PHYSEVO: ASTRA CAN ACT, LET IT

Anonymous authors
Paper under double-blind review

ABSTRACT

Astra can act, yet reliable manipulation depends on the system through which it observes and controls the world. We introduce PhysEvo, a framework for physical recursive self-improvement (RSI) around a single frozen model. A task agent executes robot tasks; a meta-agent uses the resulting trajectories to diagnose failures, revise tools and skills, and test corrections. The meta-agent can also improve its own diagnostic tools, so retained revisions support both later action and later self-improvement. This process develops joint-level control, evidence-seeking observation, and reusable manipulation skills without model-weight updates or a separately trained action policy. Across 42 RoboDojo tasks, held-out-layout evaluation of retained task-specific deployment versions yields a five-dimension average score of 68.14/100 and 62.00% success, compared with 47.17% for RoboDawn's one-shot Astra agent, the strongest published reference in our comparison. On eight manipulation tasks challenging direct Astra, success rises from 1.25% to 55.00%. Deploying the simulation-evolved harness on AgileX PiPER and continuing skill revision yields successful executions of five real-world tasks, adapting pouring and localization to physical discrepancies. PhysEvo turns the consequences of action into persistent, testable changes to how a frozen model acts and improves.

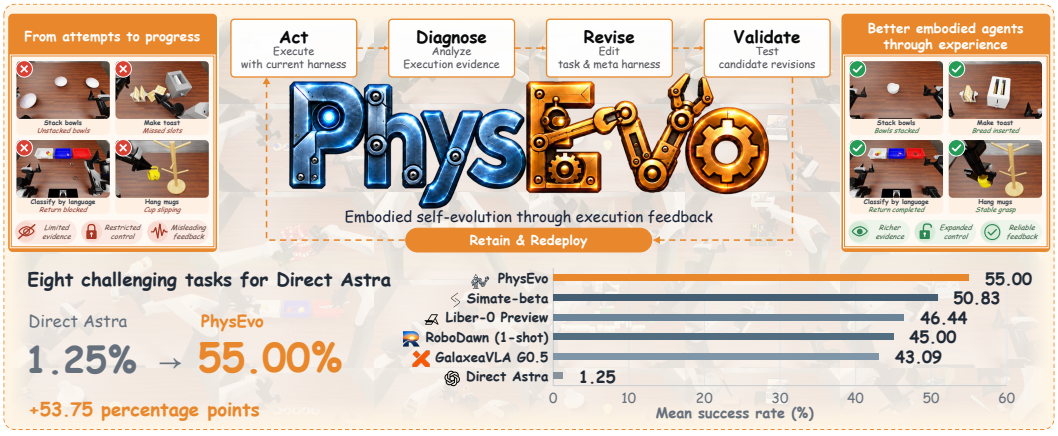


Figure 1: Physical RSI in PhysEvo: execute, diagnose, revise, test, and inherit. The lower panel compares success on eight tasks challenging direct Astra (table 3).

1 INTRODUCTION

Astra can already control a robot, but demonstrated competence does not guarantee reliable execution. In a ten-task RoboDojo evaluation, direct Astra reaches 100% success on object classification yet succeeds on only 26% of attempts overall. Pairing it with the specialized vision–language–action (VLA) policy $\pi_{0.5}$ raises success to 48% (Su et al., 2026). These results pose a question: *what prevents capabilities that are already present from being used reliably?*

Our executions expose bottlenecks in the *harness*—the tools, skills, and execution logic between the model and the robot. During pouring, the held bottle obscures the wrist view needed to monitor liquid delivery. With a tool-center-point (TCP) interface, local inverse kinematics (IK) can fold the

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.

arm into an obstructive posture, while leaving Astra no explicit way to choose the joint configuration (figures 3 and 4). These failures call for more than a new action through an unchanged interface: the agent needs a useful view or direct authority over posture.

