RESET: One Demonstration, One Verified World, Self-Improved Skills Across Embodiments

Zheng Sun[1*] Zhuo Li[1*] Zhihao Li[1*] Heng Yi Sim[1]
Ge Wang[2] Ziyue Zhu[3] Mingkai Jia[4] Fei Chen[1]

Project page: zainzh.github.io/RESET

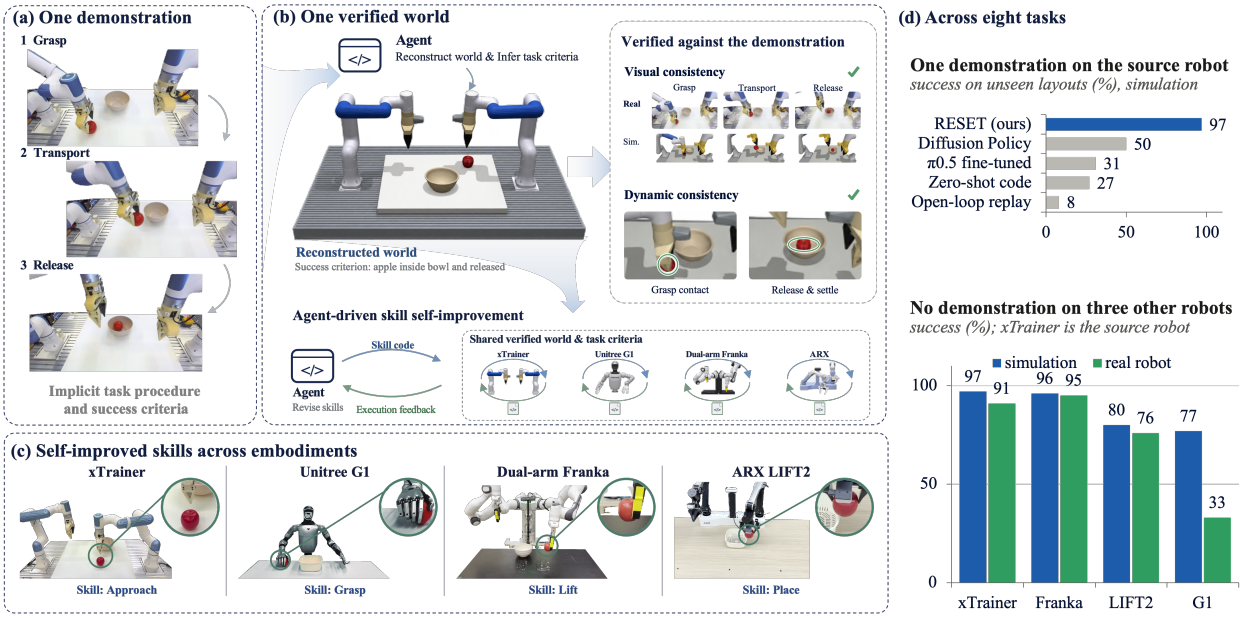


Figure 1. **Overview of RESET.** (a) One real demonstration implicitly specifies the task procedure and terminal success condition. (b) One verified world: RESET reconstructs a contact-rich world and read-only criterion (Rebuild), checks it against the demonstration (Examine), and refines white-box control code from execution feedback (Self-improve). (c) Cross-embodiment transfer: the shared verified world and criterion enable heterogeneous robots to acquire adapted skills without new demonstrations (Transfer) and run them on real robots (Execute). (d) Results across the eight tasks: the source robot against baselines, and all four robots in simulation and on real robots.

Abstract

*Large language model (LLM) coding agents can iteratively self-improve robot manipulation skills in simulation benchmarks, yet real-world tasks provide neither ground-truth success scorers nor resettable practice environments. Our key observation is that a single real demonstration simultaneously specifies scene geometry, defines the terminal success criterion, and serves as a physical reference trajectory for verifying world fidelity—all decoupled from the source robot's kinematics. We present **RESET**: we reconstruct a contact-rich world and derive a frozen read-only criterion from one demonstration (**R**ebuild); check that world against the demonstration by open-loop replay and silhouette consistency (**E**xamine); evolve modular white-box task graphs and skill code from per-node failure evidence (**S**elf-improve); deploy frozen code on real robots with zero runtime LLM calls (**E**xecute); and share the verified world and criterion with unseen robots without new demonstrations (**T**ransfer). On 8 manipulation tasks, each rebuilt from one real demonstration, RESET achieves 97% average success on unseen layouts in simulation, whereas zero-shot code reaches 27%, per-task fine-tuned $\pi_{0.5}$ policies 31%, and Diffusion Policy 50%; injecting reconstruction defects shows that practice tolerates many reconstruction errors, that geometric errors*

*Equal contribution. [1]The Chinese University of Hong Kong. [2]The Chinese University of Hong Kong, Shenzhen. [3]Nankai University. [4]The Hong Kong University of Science and Technology.

can still stall or mislead it, and that the check flags most of them while physical randomization covers the rest. Without any target demonstrations, 3 heterogeneous robots acquire the same task skills in the shared world: a robot with similar arms copies 95% of its skill functions unchanged from the source robot, while robots with different arms and hands write their own. In real-robot experiments, the resulting programs run on 4 robots and succeed in 80% of 280 trials.

1. Introduction

LLM-driven coding agents are emerging as a compelling paradigm for acquiring robot manipulation skills. Whereas hand-crafted or one-shot synthesized control programs [10] struggle to scale across contact-rich tasks and diverse robots, closed-loop recursive self-improvement (RSI) lets an agent inspect failure evidence, revise control code, and retain verified improvements, steadily advancing performance on simulation benchmarks such as RoboDojo [2, 6, 9, 15, 20, 24]. Yet this paradigm relies on a strong hidden premise: human engineers have already built the simulation scene, reset mechanism, and ground-truth success scorer. Physical tasks supply none of these. Recent attempts to run self-improvement directly on real robots [22] rely on manual or heuristic self-checks and custom reset fixtures, requiring hours to tens of hours of physical trial-and-error per task without generalizing across embodiments. Scaling code self-improvement to the real world thus hinges on a foundational question: *without a hand-engineered benchmark, where do a reliable success criterion and a trustworthy practice world come from?*

Reconstructing real scenes into simulation, practicing inside them, and transferring back to the real world (Real-to-Sim-to-Real) offers a natural route [4, 5, 17], but existing pipelines fall short in three ways. First, *reconstructed worlds lack pre-training verification*: prior methods proceed straight from scene reconstruction to policy learning, leaving world fidelity to be discovered only post hoc via real-world failure [12]. Second, *success criteria rely on manual reward design and invite reward hacking*: under multi-round code search, agents readily exploit loose criteria with unintended behaviors [17]. Third, *demonstrations remain entangled with source kinematics*: imitating recorded trajectories binds task knowledge to one robot morphology, invalidating the demonstration whenever degrees of freedom (DoF) or end-effectors change [4, 17].

Our core insight is that **a single successful real-world demonstration simultaneously provides a blueprint of the world, a success criterion, and a physical reference for verifying that world—all independent of the source robot's kinematics.** The initial RGB-D frame captures scene geometry, while the terminal state specifies exact geometric completion relations. Crucially, because the demonstration is a known-successful physical trajectory, an accurate world should reproduce object grasps and motion under open-loop contact replay of the recorded joint commands and align rendered object silhouettes with observed masks. Moreover, by distilling the demonstration into an embodiment-agnostic specification (*world + frozen criterion + local contact priors*) rather than imitating its trajectory, an agent can write modular control code on its own. This both resolves the scalability bottleneck of manual code-skill authoring and exploits the white-box nature of code skills—explicit geometric guards, stage-level verification, and deterministic execution—for long-horizon stability.

Guided by this insight, we propose **RESET** (**R**ebuild, **E**xamine, **S**elf-improve, **E**xecute, **T**ransfer; Fig. 1). The name reflects what the method restores to a real task: a world that can be reset and practiced in at will. Rebuild reconstructs an interactive MuJoCo scene from one RGB-D demonstration and extracts a read-only terminal scorer. Examine checks the rebuilt world against the demonstration by dynamic replay and silhouette alignment before practice, leaving unidentifiable mass and friction to randomized practice. Self-improve evolves an object-centric task graph and white-box skill library from per-node failure evidence under diagnostic/hold-out gating. Execute runs frozen code online with zero LLM calls, low cost, and node-level debuggability. Transfer hands the verified world and criterion unchanged to target robots that adapt only a thin hardware layer.

We evaluate RESET across 4 diverse embodiments and 8 real manipulation tasks. From one demonstration on the source robot, RESET achieves 97% success on unseen layouts in simulation, where zero-shot code and per-task fine-tuned $\pi_{0.5}$ policies stay near 30% and Diffusion Policy reaches 50%. Controlled defect injection shows that practice tolerates many reconstruction errors. A geometric error, however, can make a task unsolvable or hide a failure during practice, and it changes what a simulated success rate means; Examine flags most such errors, and physical randomization covers mass and friction. Without any target demonstration, Dual-arm Franka, ARX LIFT2, and Unitree G1 acquire their own skills in the shared world. The final programs succeed in 80% of 280 real-robot trials on 4 physical platforms.

