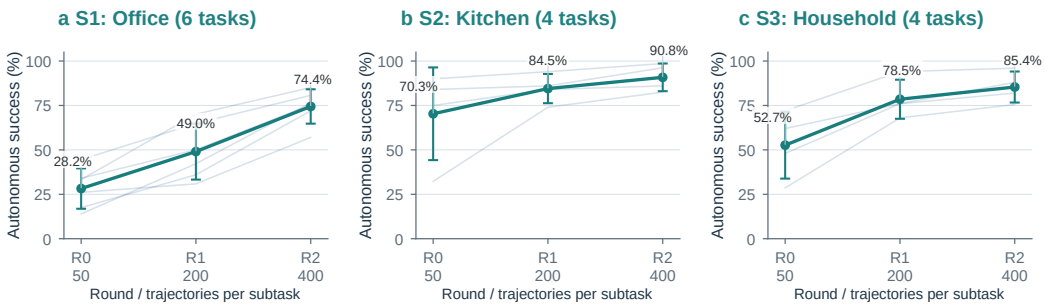


Figure 4: **Autonomous policy performance across RSI rounds.** Thin lines show individual subtasks; markers and error bars indicate scene means ± sample SD across subtasks. Discrete rounds R0/R1/R2 use 50/200/400 retained simulated training trajectories per subtask, respectively.

Table 1: **Effect of RSI under equal data budgets.** Scene-balanced autonomous success (%) across S1 and S3 at 400 simulated trajectories per subtask. Complete 200/400 comparisons, scene-level dispersion and task-level exceptions are retained in Appendix E.

Method	Simulation	R2S2R	R2S2R + Human in the Loop
EmbodiRSI w/o RSI	59.3	46.3	67.9
EmbodiRSI	**79.9**	**59.2**	**82.1**
Gain (pp)	+20.6	+12.9	+14.2

policy architecture and training recipe. Static continues generating demonstrations under the original configuration distribution; EmbodiRSI adds rollout-derived experience and targeted recollection. Cumulative retained counts are matched at 200 and 400 trajectories per subtask.

At a matched budget of 400 simulated trajectories per subtask, EmbodiRSI's adaptive collection achieves 79.9% scene-balanced simulation success, compared with 59.3% for Static—a gain of 20.6 percentage points (Table 1). Adaptive collection yields higher scene-level mean success in both scenes at both evaluated budgets. This advantage extends to real-world execution, reaching 12.9 percentage points before physical refinement and 14.2 points after both policies are refined with ten human-in-the-loop (HIL) trajectories per subtask. The benefits of adaptive simulation collection therefore persist after limited real-world refinement. Detailed task-level simulation results are provided in Appendix E.

4.3 Does RSI Enable Data-Efficient Real-World Adaptation?

With 400 simulated and ten real-world refinement trajectories per subtask, EmbodiRSI reaches 83.1% scene-balanced autonomous real-world success, compared with 75.0% for Real-only adaptation using 200 real demonstrations per subtask (Table 2). Its observed mean is higher in each scene, using one-twentieth as many retained physical training trajectories.

Before refinement, R2 already achieves 62.8% physical success. Here *zero-shot* means no task-specific real-world policy refinement. The policy generally approaches the instructed object, while remaining failures concentrate near grasping and placement. HIL corrections address these residual errors rather than providing the entire strategy from scratch. For example, S1 red-pen placement remains at 2/10 before refinement at both simulation checkpoints but reaches 7/10 after R2 refinement; glasses placement already achieves 10/10 before refinement.

These patterns suggest complementary roles: simulation RSI develops scene-specific capability through repeated practice and corrective experience, while physical refinement addresses residual deployment errors. Figure E.2 contrasts some real-world executions; Appendix E retains all task-level outcomes and sampling intervals. Reattempts during these tests are actions of the learned policy, not training-time agent recovery. Further details are provided in Appendix E.

Table 2: **Data-efficient real-world adaptation.** Success rates (%) are mean ± sample SD across subtasks; the final column weights scenes equally. R1/R2 use 200/400 simulated trajectories per subtask; +10 adds ten real human-in-the-loop (HIL) trajectories. Real-only labels give demonstration counts per subtask. Each subtask/setting has ten autonomous test trials.

Training setting	**S1: Office** $n = 6$ tasks	**S2: Kitchen** $n = 4$ tasks	**S3: Household** $n = 4$ tasks	**Scene-balanced mean**
R1	43.3 ± 22.5	60.0 ± 8.2	52.5 ± 9.6	51.9
R1+10	63.3 ± 13.7	82.5 ± 5.0	75.0 ± 17.3	73.6
R2	53.3 ± 32.0	70.0 ± 14.1	65.0 ± 12.9	62.8
R2+10	$\mathbf{81.7 \pm 13.3}$	$\mathbf{85.0 \pm 12.9}$	$\mathbf{82.5 \pm 9.6}$	**83.1**
Real-only 10	13.3 ± 13.7	45.0 ± 5.8	20.0 ± 8.2	26.1
Real-only 100	56.7 ± 15.1	62.5 ± 5.0	52.5 ± 5.0	57.2
Real-only 200	70.0 ± 6.3	77.5 ± 9.6	77.5 ± 9.6	75.0

5 LIMITATIONS

Within the available time and computational budget, we prioritized evaluating scene-specific adaptation across the selected workspaces. We plan to extend evaluation to held-out workspaces and longer improvement runs to assess generalization and sustained policy improvement.

Our evaluation centers on the complete pipeline, in which recovery, recollection and agent reasoning operate jointly. While component-wise attribution is not the primary focus of this system-level study, we plan targeted ablations to better understand their contributions and interactions, informing further refinement of the pipeline.

6 CONCLUSION

EmbodiRSI enables recursive policy self-improvement within a real-to-sim-to-real (R2S2R) workflow by making experience acquisition responsive to the evolving policy. With agent assistance, Collaborative Error Correction provides corrective continuations from policy-failed states, while Adaptive Data Collection directs practice toward persistent weaknesses. Supported by task-specific simulation reconstructed from real-world observations and validated trajectory generation, these mechanisms turn rollout feedback into training experience for successive policy updates. After recursive improvement in simulation, the policy is refined with limited real-world data for autonomous deployment in the corresponding physical workspace. Experiments demonstrate improved autonomous performance across the evaluated rounds, gains over static collection at matched retained-trajectory counts, and data-efficient real-world adaptation. Together, these results highlight the value of refining not only the policy, but also the experience that drives its improvement.

AI USE STATEMENT

Generative AI assisted earlier manuscript restructuring, language revision, literature checks and descriptive analysis code. Additional AI-assisted editing prepared this ICLR version polishing, including language revision, interpretation of existing results, LaTeX formatting, figure-layout exploration and editable vector diagrams. Submitted experimental images are retained from the supplied materials; generated layout concepts are not experimental evidence. Manuscript preparation introduced no new robot trials or simulation measurements. The diagrams are method illustrations, not generated experimental evidence. The reconstruction gallery includes labeled AI-generated scene inputs for illustration, not evaluation. The human authors retain responsibility for the final text, claims, references, implementation details and AI-use disclosure.

REPRODUCIBILITY STATEMENT

The full code for the entire pipeline will be released soon.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauzá, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7220–7232, 2026.

Prithwish Dan, Kushal Kedia, Angela Chao, Edward Weiyi Duan, Maximus Adrian Pace, Wei-Chiu Ma, and Sanjiban Choudhury. X-sim: Cross-embodiment learning via real-to-sim-to-real. *arXiv preprint arXiv:2505.07096*, 2025.

Yu Fang, Yue Yang, Xinghao Zhu, Kaiyuan Zheng, Gedas Bertasius, Daniel Szafir, and Mingyu Ding. Rebot: Scaling robot learning with real-to-sim-to-real robotic video synthesis. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11351–11358. IEEE, 2025.

Haotian Fu, Ran Gong, Xiaohan Zhang, Maria Vittoria Minniti, Jigarkumar Patel, and Karl Schmeckpeper. Data-efficient multitask dagger. *arXiv preprint arXiv:2509.25466*, 2025.

Caelan Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment. *arXiv preprint arXiv:2410.18907*, 2024.

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.

Ryan Hoque, Ajay Mandlekar, Caelan Garrett, Ken Goldberg, and Dieter Fox. Intervengen: Interventional data generation for robust and data-efficient robot imitation learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2840–2846. IEEE, 2024.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

Yixuan Huang, Novella Alvina, Mohanraj Devendran Shanthi, and Tucker Hermans. Fail2progress: Learning from real-world robot failures with stein variational inference. *arXiv preprint arXiv:2509.01746*, 2025.

Dongjie Huo, Haoyun Liu, Guoqing Liu, Dekang Qi, Zhiming Sun, Maoguo Gao, Jianxin He, Yandan Yang, Xinyuan Chang, Feng Xiong, et al. Abot-claw: A foundation for persistent, cooperative, and self-evolving robotic agents. *arXiv preprint arXiv:2604.10096*, 2026.

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*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Guangqi Jiang, Haoran Chang, Ri-Zhao Qiu, Yutong Liang, Mazeyu Ji, Jiyue Zhu, Zhao Dong, Xueyan Zou, and Xiaolong Wang. Gsworld: Closed-loop photo-realistic simulation suite for robotic manipulation. *arXiv preprint arXiv:2510.20813*, 2025a.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16923–16930. IEEE, 2025b.