Why should the same model be able to fix a failure it did not prevent? *Retrospective diagnosis has access to evidence that was unavailable during action selection.* A trajectory reveals the consequences of a command; aligned images, robot states, and code help trace those consequences to a cause. Astra can then propose a concrete intervention—maintain a free-arm side view or expose joint reconfiguration—and test it through further execution. In our pouring case, revised skill guidance changes the observation strategy, and the subsequent episode succeeds. Physical interaction thus provides both evidence for a correction and a test of whether it works.

A correction need not end with the episode. Retained tools and skills change later execution; retained diagnostic tools change how later failures can be investigated. In our packing case, Astra reuses a review tool it built earlier to trace box displacement to preceding contact and revise clearance guidance (figure 5). We call this loop *physical recursive self-improvement* (RSI): experience revises the system used for action and diagnosis, and the revised system is inherited by subsequent rounds.

We introduce **PhysEvo**, which implements this loop with one frozen Astra in two roles. The task agent observes and controls the robot. The meta-agent inspects execution, edits tools and skills, tests revisions, and retains useful changes, including changes to its own diagnostic resources. The resulting framework combines joint-level control, active and historical observation, geometry tools, and manipulation skills. No model weights are updated, and no separately trained action policy is required.

Across 42 RoboDojo tasks, held-out-layout evaluation of retained task-specific deployment versions achieves **68.14/100 score and 62.00% success** under equal five-dimension weighting. This exceeds the 47.17% reported by RoboDawn's one-shot Astra agent, the strongest reference in our comparison (table 1) (Guo et al., 2026a). On the ten-task direct/hybrid comparison, PhysEvo reaches 68.00% success versus 26.00% and 48.00%, respectively. On eight manipulation tasks challenging direct Astra, it reaches **55.00% versus 1.25%**, a **53.75-point gain**; Simate-beta, the strongest robot-policy reference on this subset, reports 50.83% (figure 1 and table 3) (RoboDojo Team, 2026; Zhang et al., 2026d).

On AgileX PiPER, the simulation-evolved harness serves as a starting point for continued adaptation. Trajectory analysis and skill revision yield successful executions of five real-world tasks. Revisions compensate for liquid-flow effects and replace geometry-first localization with a visually guided approach followed by close-range geometric refinement (section 5.6). The same loop can therefore revise both how the robot moves and when it seeks geometric evidence.

Our contributions are **(i)** a physical RSI framework in which one frozen model acts, revises, and inherits task and meta resources; **(ii)** an evolved interaction architecture combining joint control, evidence-seeking observation, and reusable skills; and **(iii)** a 42-task evaluation with mechanism-focused cases and five examples of continued adaptation on real hardware.

2 Related Work

Foundation models for robot control. Vision–language–action models such as RT-2, OpenVLA, and $\pi_{0.5}$ learn control from robot data (Brohan et al., 2023; Kim et al., 2024; Physical Intelligence et al., 2025). Code as Policies instead composes perception and control APIs, while Inner Monologue incorporates environmental feedback (Liang et al., 2022; Huang et al., 2022). CaP-X and Guava study tool, action, and observation design (Fu et al., 2026; Liu et al., 2026a); Show-Harness and obstacle-aware harnesses address embodiment and geometric constraints (Chen et al., 2026; Xu et al., 2026). RoboDawn couples discrete commands with feedback and demonstrations (Guo et al., 2026a). Thea and PhyAgentOS structure persistent state and outcome verification (Wang et al., 2026b; Liu et al., 2026b), while Harness VLA and RoboHarness use execution memory to guide primitives or route heterogeneous policies (Zhang et al., 2026e; Huang et al., 2026). These systems establish the importance of the interface around a model.