Our main contributions are threefold:

- **A framework from one demonstration to skills across embodiments.** RESET turns a single real demonstration into frozen, deployable code skills, both for the demonstrated robot and for robots that never received one.
- **The demonstration replaces the benchmark.** One demonstration supplies what self-improvement otherwise takes from a hand-built benchmark: a practice world, a success criterion, and a reference for checking that world. Agent self-improvement can therefore run on real tasks without hand-built scenes, rewards, or resets.

- **Evidence on eight real tasks and four embodiments.** RESET reaches 97% success on unseen layouts in simulation where zero-shot code reaches 27%; three target robots learn the tasks with no new demonstration, by practicing in the shared world rather than by copying the source robot's code; practice tolerates many reconstruction errors; and frozen skills run on 4 physical platforms (80% success over 280 real-robot trials).

2. Related Work

Code-skill self-improvement in simulation benchmarks. Code as Policies [10] and Voyager [18] pioneered synthesizing control programs and skill libraries with LLMs. Recent closed-loop RSI frameworks—including RATs [24], the concurrent RPG [20], EmbodiedEvo [9], PhysEvo [15], Embodied Turing Machines [6], PyRUA-Lean [13], and ReSkill [14]—iteratively refine task graphs and skill code from execution feedback on benchmarks such as RoboDojo [2]. However, all assume human-engineered simulation scenes, resets, and ground-truth scorers. RESET removes this benchmark dependency by running self-improvement inside a world reconstructed from and verified against a single real demonstration, while also enabling transfer across robot embodiments.

Self-improvement on physical robots. To bypass simulation benchmarks, ENPIRE [22] explores and revises policies directly on real robots. While demonstrating real-world RSI feasibility, physical exploration demands hours of robot wear, custom reset mechanisms, and robot-specific verification logic. RESET confines trial-and-error to a verified simulation world, reserving the real robot for executing the final code and reusing the world across embodiments.

Real-to-Sim-to-Real and scene verification. RialTo [17], Re3Sim [5], and RoboSplat [23] reconstruct scenes to train neural policies or synthesize data; X-Sim [4] derives cross-embodiment dense rewards from simulated object motion; EmbodiRSI [8] and D-World [21] update digital twins from single images or online rollouts. Yet prior pipelines treat reconstruction as an open-loop preprocessing step without checking the rebuilt world before training [12]. RESET uses the demonstration itself as a dynamic and visual verification standard, characterizes how reconstruction errors affect code policies, and outputs inspectable white-box programs.

Learning from a single demonstration. Demo2Code [19] summarizes demonstrations into code specifications, RoboSplat [23] augments single demonstrations via novel-view synthesis, and Human2Sim2Robot [11] bridges human-robot morphology gaps via RL from one video. Unlike methods that imitate demonstrated trajectories, RESET uses the demonstration solely to build and verify the environment and terminal criterion; runtime controllers never replay the demonstration, unlocking zero-demonstration transfer.

3. Method

3.1. Motivation

Three questions shape the design of RESET. Each starts from an observation about what happens without the corresponding component (Sec. 4).

Why does code need practice, and where can it practice? Code written once is not enough. Given the same interfaces and contact hints that RESET uses, an agent's first program succeeds on 27% of unseen simulated layouts on average, varies widely between samples (70%, 0%, and 36% on one task), and never succeeds on the four hardest tasks (Tab. 1). Such a program improves only by being run, failing, and being revised, which takes an environment that resets at will and a score that needs no human. A real task offers neither, and practicing on a real robot costs hours of supervised trials per task [22]. One successful demonstration, however, already contains both: its first frame shows the scene and its final state defines success, which motivates *Rebuild*.

Why check the rebuilt world against the demonstration? No reconstruction is perfect, and a success rate measured in a wrong world says little about the real one. When object scale is off by 10% or the table is 2 cm too low, a skill that is perfect in the correct world falls to 0% on several tasks (Tab. 2); and under a loosely worded criterion, a program that knocks the carton flat is scored 100%. The demonstration gives a check for both. It is a physical execution known to succeed, so a usable world must reproduce it; and its final state, not a text description, defines success. What one demonstration cannot determine, mass and friction, we do not claim to verify and instead randomize during practice. This motivates *Examine*.

Why code, and why share the task rather than the motion? The demonstrated motion itself transfers poorly: replayed open-loop it succeeds on 8% of new layouts, and a vision-language-action policy fine-tuned on 100 trajectories per task reaches 31% and a Diffusion Policy trained on the same trajectories 50%. A program differs in three ways. It exposes which step failed, which gives an agent something specific to revise (Sec. 4.4). It runs on the real robot with no model in the loop. And it keeps what the task requires, the world and the criterion, apart from how one robot meets that requirement, so a robot with different arms or hands can practice in the same world (Sec. 4.5). This motivates *Self-improve*, *Execute*, and *Transfer*.

3.2. Problem Formulation and Overview

Let $\mathcal{D} = \{(I_t, D_t, q_t, g_t)\}_{t=0}^{T}$ be one successful real demonstration on a source robot $\mathcal{E}_s$, with RGB image I_t, depth map D_t, joint angles q_t, and gripper command g_t. The scene has observable geometry θ_{obs} (object scale, table height, camera pose), which $\mathcal{D}$ constrains, and latent physics θ_{lat} (mass,

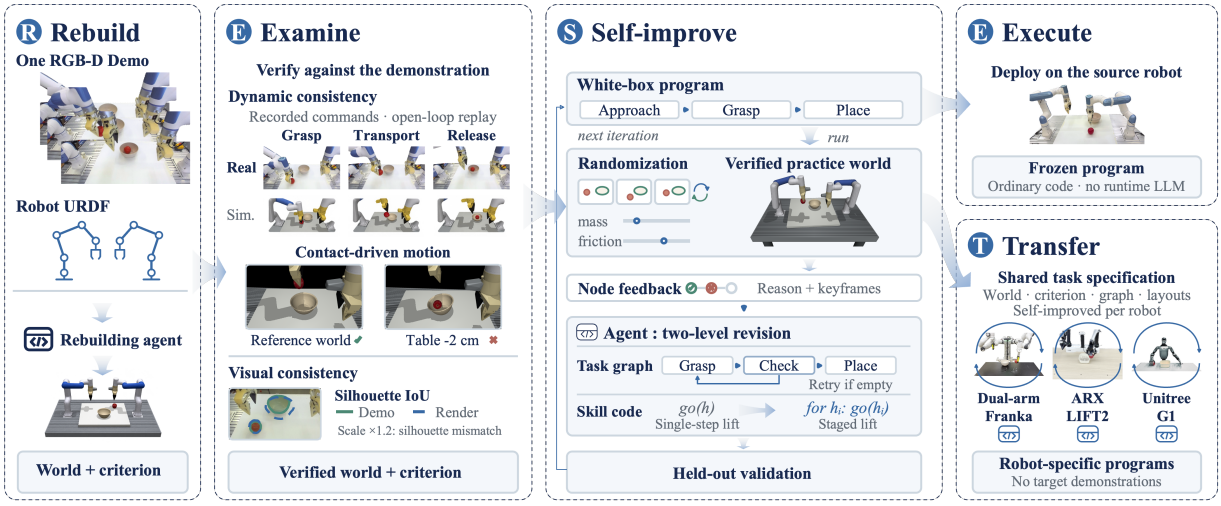


Figure 2. **The five steps of RESET, illustrated on the *Apple* task.** Rebuild turns one RGB-D demonstration and the robot's URDF into an interactive world and a success criterion. Examine checks that world against the demonstration by open-loop replay and by rendered silhouettes. Self-improve revises a white-box program from node-level failures and keeps a revision only if it does better on held-out layouts. Execute runs the final program on the real robot with no LLM call. Transfer gives other robots the same world and criterion to practice in, without a demonstration of their own.

friction), which one successful execution cannot identify. Given target robots $\{\mathcal{E}_k\}$ with no demonstration, we synthesize control programs P_s and $\{P_k\}$ that complete the task from unseen object layouts, without hand-written rewards. RESET does so in five stages (Fig. 2): Rebuild, Examine, Self-improve, Execute, and Transfer.

3.3. Rebuild: World and Read-Only Criterion

Contact-rich world reconstruction. We recover the camera pose by aligning the robot's kinematics with the images across the demonstration, and segment the task objects in the first frame (I_0, D_0) with a pretrained promptable segmentation model. A multimodal coding agent iteratively models each object as a mesh, keeping the cavities and articulated parts that stacking, insertion, and drawer pulling depend on. The meshes are registered and scaled to the first-frame depth point cloud, the table plane is fitted, and the scene is exported to a physics simulator together with the robot's URDF. θ_{lat} receives nominal values.

Terminal criterion and read-only scorer. The success criterion is read from the demonstrated final state as a conjunction of geometric relations (containment, support, insertion, hanging, upright orientation) plus a stillness condition after release. The criterion is confirmed once by a person, after the world has been rebuilt and has passed Examine. The scorer is then read-only and hidden from the controller, so practice cannot game it.