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.

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, et al. Roboclaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026.

Xinhai Li, Jialin Li, Ziheng Zhang, Rui Zhang, Fan Jia, Tiancai Wang, Haoqiang Fan, Kuo-Kun Tseng, and Ruiping Wang. Robogsim: A real2sim2real robotic gaussian splatting simulator. *arXiv preprint arXiv:2411.11839*, 2024.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. Failsafe: Reasoning and recovery from failures in vision-language-action models. *arXiv preprint arXiv:2510.01642*, 2025.

Yaru Liu, Ao-bo Wang, and Nanyang Ye. V-cage: Vision-closed-loop agentic generation engine for robotic manipulation. *arXiv preprint arXiv:2604.09036*, 2026.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. Robotwin: Dual-arm robot benchmark with generative digital twins. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27649–27660. IEEE, 2025.

Adithyavairavan Murali, Balakumar Sundaralingam, Yu-Wei Chao, Wentao Yuan, Jun Yamada, Mark Carlson, Fabio Ramos, Stan Birchfield, Dieter Fox, and Clemens Eppner. Graspgen: A diffusion-based framework for 6-dof grasping with on-generator training. *arXiv preprint arXiv:2507.13097*, 2025.

Shivansh Patel, Xinchen Yin, Wenlong Huang, Shubham Garg, Hooshang Nayyeri, Li Fei-Fei, Svetlana Lazebnik, and Yunzhu Li. A real-to-sim-to-real approach to robotic manipulation with vlm-generated iterative keypoint rewards. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8258–8266. IEEE, 2025.

Nicholas Pfaff, Evelyn Fu, Jeremy Binagia, Phillip Isola, and Russ Tedrake. Scalable real2sim: Physics-aware asset generation via robotic pick-and-place setups. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6296–6303. IEEE, 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.

Balakumar Sundaralingam, Adithyavairavan Murali, and Stan Birchfield. curobov2: Dynamics-aware motion generation with depth-fused distance fields for high-dof robots. *arXiv preprint arXiv:2603.05493*, 2026.

Tang Tao, Likui Zhang, Youpeng Wen, Kaidong Zhang, Jia-Wang Bian, Xia Zhou, Tianyi Yan, Kun Zhan, Peng Jia, Hefeng Wu, et al. Robopearls: editable video simulation for robot manipulation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1–12. IEEE, 2025.

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.

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4890–4924, 2024.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

Zhihao Wang, Jianxiong Li, Jinliang Zheng, Wencong Zhang, Dongxiu Liu, Yinan Zheng, Haoyi Niu, Junzhi Yu, and Xianyuan Zhan. Physiagent: An embodied agent framework in physical world. *arXiv preprint arXiv:2509.24524*, 2025.

Yuxuan Wu, Lei Pan, Wenhua Wu, Guangming Wang, Yanzi Miao, Fan Xu, and Hesheng Wang. Rl-gsbridge: 3d gaussian splatting based real2sim2real method for robotic manipulation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 192–198. IEEE, 2025.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, et al. Sage: Scalable agentic 3d scene generation for embodied ai. *arXiv preprint arXiv:2602.10116*, 2026.

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.

Sheng Xu, Ruixing Jin, Huayi Zhou, Bo Yue, Guanren Qiao, Yunxin Tai, Yueci Deng, Kui Jia, and Guiliang Liu. From reaction to anticipation: Proactive failure recovery through agentic task graph for robotic manipulation. *arXiv preprint arXiv:2605.11951*, 2026.

Zhengrong Xue, Shuying Deng, Zhenyang Chen, Yixuan Wang, Zhecheng Yuan, and Huazhe Xu. Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning. *arXiv preprint arXiv:2502.16932*, 2025.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, et al. Rise: Self-improving robot policy with compositional world model. *arXiv preprint arXiv:2602.11075*, 2026.

Kaifeng Zhang, Shuo Sha, Hanxiao Jiang, Matthew Loper, Hyunjong Song, Guangyan Cai, Zhuo Xu, Xiaochen Hu, Changxi Zheng, and Yunzhu Li. Real-to-sim robot policy evaluation with gaussian splatting simulation of soft-body interactions. *arXiv preprint arXiv:2511.04665*, 2025.

Mingtong Zhang and Dhruv Shah. Visual verification enables inference-time steering and autonomous policy improvement. *arXiv preprint arXiv:2606.18247*, 2026.

Haoyu Zhao, Cheng Zeng, Linghao Zhuang, Yaxi Zhao, Shengke Xue, Hao Wang, Xingyue Zhao, Zhongyu Li, Kehan Li, Siteng Huang, et al. High-fidelity simulated data generation for real-world zero-shot robotic manipulation learning with gaussian splatting. *IEEE Robotics and Automation Letters*, 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.

Xueyang Zhou, Yihan Sun, Xijie Gong, Guiyao Tie, Pan Zhou, Lichao Sun, and Yongchao Chen. Embodiedclaw: Conversational workflow execution for embodied ai development. *arXiv preprint arXiv:2604.13800*, 2026.

Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. *arXiv preprint arXiv:2407.20635*, 2024.

A RSI IMPLEMENTATION AND AGENT COORDINATION

This appendix expands the RSI loop in Section 3.3. It documents state-conditioned handover, failure-attribution records and supervisory coordination, distinguishing proposed operations, executed corrections and final outcomes. Environment and trajectory construction are detailed separately in Appendix B.

A.1 TASK INTERFACES AND RECOVERY HANDOVER

Task configurations associate the manipulated object and target anchor with a language instruction, grasp candidates, an ordered robot program and reset groups. Each associated waypoint follows the transformation of its source or target object. Members of a reset group share a sampled transformation, while sequential placement and collision handling can introduce dependencies between groups. These associations allow the same task-relative program to be re-grounded after a reset without specifying a new task goal.

The object-route generator returns intermediate world-frame poses with stage labels for lift, optional bypass and pre-placement. Its structured-output checks test object identity, numerical validity, pose representation and stage ordering before geometric review. Failed checks provide feedback for candidate revision. This separates a proposed route from its validation and subsequent execution; a plausible textual proposal alone is not an executable demonstration.

During monitored simulation collection, visual diagnosis and recovery planning are synchronous: the control loop does not advance the simulation while waiting for these operations. Recovery uses the current object and robot states. On return from a correction, observations are refreshed and completion is checked again; the continuation path can return control to the policy under the configured recovery gates. Reaching a feasible entry pose does not guarantee completion of the remaining manipulation.

Event records distinguish a recovery request, any corrective actions actually executed, and the final episode outcome. A request can fail before any correction is executed, while a partially executed correction can be followed by policy completion. Thus, triggering recovery, executing recovery and passing the final task check are distinct events. These training-collection events are also distinct from the policy-generated reattempts observed during unassisted evaluation.

A.2 RECOVERY EXPERIENCE WITHIN THE RSI LOOP

Algorithm 1 in Section 3 summarizes the learning loop. Figure A.1 illustrates its state-conditioned recovery step during training-data collection, distinct from policy-generated reattempts during testing.



Figure A.1: **Corrective experience from a simulated intermediate state.** The frames show an imperfect state, motion replanning and corrective execution during training-data collection. Externally generated recovery is disabled in performance tests; policy-generated reattempts remain part of autonomous execution.

A.3 GRID ATTRIBUTION RECORDS AND COUNTING

The grid analysis uses structured rollout records, including reset positions, final outcomes and subject-attribution labels. A VLM request can assign a failure to the source-object reset, target-anchor reset or neither; the request uses the recorded execution evidence rather than requiring a new replay of the rollout video. Existing attribution records can be reused, so issuing a new VLM request is optional and does not replace the normalized-grid procedure. For each subject and cell, valid completed episodes contribute to the denominator, whereas only final failures with supported attribution to that subject contribute to the numerator. Low-confidence and unassigned failures do not increase the corresponding attributed-failure count. An episode that finishes successfully after recovery is not counted as a final failure. These statistics describe recorded outcomes, not a counterfactual estimate of whether an assisted episode would have succeeded without intervention.

A.4 SUPERVISOR SCHEDULING AND OUTPUT VALIDATION

Skills and workflow state. The supervisor coordinates the workflow through a typed skill library,

$$\mathcal{K} = \{k^{\text{scene}}, k^{\text{capture}}, k^{\text{grasp}}, k^{\text{config}}, k^{\text{motion}}, k^{\text{train}}, k^{\text{eval}}\}, \tag{A.1}$$

covering scene construction, observation capture, grasp inference and verification, task-configuration generation, motion synthesis, policy training and monitored evaluation. Each skill specifies typed inputs, explicit preconditions, an execution backend and the outputs needed by downstream operations. The persistent workflow state is

$$s_t = (g, b, r, \mathbf{z}_t, \mathcal{J}_t, \mathcal{A}_t), \tag{A.2}$$

where g is the user-specified objective, b the requested stopping boundary, r the scene and run identifier, $\mathbf{z}_t$ the stage states, $\mathcal{J}_t$ the asynchronous jobs, and $\mathcal{A}_t$ the indexed artifacts. Before execution, the supervisor constructs the dependency chain, assigns stage-specific computing resources and fixes the requested stopping boundary.