Self-improving agents. Reflexion retains verbal feedback, and GEPA evolves prompts from execution traces (Shinn et al., 2023; Agrawal et al., 2025). ADAS searches code-defined agents; SICA

and DGM modify agent implementations (Hu et al., 2024; Robeyns et al., 2025; Zhang et al., 2025). AutoHarness, Meta-Harness, and Self-Harness synthesize or revise executable harnesses from feedback (Lou et al., 2026; Lee et al., 2026b; Zhang et al., 2026a). Recursive Harness Self-Improvement evolves a prompt-level harness with a fixed optimizer (Lee et al., 2026a), whereas STOP and Hyperagents also modify the improvement procedure (Zelikman et al., 2023; Zhang et al., 2026b). RRSI regularizes harness edits, and Dream-RSI uses discovery histories to improve exploration (Xia et al., 2026; Zheng et al., 2026). Hyperagents' robotics experiment evolves rewards for PPO controllers (Zhang et al., 2026b, App. C.3); PhysEvo instead uses the same frozen model for robot decisions and software revision.

Experience-driven embodied agents. REFLECT diagnoses failures from execution summaries (Liu et al., 2023); Voyager and RATs accumulate executable skills through interaction and autonomous play (Wang et al., 2023; Zhang et al., 2026c). EmbodiSkill distinguishes faulty skills from failures to follow valid guidance (Ju et al., 2026); ASPIRE retains manipulation and debugging knowledge from multimodal traces (Lu et al., 2026). RoboRSI organizes skills hierarchically (Noematrix Team, 2026), and Zetta evolves critics, recovery, and tools around a frozen action policy (Ding et al., 2026). SHAPER evolves skills and context-construction code with one frozen planner/optimizer, while fixing the action interface and executor (Wang et al., 2026a). ENPIRE automates real-world policy development (Xiao et al., 2026). RHO evolves multi-file robot repositories, including prompts, tools, and control code for LLM-in-the-loop execution (Elmaaroufi et al., 2026). HarnessPAI revises programs and repair skills across rollouts, keeping the selected high-level program fixed within each rollout (Wang et al., 2026c). These close precedents motivate our focus on the joint evolution of skills, robot-facing action and observation interfaces, and diagnostic resources: the same frozen model acts through revised interfaces and reuses self-built tools to diagnose later failures.

3 PhysEvo: Physical Recursive Self-Improvement

PhysEvo alternates between acting in the environment and revising the system used to act. A single frozen model performs both roles. Physical outcomes provide evidence; tested changes to the harness preserve what is learned from that evidence (figure 2).

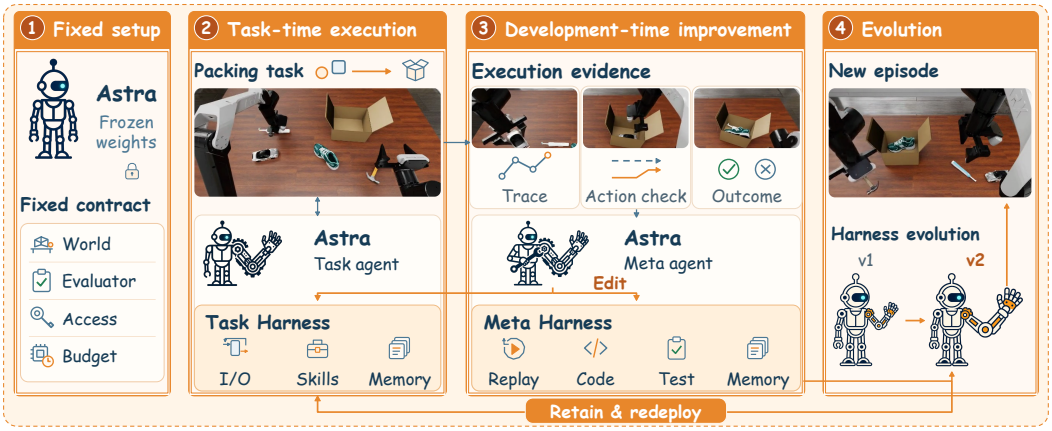


Figure 2: Physical RSI in PhysEvo. A frozen Astra acts through the task harness and revises both task and meta resources from execution feedback. Accepted revisions support later episodes and improvement rounds.