3.4. Examine: Dual-Channel World Verification

Sensor noise, camera error, and scale error make every rebuilt world deviate from the real scene. Because $\mathcal{D}$ is known to succeed, Examine checks whether the rebuilt world can reproduce it, through two complementary channels. *The dynamic channel* places objects at their first-frame poses and replays the recorded joint and gripper commands open-loop; objects are never attached to the gripper, so grasping and placement must arise from contact. It measures how many of the demonstrated grasps are reproduced and how far the objects deviate from their tracked motion, which constrains table height and contact support. *The visual channel* renders the world at the recorded joint angles before contact and measures the silhouette IoU of each task object against the demonstration masks, which constrains camera pose and object scale. These measurements are used to compare reconstructions of the same scene and to discard those that reproduce the demonstration clearly worse; the margins are fixed on development tasks before held-out evaluation. Examine is a check that the world is usable, not a guarantee of fidelity: mass and friction leave both channels almost unchanged and are covered by randomized practice.

3.5. Self-Improve: Two-Level Code Evolution

The verified world, randomized collision-free layouts, and the read-only scorer form a task-specific benchmark that resets automatically. In it the controller is a white-box program $P = (G, \Pi, \Lambda)$: an object-centric task graph G whose nodes call skills with entry and exit checks and retry budgets, a skill library Π (pose estimation, inverse kinematics, and grasp, place, sweep, and handover primitives), and the bindings Λ between them. P may read only what a physical robot provides: camera streams, calibration, joint encoders, and the gripper-to-object contact offsets observed in the demon-

Table 1. **Main blind-test success rates (%) on source embodiment xTrainer across 50 unseen layouts per task (TEST 0–49), in simulation.** Zero-shot Code reports the mean and best of 3 independent samples. $\pi_{0.5}$ and Diffusion Policy each train one dedicated model per task on the same 100 successful RESET rollouts ($\pi_{0.5}$ is fine-tuned; Diffusion Policy is trained from scratch).

Method	Apple	Cubes	Stack Cups	Build Tower	Mug-Box	Open Drawer	Sweep	Lift-Box	Mean
Open-loop Demo Replay	4%	0%	12%	0%	38%	0%	0%	10%	8%
Zero-shot Code (3-run mean)	85%	35%	55%	39%	0%	0%	0%	0%	27%
Zero-shot Code (3-run best)	**100%**	70%	**96%**	76%	0%	0%	0%	0%	43%
$\pi_{0.5}$ fine-tuned (per-task) [7]	4%	0%	76%	6%	72%	4%	32%	52%	31%
Diffusion Policy (per-task)	20%	0%	94%	6%	94%	26%	70%	88%	50%
RESET (Ours)	**100%**	**98%**	**96%**	**100%**	**100%**	**100%**	**88%**	**94%**	**97%**

stration. It never reads simulator state or the scorer, so the same program runs unchanged on the real robot.

In each round, a coding agent runs P on diagnostic layouts, reads the status and failure reason of every node together with keyframes, locates the earliest node that went wrong, and revises the program at two levels. Task-level revisions change the graph (e.g., a retry after an empty grasp, a handover between arms); skill-level revisions change the geometry inside a skill (e.g., where to grasp). To avoid fitting to particular failures, a revised program replaces the current one only if it succeeds more often on a separate set of hold-out layouts. When practice converges or the round budget is spent, P is fixed and evaluated once on sealed layouts.

3.6. Execute and Transfer

The final program runs on the physical robot as ordinary code, with no LLM call and no human intervention. Before each node it estimates object poses from segmentation and depth, then executes the skill. All reasoning happens offline, and because the program is frozen after practice, the code that runs on the real robot is exactly the code evaluated in simulation; a failure can be traced to a specific node.

A target robot $\mathcal{E}_k$ receives everything that is independent of the robot unchanged: the verified world, the criterion, the layout distribution, and the source robot's task graph with its notes on physical pitfalls. It supplies its own URDF, camera calibration, and contact offsets for its end-effector, and runs Self-improve in the shared world to obtain P_k. No trajectory or skill code of the source robot is required: a target robot reuses a source function when it suits its own arms or hands and otherwise writes its own (Sec. 4.5). The demonstration is thus used once, to build and check the world, and never as a motion to imitate.

4. Experiments

We evaluate five core questions—Q1: layout generalization vs. baselines; Q2: tolerance to reconstruction errors; Q3: Self-improve component ablations; Q4: zero-demonstration cross-embodiment transfer; and Q5: physical deployment and episode-level Sim-to-Real agreement.

4.1. Experimental Setup

Embodiments and tasks. We evaluate 4 platforms with different kinematics, cameras, and end-effectors (Fig. 1c). The dual-arm 6-DoF xTrainer (parallel grippers) is the only *source embodiment* and provides one demonstration per task. The three *target embodiments* receive none: ARX LIFT2 (6-DoF arms on a lift), Unitree G1 (five-finger hands; lower body fixed in simulation), and a 7-DoF Dual-arm Franka. We test 8 tabletop tasks (Fig. 3): *Apple*, *Cubes*, *Stack Cups*, *Build Tower*, *Mug-Box* (bimanual handover), *Open Drawer* (articulated pull + place), *Sweep* (tool sweeping), and *Lift-Box* (upright orientation).

Implementation. Rebuild uses GPT-6 Astra (object assets, in Blender), SAM3 [1] (segmentation), and MuJoCo [16]; Self-improve uses Claude Code with Claude Opus 5.5. A person confirms the success criterion once per task; this changed the criterion only for *Lift-Box*.

Protocol and sandbox isolation. Q1–Q4 are evaluated in simulation, in the rebuilt worlds. Layouts use disjoint seeds: Diagnostic (DEV 0–19), Hold-out Adoption (DEV 20–39), and sealed Blind Test (TEST 0–49, evaluated once, after development ends, under the final read-only scorer). Each layout moves every task object by up to $\pm$3–6 cm along each table axis and rotates it by up to $\pm 10°$–$180°$ about the vertical, around its demonstrated pose; layouts with collisions or objects out of reach are discarded. Ablations and reruns use isolated agents that see only the initial code, the task text, and their own logs.

4.2. Main Results on Source Embodiment (Q1)

Baselines and quantitative comparison. Table 1 compares RESET against Open-loop Demo Replay, Zero-shot Code (3 independent samples per task with the same interfaces and contact priors), and per-task $\pi_{0.5}$ policies [7] (fine-tuned for 6,000 steps on 100 RESET trajectories from disjoint layouts). Open-loop replay averages 8%: the layouts cannot be solved by memorizing a trajectory. Zero-shot code averages 26.8% with extreme variance (*Cubes*: 70/0/36%; *Stack Cups*: 96/18/50%) and scores 0% across all 3 samples on *Mug-Box*, *Open Drawer*, *Sweep*, and *Lift-Box*. Per-task $\pi_{0.5}$ averages

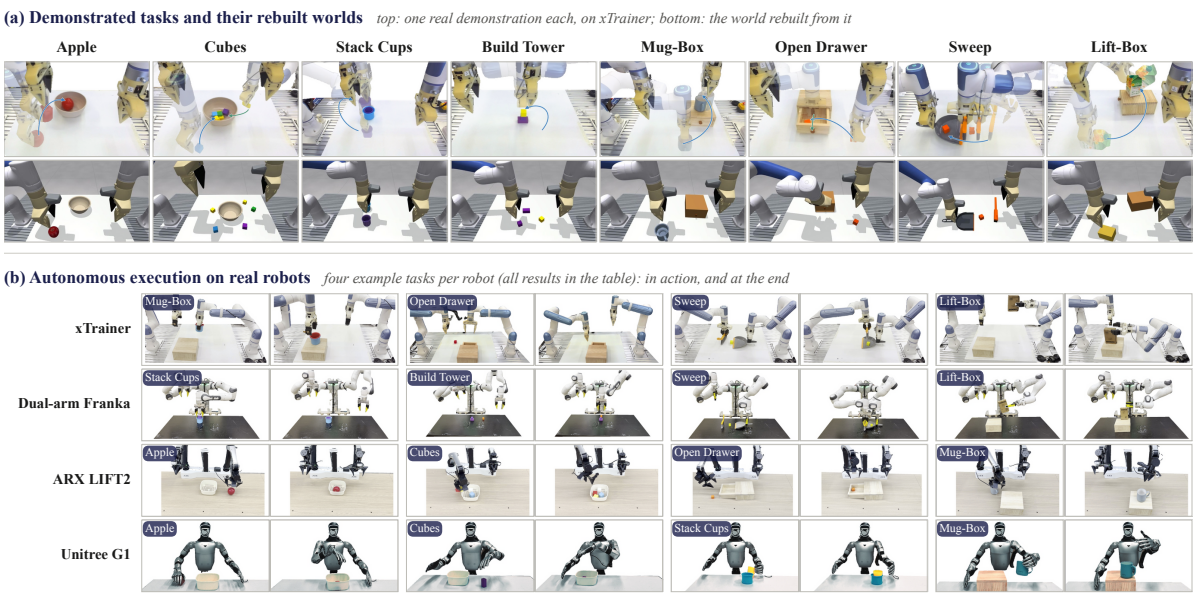


Figure 3. **Demonstrated tasks, their rebuilt worlds, and autonomous real-robot execution.** (a) Top: the eight tasks, each demonstrated once on the source robot xTrainer. Bottom: the world rebuilt from each demonstration, rendered from the demonstration camera. (b) The final programs running on four robots without human intervention. For each robot we show four of the tasks it ran (all results are in Tab. 5), each in action and at the end. Franka, LIFT2, and G1 received no demonstration.