Eligibility and stage completion. At time t, eligible skills satisfy

$$\mathcal{K}_t^{\text{ready}} = \{k_i \in \mathcal{K} \mid \text{Pre}_i(s_t) = 1,\ \text{Valid}(\mathcal{A}_i^{\text{req}}) = 1,\ k_i \preceq b\}. \tag{A.3}$$

Here Pre_i represents dependency and resource constraints, $\mathcal{A}_i^{\text{req}}$ is the required upstream artifact set, and $k_i \preceq b$ prevents execution beyond the requested boundary. The supervisor selects $k_t = \pi_{\text{sup}}(s_t, \mathcal{K}_t^{\text{ready}})$. Independent scene- or object-level jobs can be dispatched in parallel on different devices. Long-running skills return a durable job handle and are monitored through their runtime state, logs and artifact manifests. Successful completion requires both a normal process exit and valid required outputs:

$$\text{Complete}(k_i) = \mathbb{I}[e_i = 0] \prod_{a \in \mathcal{A}_i^{\text{out}}} \mathbb{I}[\text{Valid}(a) = 1]. \tag{A.4}$$

Here e_i is the exit code and $\mathcal{A}_i^{\text{out}}$ contains outputs required downstream. Persistent services use readiness checks rather than process-exit semantics.

Failure handling. When a precondition or artifact check fails, the supervisor localizes the earliest invalid dependency and requests the corresponding upstream repair, rather than restarting the entire workflow. Transient communication failures are handled by recovering the latest durable job and stage state before making a new scheduling decision. These runtime checks support the scene-to-policy coordination described in Section 3 without treating a process exit or an accepted tool request as sufficient evidence of task completion.

B ENVIRONMENT AND TRAJECTORY INFRASTRUCTURE

The simulator and trajectory generator provide the reusable infrastructure for RSI. This appendix retains the scene compilation, collision rectification, trajectory validation and reset-aware execution details summarized in Sections 3.1 and 3.2. These tools serve initial collection, state-conditioned recovery and targeted recollection; they are not separate deployment-time controllers.

B.1 AGENTIC REAL2SIM SCENE CONSTRUCTION AND REFINEMENT

A reconstructed workspace is useful for learning only when it supports both physically valid interactions and explicit task goals. We therefore formulate Real2Sim construction as an *agentic scene compiler*: visual and semantic reasoning specifies the objects and their intended relations, while geometric tools implement and check the corresponding scene operations. Given a tabletop image, optionally accompanied by a target image and language prompt, the compiler produces an initial scene and an organized target layout (Figure B.1). Their difference defines the object-level goals for demonstration generation.

Object-centric reconstruction. A vision–language model (VLM) identifies the table and visible movable objects in the input image I_0. SAM 3 produces instance masks and SAM 3D reconstructs the corresponding textured meshes (Carion et al., 2026; Chen et al., 2026). The scene is represented as

$$\mathcal{S}_0 = \{o_T, o_1, \ldots, o_N\}, \qquad o_i = (c_i, m_i, T_i^0), \tag{B.1}$$

where o_T is the table, and c_i, m_i and $T_i^0 \in \mathrm{SE}(3)$ are an object's semantic category, mesh and initial pose. Object identity is maintained through scene editing and trajectory generation, so that an instruction refers to the same object in the rendered workspace, geometric checks and robot program.

Simulation instantiation. The scene agent configures the reconstructed assets for simulation, setting and adjusting object scale, poses, and physical properties to obtain an executable task environment. The resulting scene undergoes collision and support checks before task and trajectory generation.

Support-aware collision rectification. Imperfect reconstruction can leave objects interpenetrating, particularly in cluttered tabletop scenes. Agent-Guided Adaptive Collision Rectification (AACR) resolves these collisions while preserving the support and containment relations inferred from I_0. The agent first partitions the non-table objects into support-aware groups $\mathcal{G} = \{G_1, \ldots, G_K\}$. *Parallel Horizontal Rectification* separates intersecting groups by selecting a central root and moving other groups outward. A group moves as a unit, preserving its internal relations; only groups with disjoint swept regions are adjusted in the same parallel batch. The settled set is then updated before the next batch is processed.

Adaptive Intra-group Rectification resolves the remaining collisions within each group. The agent selects a correction according to the object relation: horizontal separation for objects sharing a support, vertical adjustment for containment, or bounded tilt for insertion-like arrangements. The selected geometric operator applies and validates the pose change. Thus, semantic reasoning determines which adjustment is appropriate, rather than replacing collision checking. The group-selection and displacement rules are given in Section B.3 below.

Task-oriented scene organization. The scene agent compares the original image I_0 with a rendering of the rectified scene and generates an organization instruction $\mathcal{L}$. A supplied target image or language prompt can further specify the intended layout. The agent first uses relation-specific tools, such as `on-top-of`, `inside` and `beside`, to establish a coarse arrangement. It then refines object translations and rotations, with individual object agents operating in parallel. Rendered views and structured object states make the consequences of an edit available for critique and revision. This coarse-to-fine process separates deciding *which relation should hold* from finding poses that realize that relation without invalid geometry.

The accepted target scene is

$$\mathcal{S}_{\mathrm{org}} = \{(c_i, m_i, T_i^{\mathrm{org}})\}_{i \in \{T,1,\ldots,N\}}. \tag{B.2}$$

The initial and organized layouts specify what the robot should change. During task compilation, these goals are associated with language instructions, grasps, waypoints and reset groups. Objects in a reset group share a sampled transformation. Source- and target-associated waypoints subsequently follow their respective anchors, allowing the same task to be practiced under different configurations without redefining its goal.

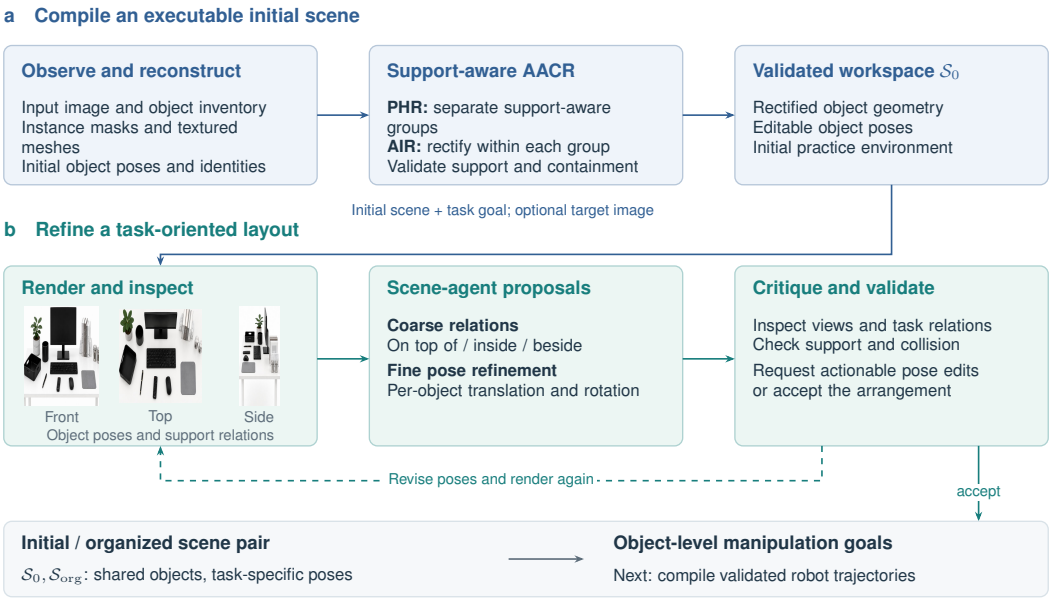


Figure B.1: **From deployment observations to task-defining scenes.** Object reconstruction and support-aware collision rectification produce the initial scene. The scene agent iteratively refines object poses using multi-view feedback and geometric validation to obtain the target scene. The scene pair defines manipulation goals for trajectory generation.

B.2 Agentic Trajectory Generation

The scene pair defines the desired transition, but not a feasible way for the robot to execute it. The trajectory-generation agent $\mathcal{A}_{\text{traj}}$ bridges this gap by proposing object motions, projecting them through candidate grasps and validating the resulting robot programs. Its interface is

$$\mathcal{A}_{\text{traj}} : (\mathcal{S}_0, \mathcal{S}_{\text{org}}, l) \longmapsto (\mathcal{C}, \mathcal{D}_{\text{demo}}, \mathcal{E}), \tag{B.3}$$

where l is the task instruction, $\mathcal{C}$ is a library of executable configurations, $\mathcal{D}_{\text{demo}}$ contains accepted demonstrations and $\mathcal{E}$ records validation evidence. The same trajectory tools later support recovery from policy-failed states and recollection under revised reset ranges. Demonstration generation is therefore a reusable part of the learning loop, not a one-time preprocessing stage.

Object-centric task compilation. The agent compares $\mathcal{S}_0$ and $\mathcal{S}_{\text{org}}$, identifies objects whose states must change and compiles an ordered list of subtasks,

$$\mathcal{T} = (\chi_j)_{j=1}^{M}, \qquad \chi_j = (o_{i_j}, b_j, \rho_j). \tag{B.4}$$

Here o_{i_j} is the manipulated object, b_j its target anchor or support, and ρ_j the desired relation. The order accounts for accessibility, target occupancy and earlier placements; the subtask count M need not equal the number of reconstructed objects N. For each subtask, a VLM planner proposes object-space routes comprising lift, optional bypass and pre-placement poses. These routes are simplified, refined for clearance and densified to bound translation and rotation between consecutive states. Planning in object space allows the intended motion to be evaluated before a particular grasp is chosen.