3.1 One Frozen Model, Two Editable Harnesses

Let M_θ denote Astra with fixed parameters. At iteration r, the persistent system consists of a task harness H_r^{task} and a meta harness H_r^{meta}:

$$H_r = (H_r^{\text{task}}, H_r^{\text{meta}}). \tag{1}$$

The task harness supplies observations, action tools, execution feedback, and skills. Task-agnostic skills guide operations such as grasp verification and recovery; task-specific skills guide activities such as pouring or packing. The meta harness provides code inspection and editing, trajectory review, diagnostic experiments, and revision testing. The meta-agent may modify either harness, including its own tools. The two roles use separate contexts and tool access but the same model. Model weights, robot dynamics, and the task evaluator are not editable.

Initialization provides a working observation–action loop with TCP targets and local IK, plus a general code-editing workspace. Joint-level control, specialized observation tools, and reusable skills develop through interaction rather than being supplied as a fixed catalog of solutions.

3.2 Task Execution and Physical Feedback

The task agent receives the instruction, selected skills, camera images, and robot proprioception. It selects an action or observation tool, then uses the returned result to make its next decision. Observation tools can retrieve evidence or calculate geometry without advancing the environment. Motion tools execute commands and return the resulting state and images.

For diagnosis, the harness links each requested action to its dispatched command and observed outcome. An unsuccessful grasp can thus be examined through approach, gripper closure, and subsequent object motion, rather than reduced to a final score. During execution, the same feedback supports closed-loop correction within the episode.

3.3 Revision, Testing, and Recursive Reuse

The meta-agent reviews execution alongside the current implementation, forms a failure hypothesis, and tests it through further inspection or diagnostic experiments. A revision may change action or observation tools, execution logic, skills, or diagnostic utilities. When a tool changes, its skill guidance is revised with it so that the agent inherits both a capability and instructions for using it.

Candidate revisions are tested during development, not on the held-out layouts used for final scoring. The meta-agent first checks that edited code runs and tools return the intended outputs. For behavioral changes, it compares the candidate with the current harness on the same task and initial scene. A score gain, or progress beyond the failure without a score decrease, supports adoption; a score decrease leads to rejection. A diagnostic-tool revision is checked against the error or measurement it is intended to address.

Accepted changes take effect between episodes. Changes to the task harness shape later execution; changes to the meta harness shape later diagnosis and revision. A review tool built in one round can therefore support a different correction in a later round. The improvement process inherits changes to its own machinery, rather than merely retrying a task. A retained deployment version is then fixed for that task's held-out evaluation (section 5.1). Section A gives the full procedure.

4 The Evolved Embodied Framework

Physical RSI reshapes what Astra can control, what evidence it can obtain, and what guidance persists. We describe the resulting architecture below; tasks retain their own deployment versions rather than one globally frozen harness (section 5.1).

4.1 Joint-Level Control

The default `control_arms` interface accepts twelve absolute arm-joint targets and two continuous gripper commands. It supports both arms and remains available while an object is held. The executor enforces native actuator limits and checks measured motion; Astra chooses the approach, arm configuration, and reconfiguration.

TCP control asks the model for an end-effector goal and delegates posture selection to IK. Joint control instead makes posture part of the policy decision. This additional authority lets Astra adjust an obstructive configuration and continue manipulation, rather than repeatedly requesting a target from

an unsuitable posture. Section C.1 compares the interfaces under fixed model, skill, initialization, and budget.

4.2 Active and Retrospective Observation

Astra can move a free arm to obtain a useful wrist-camera view while the other arm manipulates an object. When the relevant event is no longer visible, `review_frames` retrieves earlier images and aligned robot states from the current episode without advancing physics. Active viewing seeks new evidence; historical retrieval recovers evidence already observed.