30.8%, and a Diffusion Policy [3] trained from scratch on the same trajectories 49.8%: it approaches RESET on *Stack Cups*, *Mug-Box*, and *Lift-Box*, but stays at or below 26% on *Apple*, *Cubes*, *Build Tower*, and *Open Drawer*, which need precise grasps or placements. Both policies use their default settings; 100 trajectories are little data for fine-tuning a model as large as $\pi_{0.5}$, which may explain its lower score. RESET reaches 97.0% overall and 88%–100% on the four tasks where zero-shot code scores 0%.

Repeatability and criterion precision. Independent reruns from zero-shot starts reach 100% on *Build Tower*, 98% on *Open Drawer*, and 86% on *Sweep*. On *Lift-Box*, a loose text-translated criterion ("carton on box") scores a zero-shot program 100% for knocking the carton flat; the demonstration-derived upright criterion scores it 0%.

4.3. How Much Reconstruction Error Can RESET Tolerate? (Q2)

No reconstruction is perfect, so we inject one defect at a time into the rebuilt worlds: geometric errors (object scale $\times 0.8$–$\times 1.2$, a table 2 cm too low, a shifted camera) and latent physical errors (friction $\times 0.3$, mass $\times 3$). We ask whether practice in such a world still yields a skill that works in the correct one, whether a success rate measured there can be trusted, and whether Examine flags the world.

Finding 1: Practice tolerates most reconstruction errors; a wrong world hurts when it hides a failure. We run the full Self-improve inside defective worlds and test the result in the correct world (Tab. 3). In most cases the outcome is the same as after practice in the correct world: *Mug-Box* reaches 100% from all three defective worlds, *Build Tower* 98%–100%, *Apple* 94% with the table 2 cm too low, and *Stack Cups* 82%–100% in four of six runs with enlarged objects. The revisions an agent makes re-anchor the skill on what it perceives (e.g., a grasp height relative to the perceived rim instead of the demonstrated absolute height), and such revisions carry over. Two cases fail. A geometric error can make the task unsolvable in the practice world, so the program stays where it started (*Apple* with enlarged objects). More subtly, a geometric error can hide a failure: with the table 2 cm too low, the zero-shot *Stack Cups* program happens to release at the right height and succeeds in 98% of practice episodes, so practice finds nothing to fix. In three independent runs the resulting program reaches only 18%–20% in

Table 2. **Wrong worlds: how often they break a skill, and how often Examine flags them.** A skill learned in the correct world is run in worlds with one injected defect; entries count task–defect cases over six tasks. Breaks: success of at most 50%. Last column: breaking cases that Examine flags.

Injected defect	Skill breaks	Examine flags	Breaking, flagged
None (correct world)	0/6	0/6	–
Scale $\times 0.8$, $\times 0.9$	1/12	11/12	1/1
Scale $\times 1.1$–$\times 1.2$	12/18	11/18	9/12
Table -2 cm	3/6	6/6	3/3
Camera shift	0/6	6/6	–
Friction $\times 0.3$	0/6	0/6	–
Mass $\times 3$	2/6	0/6	0/2

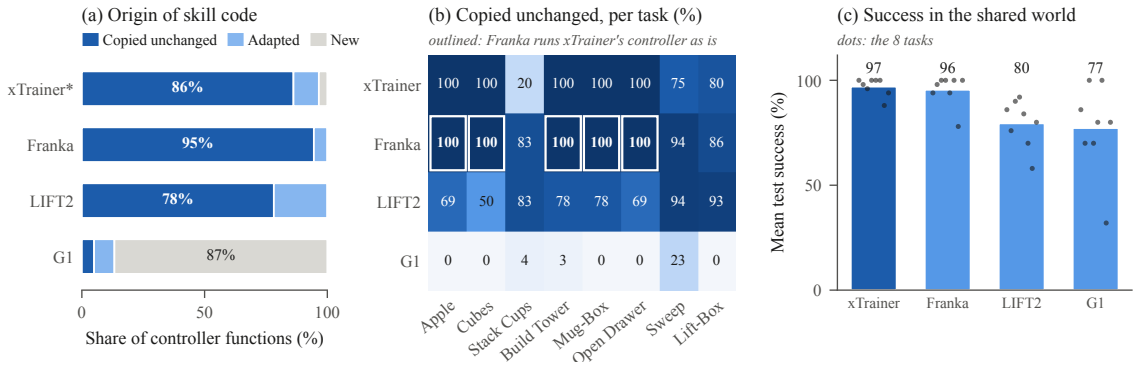


Figure 4. **Cross-embodiment skill reuse (AST analysis) and task performance.** (a) Proportion of skills reused unchanged from another embodiment, adapted, or written from scratch. (b) Per-task unchanged skill reuse, highlighting 5 tasks where Dual-arm Franka passes directly with xTrainer's controller. (c) Blind-test success distributions across embodiments. *For the source robot, the share of its functions that also appear unchanged in another robot's code.

Table 3. **Practicing in a defective world, then testing in the correct one.** Success in the correct world after Self-improve was run in a world with the given defect. Start: the zero-shot program. Several values: independent runs. *Stack Cups* at $\times 1.1$ and $\times 1.2$: criterion scaled with the world.

		Practice world			
Task	Start	Correct	Scale $\times 1.1$	Scale $\times 1.2$	Table -2 cm
Apple	78%	100%	78%	78%	94%
Build Tower	42%	100%	100%	98%	100%
Mug-Box	0%	100%	100%	100%	100%
Stack Cups	18%	96%	18/98/98%	8/82/100%	18/20/18%

the correct world, against 96% after practice there.

Finding 2: A success rate measured in a wrong world cannot be trusted. The opposite direction is not tolerant. A skill that is perfect in the correct world breaks in 12 of 18 worlds with objects enlarged by 10%–20% and in 3 of 6 with the table 2 cm lower, whereas camera shifts and friction changes are absorbed by online localization. A geometric error therefore changes what a simulated success rate means, even when it does not prevent learning, which is why we check the world against the demonstration and verify that it predicts real-robot outcomes (Sec. 4.6).

Finding 3: Examine flags most geometric errors; randomization covers physics. Examine compares versions of the same scene and flags those that reproduce the demonstration clearly worse than the best one. It flags no correct world and 13 of the 16 geometric defects that break a skill, including every lowered table (Tab. 2), and every world in Tab. 3 where practice always failed. It misses three enlarged worlds, so it is a check, not a guarantee. Mass and friction are invisible to Examine and are covered by practicing across 4 worlds with randomized mass and friction: on *Mug-Box* this lifts success under an unseen mass $\times 3$ from 10% to 100% (Tab. 4); the $3\times$ heavy *Open Drawer* stays unsolved.

Table 4. **Practicing across randomized physics.** Success (%) in the correct world and under unseen physical defects, on the only two tasks that a physical defect breaks (Tab. 2).

Task	Practice	Correct	Mass $\times 3$	Fric. $\times 0.3$	Fric. $\times 3$
Mug-Box	Nominal only	100%	10%	100%	100%
	Randomized	100%	**100%**	100%	100%
Open Drawer	Nominal only	98%	0%	100%	95–100%
	Randomized	98%	0%	100%	100%

4.4. Self-Improvement Ablations (Q3)

On *Mug-Box* and *Open Drawer*, starting from zero-shot programs that score 0% (Tab. 6), removing node-level failure logs and keyframes (outcome-only feedback) leaves performance at 12% and 0% after 8 rounds, whereas full RESET converges in 3–5 rounds to 100% and 98%: node-level failure evidence is essential. Removing hold-out gating, the rule that adopts a revision only if it does better on the hold-out layouts, still solves *Mug-Box* (100%) but stays at 0% on *Open Drawer* after all 8 rounds.

4.5. Cross-Embodiment Transfer (Q4)

Transfer performance across morphologies. We hand xTrainer's 8 verified worlds and criteria to three target robots (Tab. 5). Dual-arm Franka and ARX LIFT2 reach 95.5% and 79.5%, and Unitree G1, with five-finger hands instead of parallel jaws, reaches 77.3% (32% on *Sweep*, where its fingers lack lateral stiffness on the flat brush handle).

Is it the shared world, or just copied code? To separate the two, we run the source robot's final programs unchanged on each target robot, with no practice. On Dual-arm Franka, whose arms are close to xTrainer's, five programs work as they are (94%–100%); the other three reach 10%, 24%, and 0% (*Stack Cups*, *Sweep*, *Lift-Box*) and rise to 100%, 78%, and 100% after practice in the shared world. On ARX LIFT2

Table 5. **Simulation and real-robot results across embodiments.** Sim.: success rate (%) over 50 unseen test layouts per task (TEST 0–49). Real: successes/trials of the same frozen program on the real robot, without code edits or human intervention (–: not evaluated). The three target robots received no demonstration.