Grasp-conditioned motion synthesis. GraspGen supplies candidate grasps (Murali et al., 2025), which are screened for grasp quality, endpoint reachability and collision clearance. For a grasp $T^0_{g,i}$ on object i, the object-relative gripper transform is preserved along an object route:

$$T_{o_i \to g} = (T^0_{o_i})^{-1} T^0_{g,i}, \qquad T^j_{g,i} = T^j_{o_i} T_{o_i \to g}. \tag{B.5}$$

An accepted route can thus be reused across grasp candidates without regenerating its semantic path. Pick and place are checked jointly. When an endpoint pair is unreachable, pick and place poses

are repaired jointly using bounded $\mathrm{SE}(3)$ adjustments. The repaired poses must satisfy the shared object-relative grasp relation in Eq. (B.5). Pre-grasp, gripper closure, release and post-release retract complete the waypoint program.

Validation-guided repair. A plausible object route does not by itself establish a valid demonstration. EmbodiRSI uses three checks before admitting a configuration to $\mathcal{C}$. First, a swept full-mesh test checks the carried object along each interpolated segment, with narrowly scoped exemptions for intended support and target contact. Second, multi-view renderings allow a VLM reviewer to assess route efficiency, orientation consistency and target approach. Third, cuRobo validates the complete robot program sequentially, initializing each transition from the preceding terminal joint state (Sundaralingam et al., 2026). Because the carried object's swept geometry is checked separately, its stale source-pose mesh is disabled after grasping; robot collision checking against the rest of the scene remains active.

When a transition is infeasible, the agent first attempts route contraction, then searches bounded local pose adjustments around the first infeasible waypoint. A repaired candidate must pass the geometric and sequential-motion checks again, together with renewed visual review when its object route changes. Only programs feasible through post-release retract are accepted. These checks connect high-level proposals to executable supervision while allowing invalid candidates to be repaired rather than treated as training data.

Reset-aware execution and data curation. At collection time, scene objects are randomized within their admissible reset ranges. Once physics has settled, each stored waypoint is re-grounded to the measured pose of its associated source or target anchor $r(j)$:

$$\widetilde{T}^{j}_{ee} = T^{\mathrm{cur}}_{r(j)} (\overline{T}_{r(j)})^{-1} \overline{T}^{j}_{ee}, \tag{B.6}$$

where overbars denote poses stored in the configuration. Each segment is replanned from the measured joint state and then executed. Transit motions permit broader local pose search, whereas grasp and placement use tightly bounded planar fallbacks to preserve the task relation.

Execution supplies a fourth validation layer. The system checks the object–gripper relation after grasping, verifies transport at designated waypoints and compares the settled object pose after release with the reset-adjusted target. A generated demonstration is retained only when

$$\mathrm{Accept}(\tau) = V_{\mathrm{mesh}} \wedge V_{\mathrm{view}} \wedge V_{\mathrm{motion}} \wedge V_{\mathrm{exec}}. \tag{B.7}$$

The terms denote swept-mesh, visual-review, sequential-motion and execution-time validity. Failed demonstrations are rejected and resampled. Accepted records contain multi-camera observations, joint states, velocities, efforts and commanded actions in LeRobot format. Together, proposal, validation, repair and execution turn an object-level task into varied training demonstrations without manual waypoint scripting.

B.3 Scene geometry

Parallel Horizontal Rectification. AACR partitions non-table objects into support-aware groups $\mathcal{G} = \{G_1, \ldots, G_K\}$. PHR treats each group as a unit while resolving inter-group overlap. Let $\mathbf{p}_k$ denote the center of G_k projected onto the tabletop plane, represented in world coordinates with zero vertical component. The root minimizes the sum of planar distances to the other groups; each non-root group is assigned the corresponding outward direction:

$$r = \arg\min_{k} \sum_{j \neq k} \|\mathbf{p}_k - \mathbf{p}_j\|_2, \qquad \mathbf{d}_k = \frac{\mathbf{p}_k - \mathbf{p}_r}{\|\mathbf{p}_k - \mathbf{p}_r\|_2}, \quad k \neq r,\ \mathbf{p}_k \neq \mathbf{p}_r. \tag{B.8}$$

The displacement $\delta \mathbf{d}_k$ is horizontal. Starting from the settled anchor set $\mathcal{H}_0 = \{G_r\}$, each unresolved group is assigned the minimum displacement that removes overlap with the settled groups:

$$\delta^*_k = \min\{\delta \geq 0 \mid (G_k + \delta \mathbf{d}_k) \cap \mathcal{H}_t = \emptyset\}, \qquad \widehat{G}_k = G_k + \delta^*_k \mathbf{d}_k. \tag{B.9}$$

Intersections refer to occupied geometry. A conflict-aware parallel batch $\mathcal{B}_t$ contains only groups whose proposed swept regions are disjoint:

$$\mathrm{Sweep}(G_i, \widehat{G}_i) \cap \mathrm{Sweep}(G_j, \widehat{G}_j) = \emptyset, \qquad G_i, G_j \in \mathcal{B}_t,\ i \neq j. \tag{B.10}$$

Here Sweep denotes the spatial region traversed during the proposed translation. The selected groups are moved simultaneously, and the settled set is updated as

$$\mathcal{H}_{t+1} = \mathcal{H}_t \cup \{\widehat{G}_k \mid G_k \in \mathcal{B}_t\}. \tag{B.11}$$

The equations specify the non-degenerate geometry. AIR subsequently addresses within-group overlap using the relation-specific horizontal, vertical and bounded-tilt corrections described in Section B.1.

B.4 Sequential motion execution

Robot transitions are planned sequentially, with the terminal joint state of one transition used to initialize the next:

$$\xi^*_{i,k} = \mathrm{Plan}(q_{i,k-1}, T^k_{g,i}, \mathcal{W}_{i,k}), \qquad q_{i,k} = \mathrm{Last}(\xi^*_{i,k}). \tag{B.12}$$

Here $\mathcal{W}_{i,k}$ is the collision world used for the transition. At collection time, Eq. (B.6) re-grounds each waypoint using the current pose of its source object or target anchor. This preserves task-relative motion after a reset. The corresponding geometry and motion checks are repeated after a repair; visual review is repeated when the object route changes.

C Task environments and scene reconstruction results

C.1 Robot platform and evaluated tasks

Section 4 and Figure 3 present the robot platform and paired physical/simulated workspaces. Table C.1 lists all 14 object-level task goals shared by each simulation–real pair. One policy covers the subtasks of each scene and training setting; tasks are evaluated separately. The matched-collection comparison uses S1 and S3.

Table C.1: **Manipulation subtask definitions.** Initial and goal relations are kept fixed across the corresponding simulation and real-world task definitions.

Subtask	Goal
S1-1	Place glasses on the stand.
S1-2	Put the red-cap marker in the green cup.
S1-3	Put the yellow-cap marker in the basket.
S1-4	Place the mouse on the mouse pad.
S1-5	Put the red pen in the green box.
S1-6	Put the black pen in the pen holder.
S2-1	Place the blue bowl on the plate.
S2-2	Put the carrot in the basket.
S2-3	Put the eggplant in the basket.
S2-4	Put the red chilli in the basket.
S3-1	Place the soda bottle on the shelf.
S3-2	Put the pencil case in the metal box.
S3-3	Place the snack bag on the shelf.
S3-4	Put the candy wrapper in the can.

C.2 Reconstruction and organization examples

Figure C.1 places selected observations beside their reconstructions and illustrates how an initial layout defines an organization goal. The examples complement the actual evaluation environments in Figure 3.

Figure C.2 expands the scene-agent workflow using an example initial and target layout. It separates semantic assessment from deterministic validation and complements the method pipeline in Figure B.1.

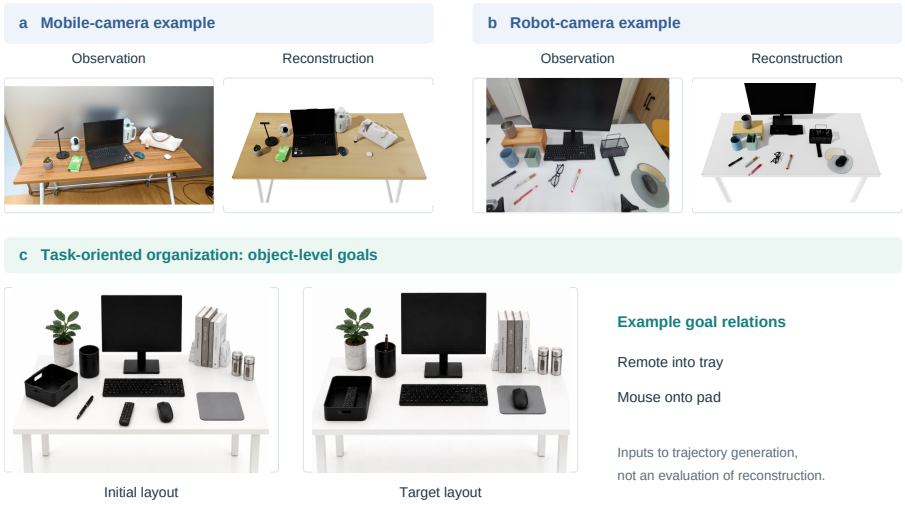


Figure C.1: **Qualitative examples of scene reconstruction and organization. a,b**, Mobile-camera and robot-camera observations paired with reconstructed scenes; views are not pixel-aligned. **c**, Initial and target layouts illustrating object-level manipulation goals.