Pixel projection and multi-view triangulation compute geometry for image features selected by Astra. Temporal triangulation uses views collected as a wrist camera moves. The model decides what to inspect and where to look; deterministic tools compute geometry and check validity using images and known camera/robot geometry, not hidden object coordinates. Temporal reconstruction requires a stationary point and a translational viewing baseline.

4.3 Reusable Skills and Diagnostic Tools

Skills link tools to the physical conditions under which they are useful: when to verify a grasp, obtain another view, reorient an object, clear an obstacle, or check a release. The pouring skill, for example, maintains a free-arm side view through pouring and upright recovery. A packing skill requires wrist and gripper clearance above raised box flaps before turning. Skills retain these lessons while leaving action decisions to Astra's closed-loop policy.

The meta harness retains trajectory-review tools that align images, robot states, and policy decisions. The task agent uses runtime tools and selected skills; the meta-agent uses the broader editing and diagnostic workspace. Experience is therefore retained in two forms: guidance for acting and tools for understanding why an action failed.

5 Experiments

We first evaluate retained deployment versions on held-out simulation layouts, then examine the revisions behind the resulting framework. Real-hardware experiments use a different setting: skill adaptation continues after deployment.

5.1 Experimental Setup

Tasks and metrics. We evaluate all 42 base RoboDojo simulation tasks across generalization, precision, long-horizon execution, memory, and open-ended instruction following. Native score ranges from 0 to 100; success rate (SR) counts scores of 100. Each task averages one attempt on each of five held-out layouts. The headline comparison weights the five dimensions equally after averaging tasks within each dimension. We also report the equal-task average and two task-level comparisons covering 16 distinct tasks (tables 2 and 3).

Development, sharing, and scoring. Simulation self-improvement precedes scoring. Development resources are shared across tasks; at the task-agent level, only tools and task-agnostic skills are shared, while task-specific skills remain task-local. Each task retains a deployment version that stays fixed across its five scored layouts. The 42-task result therefore aggregates task-specific versions, not one globally frozen harness. Scored layouts are excluded from revision and version selection. Before each scored episode, we reset session context, historical-frame indices, and writable memory. Online correction and same-episode history are allowed, but persistent tools and skills do not change. The meta-agent receives no privileged information from scored instances. Section B details these boundaries.

Comparison methods. References include RoboDojo robot and language-model policies (RoboDojo Team, 2026; Zhang et al., 2026d), GPT-as-Policy's direct Astra and Astra–$\pi_{0.5}$ hybrid (Su et al., 2026), and RoboDawn's one-shot Astra agent with closed-loop feedback and in-context demonstrations (Guo et al., 2026a). Bold and underlining indicate the best and second-best values within each comparison panel.

Table 1: RoboDojo performance across five capability dimensions. The average weights dimensions equally.

Method	General.		Precision		Long-horizon		Memory		Open		Average	
	Score	*SR (%)*	*Score*	*SR (%)*	*Score*	*SR (%)*	*Score*	*SR (%)*	*Score*	*SR (%)*	*Score*	*SR (%)*
Robot policies												
Simate-beta	35.09	27.95	34.35	26.92	57.84	43.42	33.33	33.00	9.12	8.50	33.95	27.96
Liber-0 Preview	24.99	18.61	38.28	33.17	45.98	33.17	37.77	37.33	6.68	5.33	30.74	25.52
Liber-0 Lite	25.36	19.11	36.60	31.08	45.50	32.92	35.68	35.44	3.06	2.58	29.24	24.23
DM0.5	15.78	10.95	24.82	16.75	33.70	19.50	47.74	47.44	2.43	2.08	24.90	19.34
GalaxeaVLA G0.5	18.46	12.83	28.25	20.42	44.12	32.25	8.61	7.33	1.73	1.58	20.23	14.88
Xiaomi-Robotics-1	23.55	17.00	26.69	18.83	38.39	23.67	7.81	6.56	3.94	3.58	20.07	13.93
OpenWAM-α	20.71	14.84	18.45	9.25	34.93	25.33	10.41	9.11	1.41	1.08	17.18	11.92
$\pi_{0.5}$	13.38	8.17	12.40	5.50	23.54	14.67	5.89	4.67	1.98	1.67	11.44	6.93
Language-model policies												
Official Astra	33.36	30.50	12.65	4.00	21.45	8.25	43.04	38.67	34.36	31.00	28.97	22.48
GPT-5.5	0.20	0.00	0.42	0.00	0.51	0.00	1.67	1.67	2.87	2.75	1.13	0.88
DeepSeek-Flash	2.42	1.67	1.75	0.00	2.12	0.00	2.17	1.67	6.50	6.25	2.99	1.92
RoboDawn (1-shot)	**55.08**	**50.00**	35.88	25.00	**59.88**	**45.00**	54.33	53.33	68.00	62.50	54.63	47.17
PhysEvo (ours)	51.08	45.00	**50.38**	**40.00**	56.38	**45.00**	**100.00**	**100.00**	**82.88**	**80.00**	**68.14**	**62.00**