Robot		Apple	Cubes	Stack Cups	Build Tower	Mug-Box	Open Drawer	Sweep	Lift-Box	Mean
xTrainer (source)	Sim.	100%	98%	96%	100%	100%	100%	88%	94%	97%
	Real	10/10	10/10	9/10	8/10	10/10	10/10	6/10	10/10	73/80
Dual-arm Franka	Sim.	94%	98%	100%	100%	94%	100%	78%	100%	96%
	Real	10/10	10/10	10/10	10/10	10/10	8/10	8/10	10/10	76/80
ARX LIFT2	Sim.	86%	76%	90%	92%	84%	70%	58%	80%	80%
	Real	7/10	8/10	9/10	7/10	9/10	9/10	6/10	6/10	61/80
Unitree G1	Sim.	86%	70%	100%	70%	80%	100%	32%	80%	77%
	Real	5/10	2/10	3/10	–	3/10	–	–	–	13/40

Table 6. **Ablations of Self-improve**, each started from the same zero-shot program that scores 0. Success: test success (%). Rounds: self-improvement rounds used, out of a budget of 8.

	Mug-Box		Open Drawer	
Variant	Success	Rounds	Success	Rounds
Full RESET	**100%**	3	**98%**	5
w/o hold-out gating	**100%**	3	0%	8
Outcome-only feedback	12%	8	0%	8

the source programs succeed on no task, and on Unitree G1 only on *Apple* (18%); after practice the two robots average 80% and 77%. The source robot's task graph and notes are not required either: given only the shared world and criterion and starting from scratch, LIFT2 reaches 94% and 84% on *Mug-Box* and *Open Drawer* and G1 reaches 74% and 92%, comparable to the 84%, 70%, 80%, and 100% obtained when they are inherited.

Skill-level reuse analysis. Comparing function-level syntax trees across the final skill libraries (Fig. 4), 94.5% of Dual-arm Franka's skill functions are identical to another robot's and LIFT2 reuses 78.2%, whereas Unitree G1 writes 86.7% of its skills from scratch yet still reaches 77% on average. What carries over across embodiments is the verified world and criterion, not the code.

4.6. Real-World Deployment and Sim-to-Real Agreement (Q5)

We deploy the frozen programs on four physical robots with no code edits and no human intervention (10 trials per task, objects placed by hand at varied positions). The source xTrainer and the zero-demonstration Dual-arm Franka and ARX LIFT2 run all 8 tasks, and Unitree G1 runs 4 (Tab. 5, Fig. 3). In total, 223 of 280 real-robot trials succeed (80%), and 210 of 240 (88%) on the three fixed-base robots. xTrainer succeeds in 73/80 trials (91%) and Dual-arm Franka in 76/80 (95%), close to their simulation rates (97% and 96%); both arms have a large workspace and localize objects accurately in our scenes. ARX LIFT2 reaches 61/80 (76%, against 80% in simulation): its arms have a smaller reachable workspace, so some layouts place objects near the limit of its reach. Unitree G1 succeeds in 13/40 trials on its four tasks (33%, against 84% in simulation on the same tasks). Its whole-body controller shifts the base whenever the arms move, and the open-loop program does not compensate for this shift (Sec. 5).

Does the verified world predict real outcomes? For every real trial we register the first camera frame to recover the actual object layout, reproduce that layout in the verified world, and run the same frozen program there. Simulation and the real robot give the same outcome in 95% of the 80 trials on xTrainer and in 90% on ARX LIFT2, a robot that took no part in building or examining the world. **Cost.** Self-improve converges in 2–8 rounds, using 100–320 simulation episodes and about 30M LLM tokens per task and robot. Deployment uses no LLM tokens.

5. Limitations and Conclusion

Limitations. RESET currently targets rigid, articulated, and quasi-static sweeping tasks, not deformable cloth or dynamic throwing. Extreme latent parameter jumps that demand a wholly different contact topology (e.g., a 3× heavy drawer) remain beyond parameter randomization. Finally, practice does not yet model the effect of a humanoid's whole-body controller: on Unitree G1 the controller shifts the base whenever the arms move, so real-robot success falls well below that of fixed-base arms (Tab. 5); adapting RESET to real humanoids is left to future work.

Conclusion. We presented RESET, which brings agent-driven code self-improvement from hand-built benchmarks to real tasks. One real demonstration provides the practice world, the success criterion, and the reference for checking that world, so what a task requires is separated from how a particular robot performs it. The rebuilt world need not be perfect: practice tolerates most reconstruction errors, and

checking the world against the demonstration flags those that would mislead it. More broadly, RESET offers a one-shot, cross-embodiment route to robot skills that is carried by an agent's general knowledge of the world and of code: a user demonstrates a task once, and different robots obtain inspectable, ready-to-run skills.

References

[1] 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*, pages 138846–138923, 2026. 5

[2] Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026. 2, 3

[3] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44 (10-11):1684–1704, 2025. 6

[4] 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. 2, 3

[5] 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. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*, pages 20781–20788. IEEE, 2026. 2, 3

[6] Kairui Hu, Siyuan Hu, Fangzhou Hong, Zhaoxi Chen, and Ziwei Liu. Embodied turing machines: Stateful code for robot recursive self-improvement. *arXiv preprint arXiv:2610.12369*, 2026. 2, 3

[7] 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*, 2025. 5

[8] Haoran Lang, Haotao Lu, Shiyu Sang, Haoyang Luo, Guo Chen, Qun Li, Jingyi Yu, Ye Shi, and Jingya Wang. Embodirsi: Recursive self-improvement for data-efficient robot adaptation. *arXiv preprint arXiv:2609.38905*, 2026. 3

[9] Wenyi Li, Yizhe Chi, Youjie Zheng, Chunyu Liu, Xiaoqiu Wang, Mingju Gao, Zheng Jiang, Deyao Hong, Calvin Xiao, Kaisen Yang, et al. Embodiedevo: Self-evolving coding agent for physical intelligence. *arXiv preprint arXiv:2610.18867*, 2026. 2, 3

[10] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pages 9493–9500. IEEE, 2023. 2, 3

[11] Tyler Ga Wei Lum, Olivia Y Lee, C Karen Liu, and Jeannette Bohg. Crossing the human-robot embodiment gap with sim-to-real rl using one human demonstration. *arXiv preprint arXiv:2504.12609*, 2025. 3

[12] Hiroki Nishizawa, Takehiko Ohkawa, Nicola Scianca, Tatsuya Matsushima, Shigeo Morishima, and Kei Ota. Yubi-real2sim: 3dgs scene verification for reliable policy evaluation. *arXiv preprint arXiv:2610.17521*, 2026. 2, 3

[13] Ruiyang Si, Jianxin Bi, Shunyu Yang, Rui Ni, Wenbo Huang, Qiang Wang, Shulong Jiang, Duomin Wang, Xiuyu Li, Haiwen Feng, et al. Fewer tokens, better action: Gpt-6 astra robot agents with 14% higher success rate but 65% fewer tokens. *arXiv preprint arXiv:2610.01939*, 2026. 3

[14] Zheng Sun, Zhihao Li, Zhuo Li, Wenrui Liu, Xin Wang, and Fei Chen. Reskill: Selective self-improvement for robot manipulation. *arXiv preprint arXiv:2611.10510*, 2026. 3

[15] Wenqing Tian, Zeyu Zhang, Zhaocheng Liu, Fengwei Liu, Qiang Liu, and Liang Wang. Physevo: Astra can act, let it. *arXiv preprint arXiv:2610.08995*, 2026. 2, 3

[16] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012. 5

[17] 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. 2, 3

[18] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 3

[19] Yuki Wang, Gonzalo Gonzalez-Pumariega, Yash Sharma, and Sanjiban Choudhury. Demo2code: From summarizing demonstrations to synthesizing code via extended chain-of-thought. *Advances in Neural Information Processing Systems*, 36:14848–14956, 2023. 3

[20] Yen-Jen Wang, Haozhe Jiang, Shuying Deng, Haoru Xue, Weirui Ye, Rocky Duan, Nika Haghtalab, S Shankar Sastry, Pieter Abbeel, and Haozhi Qi. Reconstruct, practice, go real: Guided self-improvement for embodied agents. *arXiv preprint arXiv:2610.02204*, 2026. 2, 3

[21] Xuan Wen and Baichao Wang. D-world: An explicit world-language-action model with recursive self-improvement under dual control. *arXiv preprint arXiv:2610.17114*, 2026. 3

[22] Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, et al. Enpire: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026. 2, 3

[23] Sizhe Yang, Wenye Yu, Jia Zeng, Jun Lv, Kerui Ren, Cewu Lu, Dahua Lin, and Jiangmiao Pang. Novel demonstration generation with gaussian splatting enables robust one-shot manipulation. *arXiv preprint arXiv:2504.13175*, 2025. 3

[24] Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong

Niu, et al. Playful agentic robot learning. *arXiv preprint arXiv:2606.19419*, 2026. 2, 3