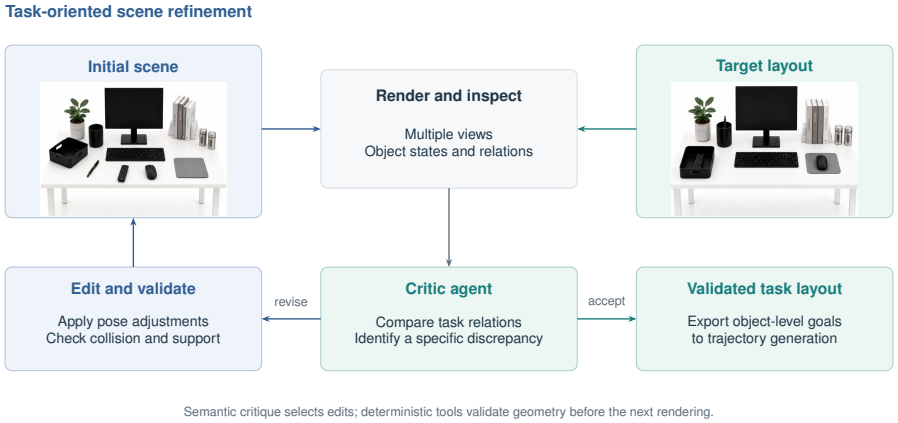


Figure C.2: **Iterative scene organization through critique, editing, and validation.** The critic compares rendered views and object states with target relations to guide pose adjustments. Collision and support checks validate each edit before re-rendering. The accepted layout defines object-level goals for trajectory generation.

Figure C.3 presents additional input–reconstruction pairs.

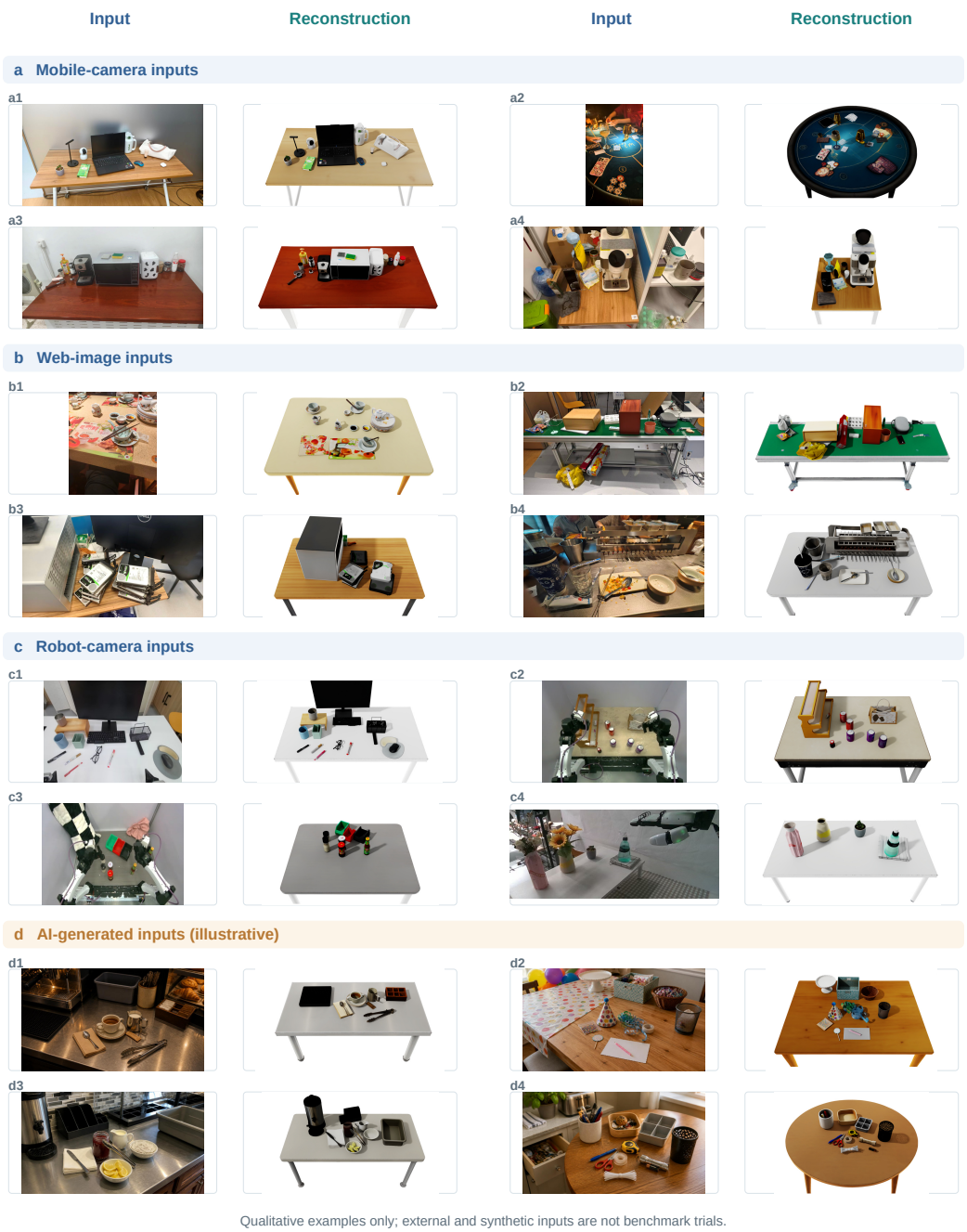


Figure C.3: **Qualitative input–reconstruction pairs. a–d**, Mobile-camera, web, robot-camera, and AI-generated inputs, respectively. Each pair shows the input on the left and reconstruction on the right. Examples extend beyond the evaluation benchmark; views are not aligned in pose or scale.

D TRAINING DATA AND EVALUATION PROTOCOL

D.1 TRAINING DATA AND RETAINED-TRAJECTORY BUDGETS

Every stated data budget is the number of retained training trajectories for *each subtask*. Data from all subtasks of a scene train one scene-specific policy. Table D.1 separates the sources of training experience; Tables D.2 and D.3 give per-subtask and scene-total counts.

Table D.1: **Experience sources and retained-data notation.** The notation separates monitored training rollouts, corrective data and newly generated demonstrations. The reported quotas distinguish newly generated from rollout-derived data, without a separate count for each rollout-derived source. No numerical allocation between base and targeted collection is inferred from those quotas. Policy and correction symbols denote sources within episodes, not separately counted episode datasets.

Data item	Source and learning role	Budget accounting
$\mathcal{D}_0$	Initial validated simulation demonstrations.	50 per subtask at R0.
$\mathcal{D}_k^{\mathrm{mon}}$	Policy-generated trajectories recorded during monitored training collection.	Part of rollout-derived data.
$\mathcal{D}_k^{\mathrm{rec}}$	Corrections generated from eligible intermediate states reached during collection.	Part of rollout-derived data.
$\mathcal{D}_k^{\mathrm{roll}}$	Retained data assembled from monitored and corrective sources.	50 added at R1; 100 at R2.
$\mathcal{D}_k^{\mathrm{new}}$	Newly generated demonstrations after collection configurations are updated.	100 added at each round.
Real HIL data	Real-world training rollouts with operator corrections.	10 per subtask; pooled within scene.
Real test records	Autonomous policy evaluation, independent of HIL training.	Ten trials per subtask and setting; not training data.

R0 uses 50 generated demonstrations per subtask. R1 adds 100 newly generated and 50 rollout-derived trajectories; R2 adds 100 of each, giving cumulative budgets of 200 and 400. The counts describe retained training sets, not total acquisition attempts, transitions or computation.

Training-data selection. Our learning formulation supports joint training on successful and unsuccessful trajectories. In the reported experiments, however, only trajectories that ultimately completed the task were retained for policy training. These include episodes that succeeded after corrective intervention and may therefore contain intermediate policy errors. Unsuccessful monitored rollouts informed failure attribution and targeted collection but were excluded from the policy-training dataset. All reported training budgets count retained successful trajectories.

Table D.2: **Simulation training schedule per subtask.** The two data columns are additions at the indicated stage; the total is cumulative. For R1/R2, newly generated demonstrations are $\mathcal{D}_k^{\mathrm{new}}$ and retained rollout-derived data are $\mathcal{D}_k^{\mathrm{roll}}$. R0 contains the initial generated demonstrations. Counts do not include performance-test episodes.

Stage	Newly generated demonstrations	Retained rollout-derived data	Cumulative total
R0	50	0	50
R1	100	50	200
R2	100	100	400

Table D.3: **Scene-level trajectory budgets.** Counts equal the reported per-subtask budget multiplied by the number of subtasks covered by the scene policy. Real-only 10 and HIL+10 contain the same number of trajectories but are different training settings. The last row is evaluation, not training; each real-world subtask is tested in ten autonomous rollouts.

Dataset / evaluation	**S1 (6 tasks)**	**S2 (4 tasks)**	**S3 (4 tasks)**
R0 simulation training	300	200	200
R1 simulation training	1,200	800	800
R2 simulation training	2,400	1,600	1,600
HIL+10 real training	60	40	40
Real-only 10 training	60	40	40
Real-only 100 training	600	400	400
Real-only 200 training	1,200	800	800
Real test episodes per setting	60	40	40

D.2 Real-world refinement and comparison settings

For each scene, the simulation-evolved policy is refined using 10 retained real-world HIL training trajectories per subtask. An operator corrects undesirable execution, such as slippage or inaccurate placement, and the retained data are annotated for the same conditioned training objective used in simulation. Data are pooled across subtasks to update the shared scene policy. Final real-world test rollouts are collected independently of HIL training.