RoboDawn uses GPT-6 Astra with one in-context demonstration; its Score and SR follow the public evaluation records.

5.2 Performance Across RoboDojo

Under five-dimension weighting, PhysEvo achieves **68.14/100 score and 62.00% SR**, exceeding RoboDawn one-shot's 47.17% SR by 14.83 points (table 1). This is the highest reported average among the compared methods, including an Astra agent already equipped with feedback and demonstrations. Equal weighting of all 42 tasks instead gives 65.00 score and 58.57% SR, corresponding to 123 successful attempts out of 210.

The gains are concentrated, not uniform. Relative to RoboDawn, SR increases by 46.67 points in memory, 17.50 in open-ended tasks, and 15.00 in precision. Memory supplies the largest contribution to the aggregate difference (section B.1). By contrast, generalization SR trails at 45.00% versus 50.00%, and long-horizon SR ties at 45.00%; RoboDawn also has the higher long-horizon score (59.88 versus 56.38). The overall lead therefore combines higher memory and instruction-dependent performance with improved precision.

5.3 Comparison with Direct and Hybrid Astra Policies

On the ten-task GPT-as-Policy set, PhysEvo reaches **68.00% SR**, versus 26.00% for direct Astra and 48.00% for the hybrid (table 2). The three systems organize Astra's capabilities differently: direct action, delegation to a trained VLA, or action through an experience-revised harness.

PhysEvo improves SR over Direct on eight tasks and ties on two. Gains span language-conditioned classification, sorting imitation, and number arrangement, as well as packing and bottle disposal. Against the hybrid, it leads on six tasks, ties on two, and trails on tower building and clothes folding. A trained action policy is thus not the only route to higher aggregate performance, but its strengths in these tasks remain complementary.

5.4 Manipulation Tasks Challenging Direct Astra

The RoboDojo Astra report identifies eight tasks with VLA SR $\geq 20\%$ and direct-Astra SR $< 5\%$ (Zhang et al., 2026d). This externally defined subset tests whether improvement reaches manipulation tasks that direct Astra rarely completes, rather than only consolidating existing successes.

PhysEvo achieves **55.00% SR and 62.88 score**, compared with official Astra's 1.25% and 5.96: a 53.75-point SR gain (table 3). Its mean SR also exceeds Simate-beta's 50.83%, Liber-0 Preview's

Table 2: Comparison on the ten-task GPT-as-Policy evaluation set. PhysEvo results average five layouts per task. Hybrid combines Astra with $\pi_{0.5}$; Direct uses Astra alone.