Appendix

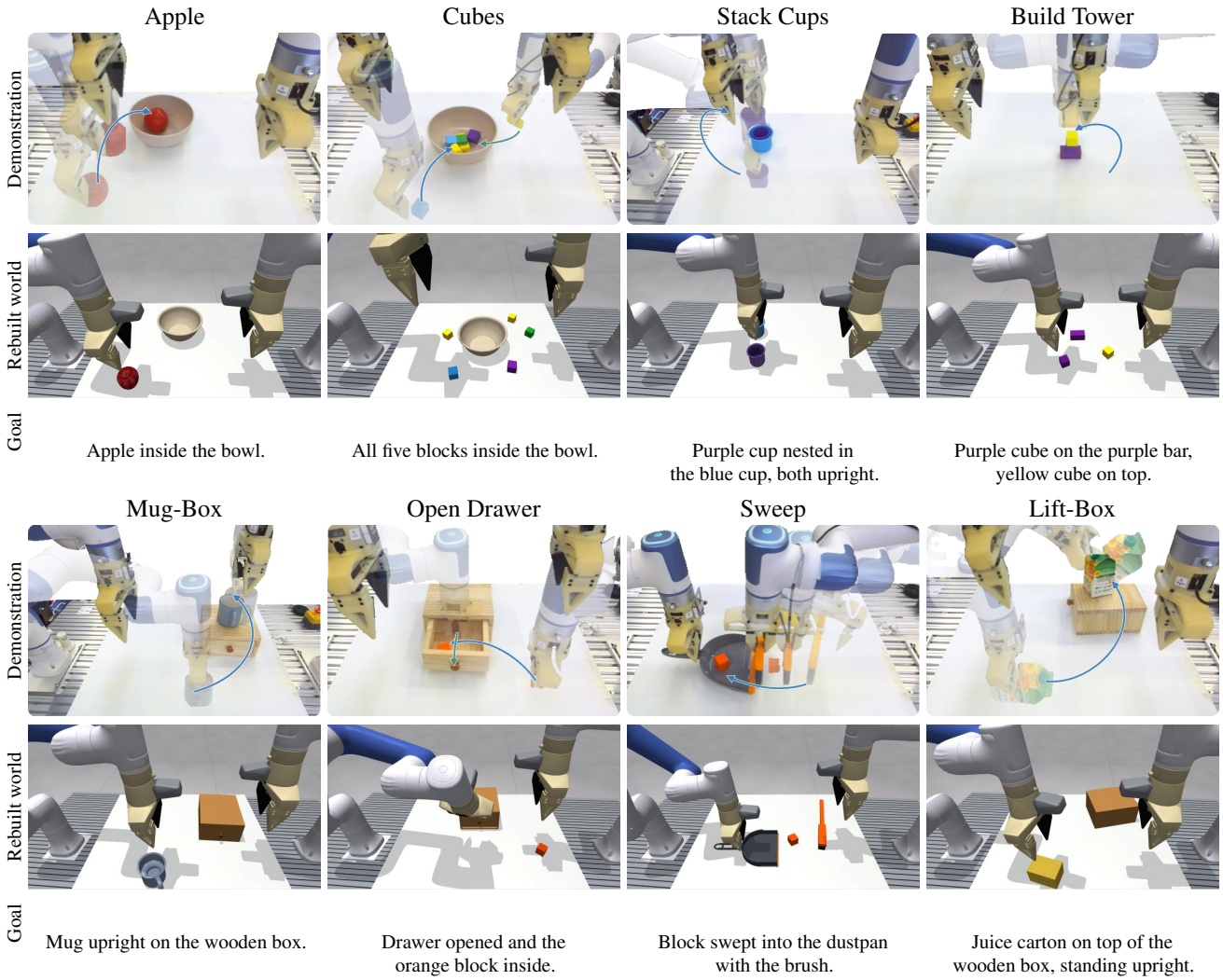


Figure S1. **The eight tasks and their rebuilt worlds.** Demonstration: the single demonstration on the source robot xTrainer, with the demonstrated object motion drawn as arrows. Rebuilt world: the interactive world rebuilt from that demonstration, rendered from the demonstration camera (a larger view of Fig. 3(a)). Goal: what the success criterion requires, in plain words.

A. Tasks and rebuilt worlds

Each of the eight tasks is demonstrated once on the source robot xTrainer (Sec. 4.1). Fig. S1 shows each demonstration together with the world rebuilt from it (Sec. 3.3) and the goal that the success criterion reads from the demonstrated final state. Every criterion also requires the objects to be released and still at the end of the episode.

B. Examine examples

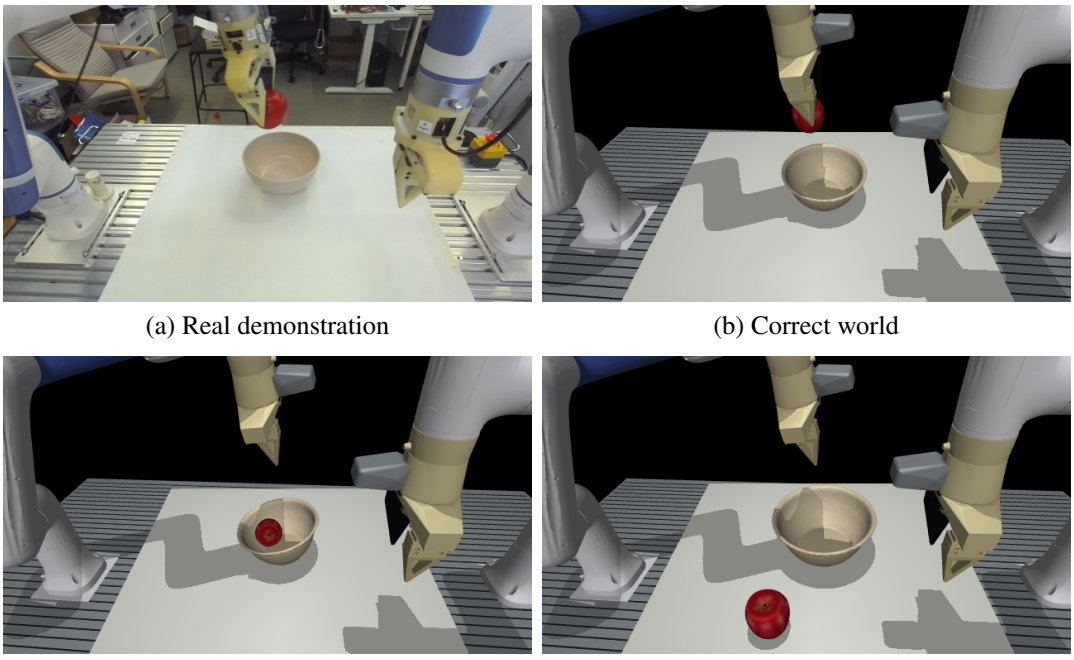
Examine (Sec. 3.4) checks a rebuilt world against the demonstration through two channels. Figs. S2 and S3 show both channels on the Apple task, in the correct world and in worlds with one of the geometric defects injected in Sec. 4.3: the table $2\,\mathrm{cm}$ too low, and all objects scaled by $\times 1.2$.

C. One round of Self-improve

In each round, Self-improve (Sec. 3.5) runs the program on diagnostic layouts, reads the status and failure reason of every node, and revises the code. Fig. S4 follows one such revision on the Apple task on the source robot xTrainer, from a failed practice episode of the round-0 program to the program after the round-1 revision.

D. Simulation execution on four robots

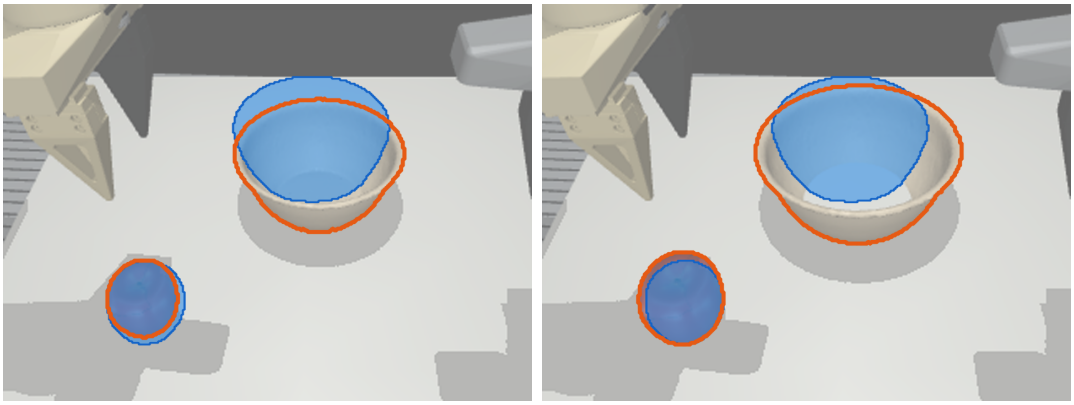
In Transfer (Sec. 3.6), each target robot practices in the same rebuilt worlds with the same criteria, without a demonstra-



(a) Real demonstration (b) Correct world

(c) Table 2 cm too low (d) Objects scaled $\times 1.2$

Figure S2. **The dynamic channel of Examine on the Apple task.** All four images show the same moment of the demonstration, shortly after the apple is lifted. (b)–(d) replay the recorded joint and gripper commands open-loop in three rebuilt worlds; the apple is never attached to the gripper, so it moves only through contact. (b) The apple is held in the gripper, as in the demonstration. (c) The gripper is empty and the apple lies in the bowl. (d) The gripper is empty and the apple is still on the table.