The Real-only baselines use real-world demonstrations at budgets of 10, 100 and 200 trajectories per subtask, without the task-specific simulation stage. These demonstrations are collected by trained data-collection operators following a standard operating procedure. Static simulation collection continues from the initial demonstration dataset under the original reset distribution; EmbodiRSI instead combines rollout-derived experience with targeted recollection. Static and adaptive comparisons share the 50-trajectory-per-subtask initial dataset and match cumulative retained trajectory counts. The compared policies use the same architecture and training recipe. Table D.4 summarizes these settings. Equal real-world trajectory budgets do not imply identical datasets; Real-only demonstrations and HIL corrections are distinct collection settings.

Table D.4: **Training and collection settings.** Budgets are retained trajectories per subtask. The table distinguishes the adaptation data sources and configuration-sampling procedures. Equal real-data counts do not imply identical trajectories.

Setting	**Real-only**	**Static simulation**	**EmbodiRSI (adaptive)**
Scene policy	One per scene	One per scene	One per scene
Architecture	Same policy architecture across compared groups		
Initial simulation data	Not used	Shared 50 demonstrations per subtask	Same initial dataset as Static
Subsequent reset sampling	Real collection	Original configuration distribution	Feedback-guided configuration update
Simulation trajectory budget	None	200 or 400	200 (R1) or 400 (R2)
Training recipe	Same policy-training recipe across compared groups		
Real training source	Demonstrations at 10/100/200 per subtask	Zero-shot or +10 HIL	Zero-shot or +10 HIL
Role in comparison	Physical-data reference	Static-collection control	Adaptive-collection workflow

D.3 AUTONOMOUS EVALUATION PROTOCOL

Each subtask is evaluated separately with the scene's trained policy. Agent takeover, planner-generated recovery and human correction are disabled; policy-generated reattempts remain part of autonomous execution. Outcomes are scored against the corresponding task definition. Here zero-shot real deployment means no task-specific real-world policy refinement; the simulation itself is constructed from observations of the target workspace.

In simulation, each task–checkpoint pair is tested in repeated rollouts under its evaluation reset configuration. The number of episodes varies across pairs. Success is $r_{s,j} = 100N_{s,j}^{\mathrm{succ}}/N_{s,j}^{\mathrm{test}}$, where $N_{s,j}^{\mathrm{test}}$ is the total number of attempted evaluation episodes. These performance tests are distinct from monitored training collection, denoted by $\mathcal{D}^{\mathrm{mon}}$.

Real-world evaluation uses ten autonomous rollouts per subtask and setting, separate from HIL training collection, giving 140 test episodes for a fully evaluated setting. The empirical task rate is $r_{s,j} = 100N_{s,j}^{\mathrm{succ}}/10$. Subtasks sharing a policy are not independently trained policy replicates.

D.4 STATISTICAL ANALYSIS AND NUMERICAL PRECISION

Scene-level standard deviations describe variation across subtasks sharing a policy, not independent training runs. Task-matched comparisons do not imply episode-wise pairing of resets. For scene s with M_s subtasks and task success percentages $r_{s,j}$, we report

$$\overline{r}_s = \frac{1}{M_s}\sum_{j=1}^{M_s} r_{s,j}, \qquad s_s = \sqrt{\frac{1}{M_s - 1}\sum_{j=1}^{M_s}(r_{s,j} - \overline{r}_s)^2}. \tag{D.1}$$

Here $M_1 = 6$ and $M_2 = M_3 = 4$. SD describes subtask heterogeneity, not confidence intervals or variation across independent training runs. Real-world rates use integer success counts; simulation summaries use the tabulated task-level percentages. For a fixed real-world task and trained policy, Figure E.1 additionally reports pointwise 95% Wilson intervals under an independent binomial model. For $\widehat{p} = N^{\mathrm{succ}}/n$, $n = 10$ and $z_{.975} = 1.959964$, the interval is

$$\frac{\widehat{p} + z_{.975}^2/(2n) \;\pm\; z_{.975}\sqrt{\widehat{p}(1-\widehat{p})/n + z_{.975}^2/(4n^2)}}{1 + z_{.975}^2/n}. \tag{D.2}$$

These conditional intervals quantify finite test-sample precision; they do not account for dependence between episodes, training variability or selection of tasks and checkpoints. They are not used to test method differences.

The main aggregate is the scene-balanced mean $\overline{r} = \frac{1}{3}\sum_{s=1}^{3}\overline{r}_s$, reported without an overall SD to keep it distinct from the within-scene statistics. Pooled real-world success counts are a different weighting: for example, R2+10 has 116/140 successes, whereas its scene-balanced mean is 83.1%. Collection-method differences are formed per subtask before summarization, $d_{s,j} = r_{s,j}^{\mathrm{adaptive}} - r_{s,j}^{\mathrm{static}}$. Their mean and sample SD are reported in percentage points. Equal-scene collection-comparison means use S1 and S3 only.

All calculations retain the supplied input precision and use round-half-up to one decimal only for final displayed percentages, SDs and gains. Simulation collection comparisons use the complete S1/S3 results at both matched budgets; no Static result is extrapolated to S2. In task-level direction counts, increases, ties and decreases refer to observed success rates, not hypothesis tests. The numerical comparisons are descriptive; no significance or equivalence test is reported.

E DETAILED RESULTS AND BEHAVIORAL OBSERVATIONS

Task-level results and qualitative observations are grouped below; experimental and statistical definitions are provided in Appendix D.

E.1 SIMULATION EVOLUTION

Table E.1 reports per-task simulation success across collection-and-training rounds.

Table E.1: **Task-level simulation success (%).** Scene summaries report mean ± sample SD across subtasks (S1: six; S2/S3: four each), not confidence intervals. The final row weights scenes equally and reports no SD. Numbers in column headers denote simulated training trajectories per subtask.

Task	R0 (50)	R1 (200)	R2 (400)
S1: Office Table			
Glasses → stand	44.5	64.9	80.7
Red-cap marker → green cup	34.0	50.1	75.3
Yellow-cap marker → basket	26.1	30.9	57.0
Mouse → mouse pad	33.3	70.1	85.1
Red pen → green box	17.5	36.0	72.0
Black pen → pen holder	14.0	42.2	76.5
Mean ± SD	28.2 ± 11.4	49.0 ± 15.8	74.4 ± 9.7
S2: Kitchen Table			
Blue bowl → plate	75.0	84.0	86.0
Carrot → basket	90.0	94.0	98.6
Eggplant → basket	32.3	74.0	82.5
Red chili → basket	84.0	86.0	96.2
Mean ± SD	70.3 ± 26.1	84.5 ± 8.2	90.8 ± 7.8
S3: Household Table			
Soda bottle → shelf	62.0	76.0	88.0
Pencil case → metal box	72.0	94.0	96.0
Snack bag → shelf	28.6	68.0	75.5
Candy wrapper → can	48.0	76.0	82.0
Mean ± SD	52.7 ± 18.8	78.5 ± 11.0	85.4 ± 8.7
Scene-balanced mean	50.4	70.7	83.5

E.2 REAL-WORLD TRANSFER

Table E.2 gives the exact physical test counts, including task-level ties and lower observed outcomes. Tests are independent of HIL(human-in-the-loop) training and exclude external correction. Scene-balanced and pooled summaries use different weighting, as defined in Appendix D.

Table E.2: **Task-level real-world performance.** Task entries are successes out of ten autonomous tests. Scene summaries report mean success (%) ± sample SD across subtasks; the final row weights scenes equally, without an SD. R1/R2 use 200/400 simulated training trajectories per subtask; +10 adds ten real HIL trajectories per subtask. Real 10/100/200 denote real-only training with 10/100/200 demonstrations per subtask.

Task	R1	R1+10	R2	R2+10	Real 10	Real 100	Real 200
S1: Office Table							
Glasses → stand	7/10	8/10	10/10	10/10	3/10	7/10	7/10
Red-cap marker → green cup	3/10	6/10	3/10	7/10	0/10	4/10	7/10
Yellow-cap marker → basket	6/10	6/10	6/10	9/10	1/10	5/10	7/10
Mouse → mouse pad	6/10	8/10	8/10	9/10	3/10	8/10	8/10
Red pen → green box	2/10	5/10	2/10	7/10	1/10	5/10	7/10
Black pen → pen holder	2/10	5/10	3/10	7/10	0/10	5/10	6/10
Mean ± SD	43.3 ± 22.5	63.3 ± 13.7	53.3 ± 32.0	81.7 ± 13.3	13.3 ± 13.7	56.7 ± 15.1	70.0 ± 6.3
S2: Kitchen Table							
Blue bowl → plate	5/10	9/10	6/10	10/10	5/10	6/10	9/10
Carrot → basket	6/10	8/10	7/10	7/10	5/10	7/10	7/10
Eggplant → basket	6/10	8/10	9/10	8/10	4/10	6/10	8/10
Red chili → basket	7/10	8/10	6/10	9/10	4/10	6/10	7/10
Mean ± SD	60.0 ± 8.2	82.5 ± 5.0	70.0 ± 14.1	85.0 ± 12.9	45.0 ± 5.8	62.5 ± 5.0	77.5 ± 9.6
S3: Household Table							
Soda bottle → shelf	6/10	9/10	7/10	8/10	3/10	5/10	7/10
Pencil case → metal box	6/10	8/10	8/10	9/10	2/10	5/10	7/10
Snack bag → shelf	5/10	8/10	6/10	9/10	2/10	6/10	9/10
Candy wrapper → can	4/10	5/10	5/10	7/10	1/10	5/10	8/10
Mean ± SD	52.5 ± 9.6	75.0 ± 17.3	65.0 ± 12.9	82.5 ± 9.6	20.0 ± 8.2	52.5 ± 5.0	77.5 ± 9.6
Scene-balanced mean	51.9	73.6	62.8	83.1	26.1	57.2	75.0

Figure E.1 shows the task-level comparison between R2+10 and Real-only 200, including ties and the negative difference.