Task	G0.5		Xiaomi		O-WAM		Hybrid		Direct		**PhysEvo**	
	Score	*SR*	*Score*	*SR*	*Score*	*SR*	*Score*	*SR*	*Score*	*SR*	*Score*	*SR*
Organize table	46.33	11.33	57.67	**14.67**	**62.50**	13.33	60.00	0.00	30.00	0.00	60.00	0.00
Classify by language	1.07	0.00	2.00	0.67	1.33	0.00	38.00	20.00	60.00	40.00	**100.00**	**100.00**
Imitate sorting	1.67	0.00	2.50	0.67	2.90	0.67	53.00	40.00	0.00	0.00	**100.00**	**100.00**
Largest number	4.11	0.53	8.56	3.73	4.36	0.53	50.00	40.00	57.00	40.00	**100.00**	**100.00**
Pack objects	17.12	2.93	18.69	2.40	20.83	3.73	50.00	20.00	50.00	20.00	**70.00**	**40.00**
Classify objects	10.33	4.00	17.20	8.67	5.53	1.33	71.00	60.00	**100.00**	**100.00**	**100.00**	**100.00**
Build tower	**82.93**	**78.67**	52.60	47.33	52.53	38.67	64.00	60.00	12.00	0.00	40.00	20.00
Make kong	**90.00**	**90.00**	41.33	41.33	32.00	32.00	40.00	40.00	0.00	0.00	60.00	60.00
Fold clothes	32.75	28.00	41.49	38.93	51.31	49.07	**100.00**	**100.00**	40.00	40.00	68.00	60.00
Bottles into dustbin	96.30	94.00	97.70	96.67	94.03	91.33	**100.00**	**100.00**	36.00	20.00	**100.00**	**100.00**
Mean	38.26	30.95	33.97	25.51	32.73	23.07	62.60	48.00	38.50	26.00	**79.80**	**68.00**

Score is on a 0–100 scale; SR is in percent. The three VLA references have the highest ten-task average scores in the source report. G0.5 = GalaxeaVLA (G0.5); Xiaomi = Xiaomi-Robotics-1; O-WAM = OpenWAM-α.

Table 3: Eight manipulation tasks challenging direct Astra. PhysEvo results average five layouts per task. The task set is defined by the RoboDojo report, with VLA SR $\geq 20\%$ and Astra SR $< 5\%$.

Task	Official Astra		G0.5		Simate-beta		Liber-0 Preview		RoboDawn		**PhysEvo**	
	Score	*SR*	*Score*	*SR*	*Score*	*SR*	*Score*	*SR*	*Score*	*SR*	*Score*	*SR*
Make kong	0.00	0.00	**90.00**	**90.00**	76.00	76.00	18.00	18.00	60.00	60.00	60.00	60.00
Build tower	16.40	2.00	82.93	78.67	**87.40**	**83.33**	84.33	81.00	54.00	40.00	40.00	20.00
Insert tubes	6.00	0.00	58.53	42.67	41.20	22.00	82.53	73.00	**88.00**	**80.00**	**88.00**	**80.00**
Tic-tac-toe	10.70	0.00	65.23	40.00	86.27	60.67	95.67	89.00	**100.00**	**100.00**	**100.00**	**100.00**
Pour balls into vase	4.00	4.00	28.00	28.00	45.33	45.33	30.67	31.00	0.00	0.00	**80.00**	**80.00**
Pour liquid into cup	4.00	4.00	35.34	35.34	55.33	55.33	34.67	34.50	80.00	80.00	**100.00**	**100.00**
Laptop / headphones	1.60	0.00	28.27	11.34	**53.87**	**34.00**	38.14	20.00	12.00	0.00	16.00	0.00
Fill pen holder	5.00	0.00	41.27	18.67	**50.37**	**30.00**	49.77	25.00	26.00	0.00	19.00	0.00
Mean	5.96	1.25	53.70	43.09	61.97	50.83	54.22	46.44	52.50	45.00	**62.88**	**55.00**

SR is in percent. G0.5 = GalaxeaVLA (G0.5); RoboDawn uses GPT-6 Astra, 1-shot. For Simate-beta and Liber-0 Preview, Std/Rand Score and SR are averaged within task before the eight-task mean. Further references appear in section D.