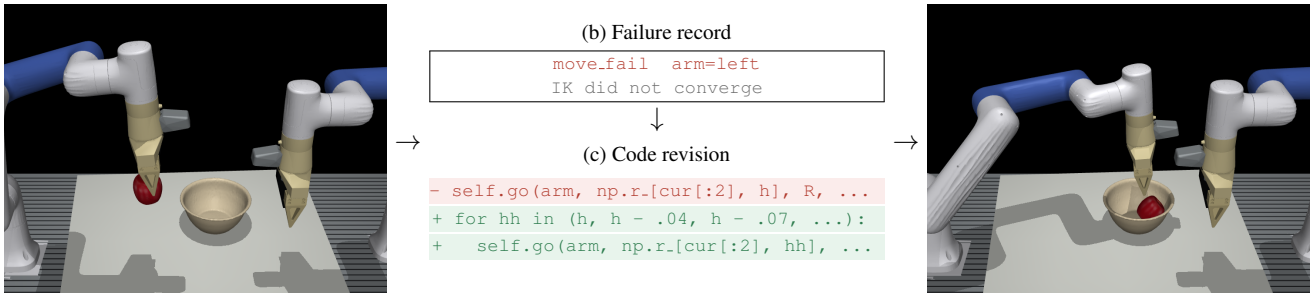
(a) Correct world (b) Objects scaled $\times 1.2$

Figure S3. **The visual channel of Examine on the Apple task.** Each rebuilt world is rendered at the recorded joint angles of the first frame, before contact (render shown lightened and cropped to the objects). Blue: the demonstration masks of the apple and the bowl. Orange: outlines of the same two objects rendered in the rebuilt world. The channel measures the silhouette IoU of each object against its demonstration mask. In (b), both rendered outlines are larger than in (a).

tion of its own. Fig. S5 shows one key frame of the final program of each of the four robots on each of the eight tasks. Success rates are reported in Tab. 5.

E. Real-robot execution

The final programs run on the real robots as ordinary code, with no code edits, no human intervention, and no LLM call (Sec. 3.6). Figs. S6a–S6d show, for each robot and task, three frames of one trial: the start, a key moment, and the final state. xTrainer, Dual-arm Franka, and ARX LIFT2 ran all eight tasks on the real robot, and Unitree G1 ran four of them (Apple, Cubes, Stack Cups, and Mug-Box), so Fig. S6d has two rows. Fig. S7 shows one program of Dual-arm Franka run from three different object layouts on each of three tasks.



(a) Failed practice episode

(d) After the revision

Figure S4. **One round of Self-improve on the Apple task.** (a) A practice episode of the round-0 program in the rebuilt world: the apple is held beside the bowl. (b) The failure record of that episode. (c) The revision the coding agent made in round 1 (lines shortened with an ellipsis): the single move of the arm to height `h` becomes a loop over decreasing heights. (d) A practice episode of the revised program: the apple is in the bowl.

F. Protocol and baselines

Real-robot protocol. The final programs are frozen after practice in simulation and run on the real robots as ordinary code, with no code edits, no human intervention, and no LLM call at run time (Sec. 3.6). Each task is run for 10 trials, with the objects placed by hand at varied positions. xTrainer, Dual-arm Franka, and ARX LIFT2 run all 8 tasks; Unitree G1 runs 4 of them (Apple, Cubes, Stack Cups, and Mug-Box). Dual-arm Franka, ARX LIFT2, and Unitree G1 received no demonstration. Results per task and robot are in Tab. 5; examples are shown in Fig. 3(b), in Figs. S6a–S6d and S7, and in the video.

Baselines. All baselines are evaluated in simulation on the source robot xTrainer, on the same 50 unseen test layouts per task (TEST 0–49) as RESET (Tab. 1).

- Open-loop Demo Replay replays the demonstration.
- Zero-shot Code takes 3 independent samples per task, with the same interfaces and contact priors as RESET; Tab. 1 reports the mean and the best of the three.
- $\pi_{0.5}$ is fine-tuned for 6,000 steps on 100 RESET trajectories from disjoint layouts, one model per task.
- Diffusion Policy is trained from scratch on the same trajectories, one model per task.

Both policies use their default settings.

G. Supplementary video

The video on the project page presents, in order, the problem (why self-improving robot code needs a world to practice in), the idea (what one real demonstration provides), the method (the five steps of RESET: Rebuild, Examine, Self-improve, Execute, and Transfer), and the simulation results. It then shows simulation rollouts of all four robots on all eight tasks and real-robot rollouts of the final programs on the four robots. Rollouts are sped up.

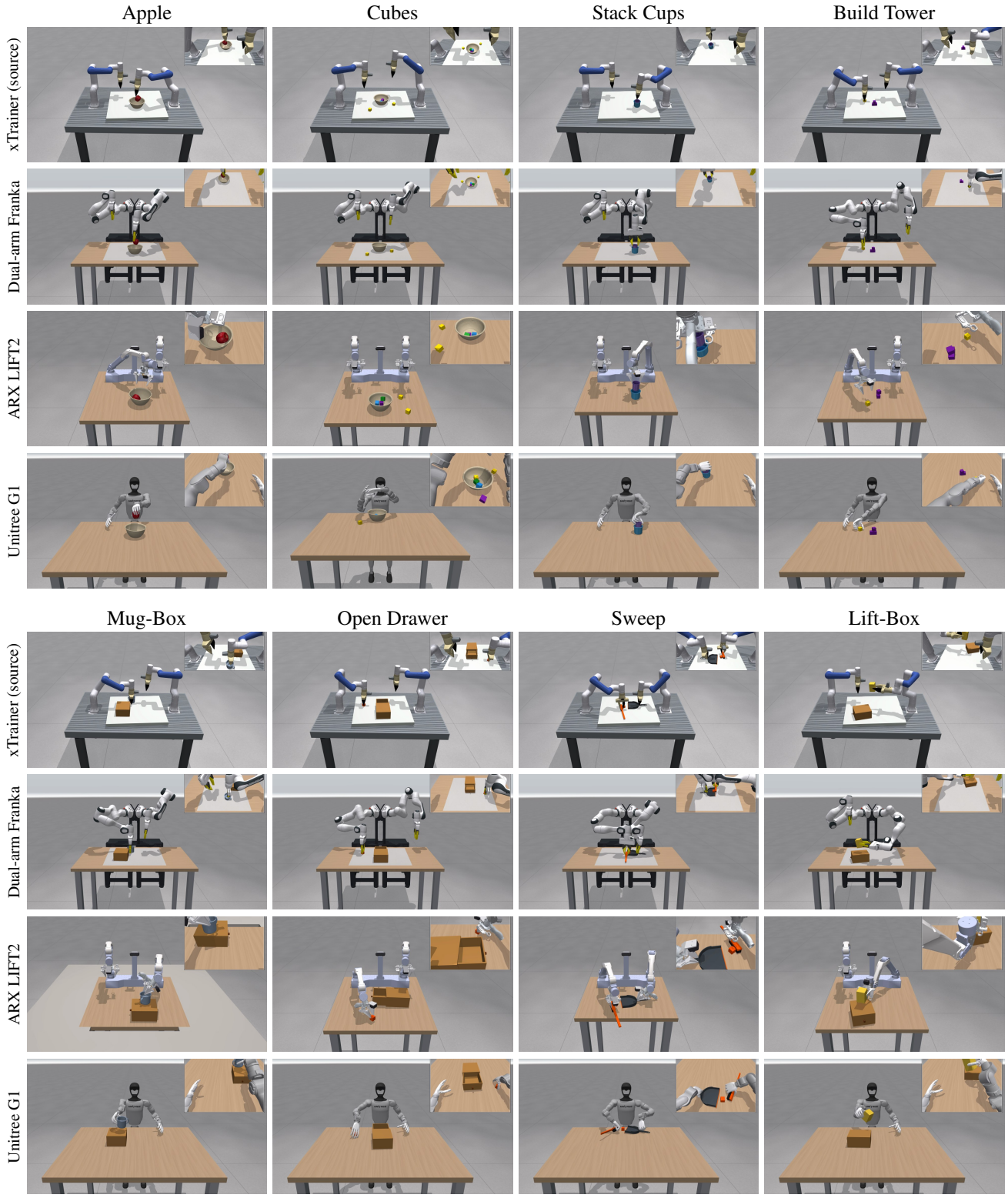


Figure S5. **Simulation execution, four robots on eight tasks.** One key frame of each robot's final program in the rebuilt world of each task; the inset at the top right shows the robot's own camera view. Dual-arm Franka, ARX LIFT2, and Unitree G1 received no demonstration.