E.3 MATCHED-TRAJECTORY COLLECTION COMPARISONS

Tables E.4 and E.5 report the complete S1/S3 simulation and physical collection comparisons at 200 and 400 simulated trajectories per subtask. In simulation, the scene-balanced adaptive advantage is 14.4 percentage points at 200 trajectories and 20.6 points at 400. These differences are calculated from the task-level percentages in Table E.4, with equal weighting of S1 and S3 and rounding only after aggregation.

Adaptive collection is higher on all six S1 subtasks and three of four S3 subtasks at both budgets. The S3 pencil-case task remains slightly better with Static: 94.8% versus 94.0% at 200 trajectories, and 97.2% versus 96.0% at 400. Larger Static datasets raise both scene means, but do not improve every task; S1 mouse placement decreases from 67.6% to 59.1%. The aggregate trends therefore coexist with task-level exceptions. These are descriptive comparisons rather than tests of statistical significance.

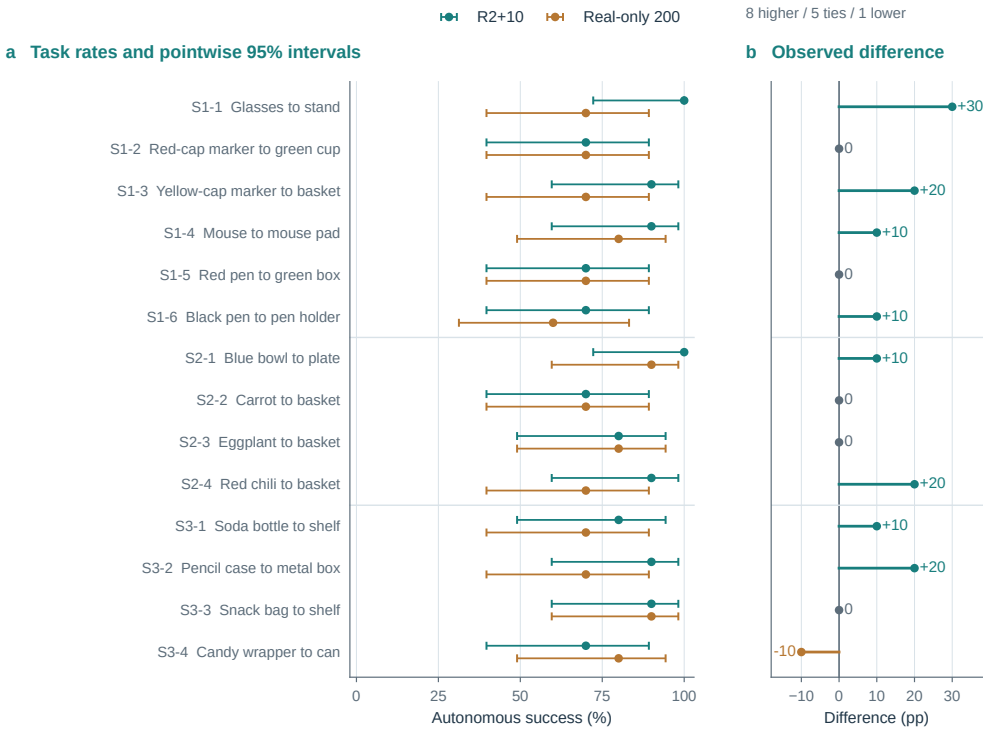


Figure E.1: **Task-level real-world transfer and sampling precision. a**, Success rates for R2+10 and Real-only 200, with pointwise 95% Wilson intervals from ten autonomous tests per task and setting. Intervals condition on each trained policy and assume comparable independent trials; they do not measure training-run variability or simultaneous uncertainty across tasks. **b**, Observed differences matched by subtask identity; positive values favor R2+10. R2+10 uses 400 simulated and ten real-world training trajectories per subtask, versus 200 real-world demonstrations for Real-only 200. Task IDs follow Table C.1.

Table E.3: **Scene-level matched-count comparisons. a**, Simulation evaluation without physical refinement. **b**, Real-world performance before and after HIL refinement. Entries are mean autonomous success (%) ± sample SD across subtasks ($n = 6$ for S1 and $n = 4$ for S3). Δ is the mean ± SD of task-matched differences, in percentage points. Equal counts do not equate total acquisition cost.

Scene	Sim. per subtask	HIL per subtask	Static collection	Adaptive collection	Δ (pp)
a	**Simulation evaluation**				
S1	200	—	40.7 ± 14.6	49.0 ± 15.8	$+8.3 \pm 12.3$
	400	—	51.6 ± 10.2	74.4 ± 9.7	$+22.9 \pm 9.3$
S3	200	—	58.1 ± 28.1	78.5 ± 11.0	$+20.5 \pm 18.0$
	400	—	67.0 ± 23.2	85.4 ± 8.7	$+18.4 \pm 16.0$
b	**Real-world evaluation**				
S1	200	0	33.3 ± 21.6	43.3 ± 22.5	$+10.0 \pm 8.9$
	200	10	48.3 ± 13.3	63.3 ± 13.7	$+15.0 \pm 5.5$
	400	0	45.0 ± 28.8	53.3 ± 32.0	$+8.3 \pm 11.7$
	400	10	68.3 ± 16.0	81.7 ± 13.3	$+13.3 \pm 10.3$
S3	200	0	40.0 ± 20.0	52.5 ± 9.6	$+12.5 \pm 12.6$
	200	10	55.0 ± 17.3	75.0 ± 17.3	$+20.0 \pm 0.0$
	400	0	47.5 ± 12.6	65.0 ± 12.9	$+17.5 \pm 5.0$
	400	10	67.5 ± 9.6	82.5 ± 9.6	$+15.0 \pm 5.8$

Table E.4: **Task-level simulation collection comparison.** R0 is the shared 50-trajectory-per-subtask initialization. Static and adaptive policies have matched cumulative training budgets per subtask. Scene rows report mean ± sample SD across subtasks. The final row weights S1 and S3 equally; it is distinct from the three-scene average in the simulation-evolution table.

	Initial	200 per subtask		400 per subtask	
Task	R0 (50)	Static	R1	Static	R2
S1: Office Table					
Glasses → stand	44.5	32.6	64.9	41.4	80.7
Red-cap marker → green cup	34.0	47.2	50.1	59.7	75.3
Yellow-cap marker → basket	26.1	30.0	30.9	38.6	57.0
Mouse → mouse pad	33.3	67.6	70.1	59.1	85.1
Red pen → green box	17.5	35.3	36.0	48.2	72.0
Black pen → pen holder	14.0	31.5	42.2	62.5	76.5
Mean ± SD	28.2 ± 11.4	40.7 ± 14.6	49.0 ± 15.8	51.6 ± 10.2	74.4 ± 9.7
S3: Household Table					
Soda bottle → shelf	62.0	64.0	76.0	72.0	88.0
Pencil case → metal box	72.0	94.8	94.0	97.2	96.0
Snack bag → shelf	28.6	30.9	68.0	54.2	75.5
Candy wrapper → can	48.0	42.5	76.0	44.4	82.0
Mean ± SD	52.7 ± 18.8	58.1 ± 28.1	78.5 ± 11.0	67.0 ± 23.2	85.4 ± 8.7
Scene-balanced mean	40.4	49.4	63.8	59.3	79.9

Table E.5: **Task-level real-world collection comparison.** Task entries are successes out of ten autonomous test rollouts. All budgets are training trajectories per subtask, and one policy covers each scene. R1 and R2 correspond to simulation budgets of 200 and 400; +10 adds ten real-world HIL training trajectories per subtask for either collection method. Summary rows report mean success (%) ± sample SD across subtasks within each scene.