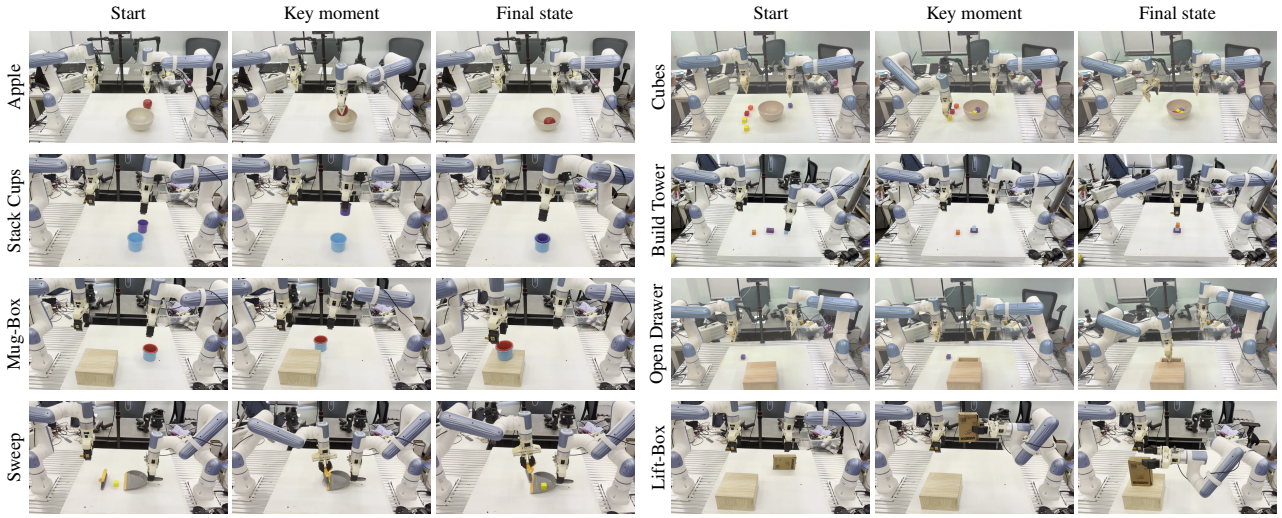


Figure S6a. **Real-robot execution on xTrainer (source robot).** Start, key moment, and final state of one trial per task.

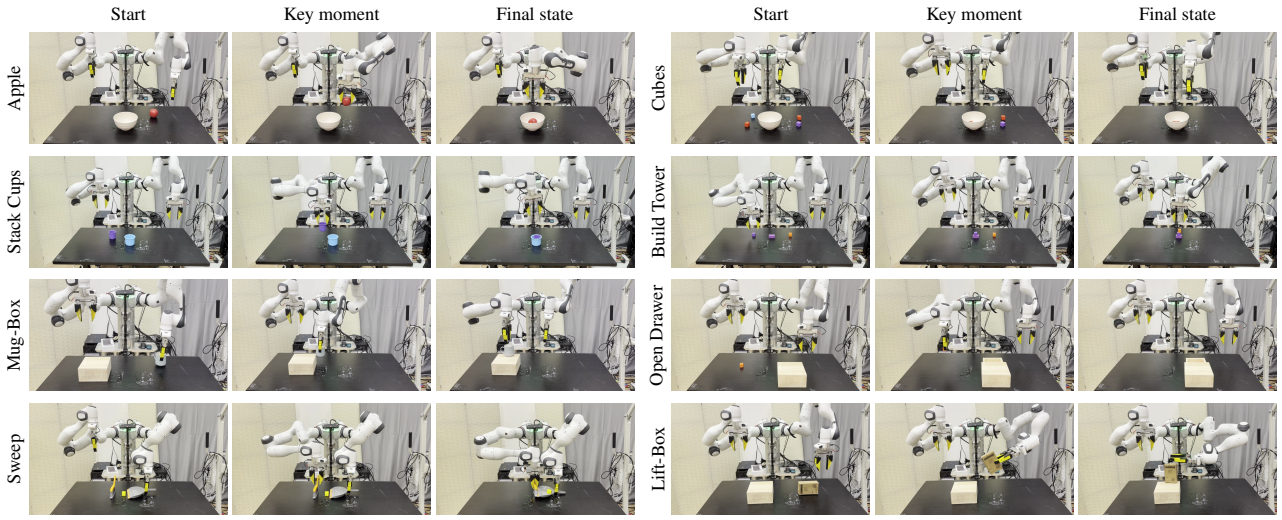


Figure S6b. **Real-robot execution on Dual-arm Franka (no demonstration).** Start, key moment, and final state of one trial per task.

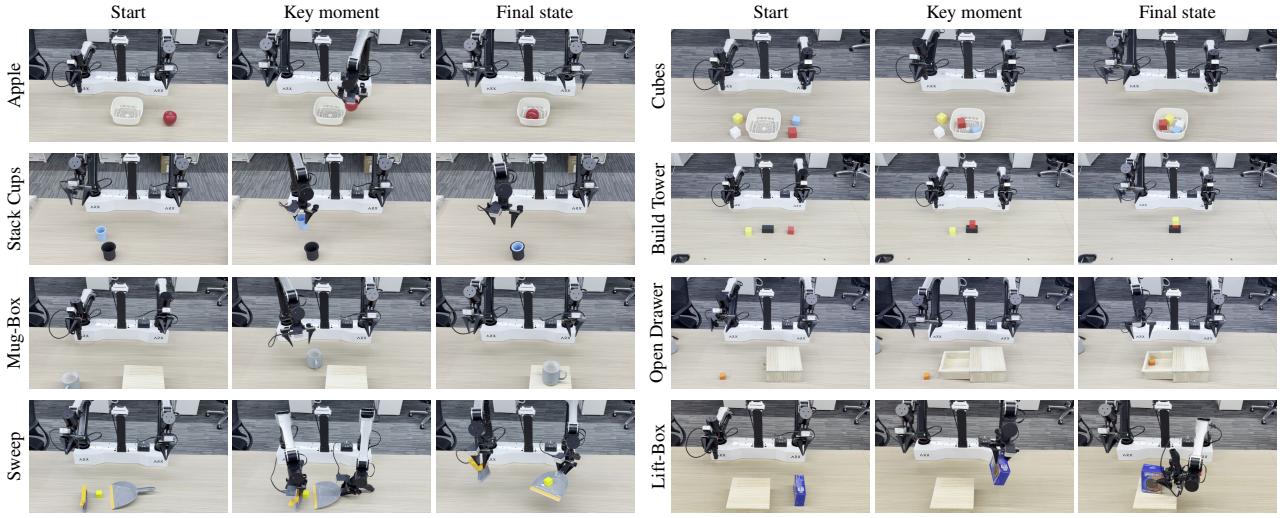


Figure S6c. **Real-robot execution on ARX LIFT2 (no demonstration).** Start, key moment, and final state of one trial per task.

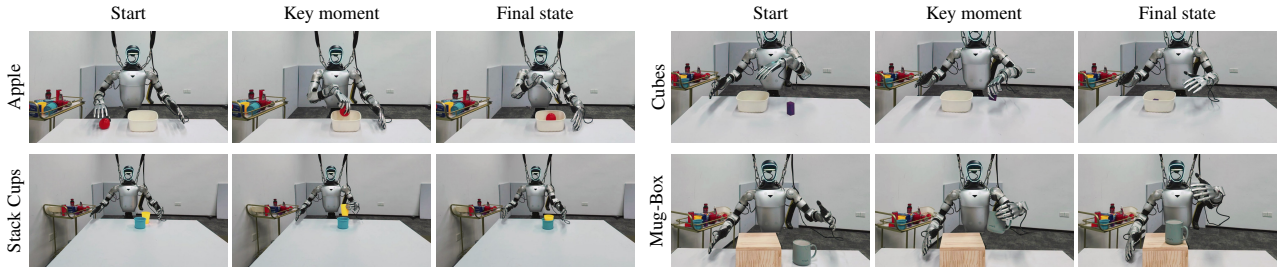


Figure S6d. **Real-robot execution on Unitree G1 (no demonstration).** The four tasks it ran on the real robot; start, key moment, and final state of one trial per task.

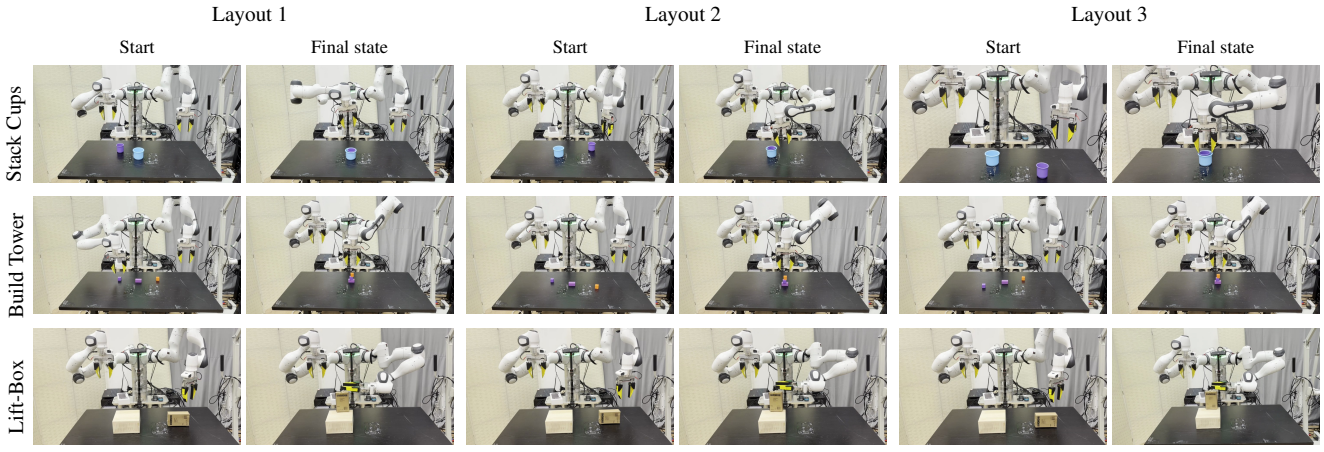


Figure S7. **Same program, different object layouts.** Dual-arm Franka on the real robot runs one program per task from three different object layouts, placed by hand; start and final frame of each trial.