Task	Static 200	Static 200+10	R1	R1+10	Static 400	Static 400+10	R2	R2+10
S1: Office Table								
Glasses → stand	7/10	7/10	7/10	8/10	7/10	8/10	10/10	10/10
Red-cap marker → green cup	2/10	4/10	3/10	6/10	3/10	6/10	3/10	7/10
Yellow-cap marker → basket	4/10	4/10	6/10	6/10	6/10	8/10	6/10	9/10
Mouse → mouse pad	4/10	6/10	6/10	8/10	8/10	8/10	8/10	9/10
Red pen → green box	1/10	4/10	2/10	5/10	1/10	4/10	2/10	7/10
Black pen → pen holder	2/10	4/10	2/10	5/10	2/10	7/10	3/10	7/10
Mean ± SD	33.3 ± 21.6	48.3 ± 13.3	43.3 ± 22.5	63.3 ± 13.7	45.0 ± 28.8	68.3 ± 16.0	53.3 ± 32.0	81.7 ± 13.3
S3: Household Table								
Soda bottle → shelf	5/10	7/10	6/10	9/10	5/10	6/10	7/10	8/10
Pencil case → metal box	5/10	6/10	6/10	8/10	6/10	8/10	8/10	9/10
Snack bag → shelf	5/10	6/10	5/10	8/10	5/10	7/10	6/10	9/10
Candy wrapper → can	1/10	3/10	4/10	5/10	3/10	6/10	5/10	7/10
Mean ± SD	40.0 ± 20.0	55.0 ± 17.3	52.5 ± 9.6	75.0 ± 17.3	47.5 ± 12.6	67.5 ± 9.6	65.0 ± 12.9	82.5 ± 9.6

E.4 Qualitative execution observations

The observations below supplement the behavioral observations in Section 4 and the selected sequences in Figure E.2. "Reattempt" denotes an action sequence produced by the learned policy after an unsuccessful interaction, not an agent- or planner-generated training correction.

Simulation errors and instruction-following. Missed grasps occurred at peripheral object positions, although these positions were not shown to be unreachable or outside the training distribution. Placement failures also occurred after successful grasping, particularly when objects missed narrow target openings. These execution errors are distinct from task-selection errors. At the 50-demonstration initialization, the policy occasionally confused the red-cap-marker and red-pen instructions in S1. Such confusion was not a recurring failure mode at larger simulation budgets, where most errors arose during execution of the requested manipulation. These observations are specific to the supplied instructions and do not characterize general language competence.

Reattempts at R1 and R2 had mixed outcomes: some completed the task, while others failed. This behavior alone does not establish the presence of an explicit internal failure detector. EmbodiRSI's learning workflow combines experience from imperfect intermediate states with additional demonstrations under difficult configurations; the observations do not isolate the contribution of either source to the resulting behavioral changes.

Physical transfer and local corrective experience. In zero-shot deployment, both Static and evolved policies generally approached the appropriate object despite pose variation. Narrow placement regions made near-contact errors conspicuous, including pens released outside their holders. Broader target regions were more forgiving, but constrained-placement trials did not invariably fail. Evolved policies showed more consistent grasp alignment and sometimes completed an episode after a new attempt. Similar dominant failure types therefore do not imply equivalent success rates across collection settings.

During HIL collection, operators mainly corrected the final approach, acquisition and placement. They perceived takeovers as brief relative to the complete rollout, but no intervention-time fraction was measured. Ten retained trajectories thus do not quantify total operator effort. After refinement,

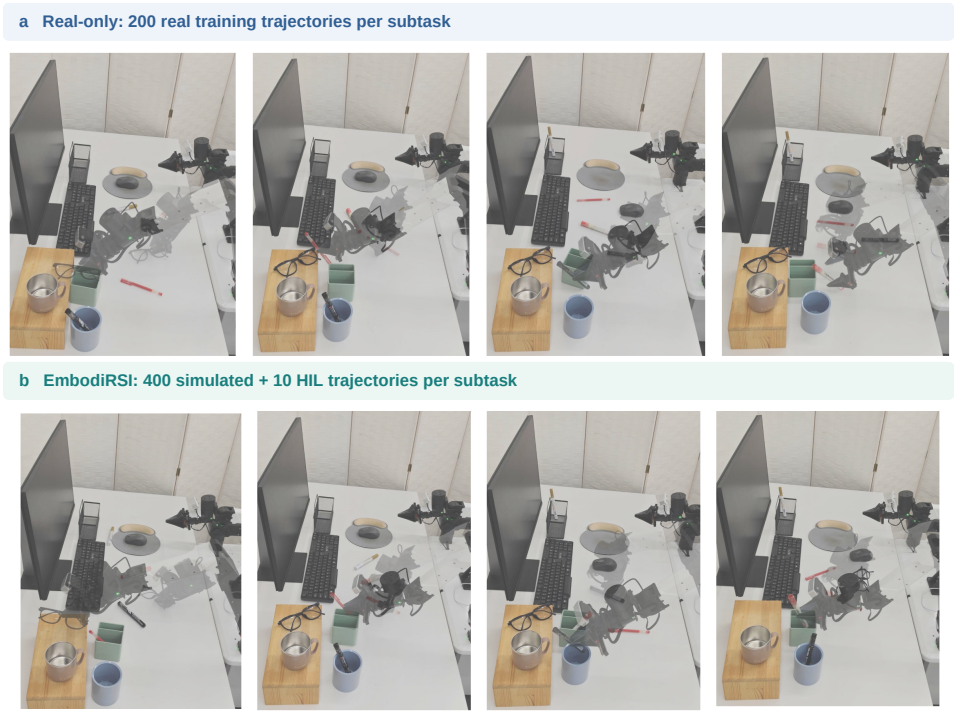


Figure E.2: **Illustrative autonomous real-world executions. a**, Real-only adaptation with 200 real demonstrations per subtask. **b**, EmbodiRSI with 400 simulated and ten real HIL trajectories per subtask. All tests use the learned policy alone. Quantitative results are reported in Table 2.

broad approach motions resembled the simulation demonstrations, while adjustments near grasping and placement resembled the physical corrections. This resemblance does not imply online inverse-kinematics control, nor that HIL changes only contact-related behavior. The pretrained policy's prior capabilities also contribute; not all task knowledge need originate in the simulation data.

Static+10 also benefited from local corrections, but remained less consistent under pose changes and showed fewer reattempts than the refined evolved policies. Equal HIL counts do not imply identical corrective trajectories, and the present comparison does not separate the effects of the different correction types.

Real-only coverage and task selection. With ten real-world demonstrations per subtask, successful behavior was largely restricted to particular configurations; the non-zero test counts are retained in Table E.2. The 200-demonstration collection covered more configurations than the 100-demonstration collection, although both policies remained vulnerable at difficult poses and sometimes selected a task inconsistent with the instruction.

Trained operators followed a standard operating procedure but differed in motion style and preferred reset configurations. Uneven coverage does not make individual demonstrations invalid. It may leave difficult configurations sparsely represented, while correlations among layouts, execution styles and task labels could encourage reliance on incidental cues instead of the instruction. These are possible explanations, not measured causes: coverage, action variability and instruction conditioning were not isolated. The comparison supports examining the relevance of experience to the current policy, rather than a general ranking of human and simulated data quality.

F Additional related work

F.1 Autonomous improvement and corrective experience

Autonomous robot learning has explored how experience generated by a policy can support subsequent policy updates. RoboCat uses robot-generated data to train successive generalist manipulation

models (Bousmalis et al., 2023), while SOAR employs foundation models to guide autonomous experience collection and evaluation for improving instruction-following skills (Zhou et al., 2024). PLD trains residual reinforcement-learning experts, collects recovery trajectories through a hybrid rollout procedure, and distills the resulting data into a VLA model through supervised fine-tuning (Xiao et al., 2026). These approaches provide precedents for learning from policy-dependent experience rather, than relying exclusively on an initial demonstration set. Within this line of work, EmbodiRSI focuses on coupling corrective continuations from policy-failed states with updates to subsequent collection configurations. Its recursion operates through the experience acquired for a fixed policy-learning formulation, rather than through changes to the learning algorithm itself.

F.2 Failure-driven acquisition and adaptive curricula

Policy feedback can guide not only corrective supervision but also the allocation of subsequent training experience. Data-Efficient Multitask DAgger distributes additional expert demonstrations across tasks according to policy performance and estimated data utility (Fu et al., 2025). Prioritized Level Replay selects training levels using estimates of their learning potential (Jiang et al., 2021). More directly related to failure-driven generation, Fail2Progress synthesizes simulation data targeted to observed failures and uses it to refine skill effect models (Huang et al., 2025). RoboPearls uses VLM-based failure analysis to formulate targeted simulation requests for further policy training (Tao et al., 2025). EmbodiRSI studies a configuration-level acquisition mechanism: attributed rollout failures guide the selection of source-object and target-anchor reset regions within each task. The evaluated procedure maintains fixed per-subtask retained-data budgets and mixes targeted configurations with the base collection distribution. Its failure-concentration statistics serve as an acquisition heuristic, not an estimate of configuration-specific learning gains.

F.3 Agentic workflows and execution-time recovery

Agentic systems operate at different levels of the robotics workflow. EmbodiedClaw exposes embodied-AI development activities, including environment construction, trajectory synthesis and evaluation, as conversationally executable skills (Zhou et al., 2026). ABot-Claw combines persistent cross-embodiment memory with coordinated execution and feedback-based correction (Huo et al., 2026), while PhysiAgent integrates monitoring and self-reflection into physical task execution (Wang et al., 2025). At the motion-generation level, ReKep optimizes visually grounded relational keypoint constraints to produce robot actions (Huang et al., 2024). AgentChord augments task graphs with precompiled recovery branches for handling anticipated execution deviations (Xu et al., 2026). These mechanisms are relevant to workflow coordination and corrective execution, but their roles should be distinguished from the training-data role of recovery in EmbodiRSI. Our diagnostic and planning tools generate experience during collection; the updated policy is subsequently evaluated without those external recovery mechanisms. Policy-generated reattempts at test time therefore reflect the learned controller, not an online planner takeover.