46.44%, and RoboDawn's 45.00%. Against Simate-beta, however, it leads on four tasks and trails on four. The 4.17-point average margin summarizes different task strengths.

Against RoboDawn, PhysEvo wins two tasks, ties five, and loses one. Both wins are in pouring: balls rise from 0% to 80%, and liquid from 80% to 100%; tower building falls from 40% to 20%. These gains align with the observation and pouring cases below. Yet laptop/headphone storage and pen-holder filling score only 16 and 19, with no full successes: partial progress still falls short of reliable completion.

5.5 What Changes Through Physical RSI?

Three cases trace execution feedback to concrete changes in control, observation, and diagnosis.

Making posture a policy decision. Figure 3 shows two independent executions: an obstructive fold under TCP/local IK, and recovery from a rejected TCP motion through joint reconfiguration. End-effector goals leave posture to IK; joint control makes that missing choice expressible. In a focused two-layout comparison with the model, skill, initialization, and budget fixed, mean scores are 40 for TCP, 100 for Joint, and 55 for Both (six episodes; section C.1). Joint outperforming Both in this small test also shows that a larger tool menu alone is not sufficient.



Figure 3: Joint reconfiguration enables recovery from an obstructive arm posture.

Maintaining evidence during action. A pouring failure occurs while the bottle blocks the holding arm's wrist view. Revising the pouring skill, with the other loaded skills fixed, directs the free arm to maintain a side view through pouring and recovery. The task agent then checks the bottle mouth, cup rim, liquid landing point, and clearance, corrects its motion, and completes the task (figure 4). The change is not a new sensor: it is a policy for keeping existing sensors useful as manipulation alters visibility.

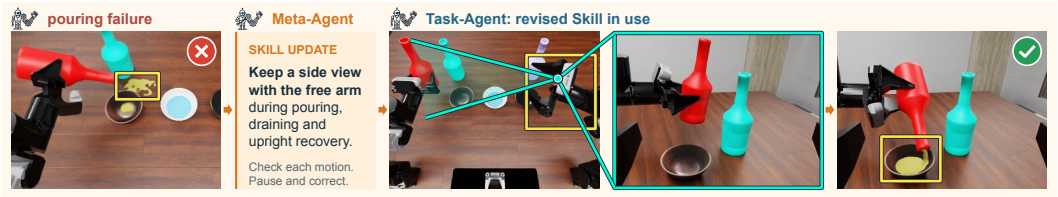


Figure 4: Skill revision maintains a useful view throughout pouring and recovery.

Inheriting a means of diagnosis. In figure 5, the meta-agent invokes a review tool built in an earlier round. Aligned images, robot states, and policy decisions help trace box displacement to wrist or gripper contact with raised flaps. The skill is then revised to require clearance above the flaps before turning. In the revised execution, the task agent follows this guidance and avoids pushing the box away. The final symptom—a box out of reach—thus leads to a specific intervention on the preceding contact. Crucially, a diagnostic resource created by one improvement round helps produce a correction in another: the system inherits a way to improve, not only a way to act.

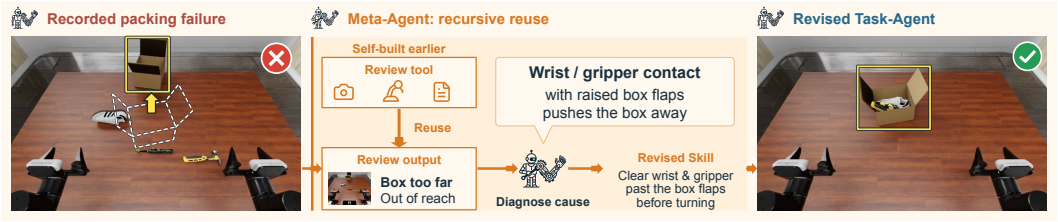


Figure 5: An inherited review tool supports a later packing-skill revision.

5.6 Real-World Adaptation through Skill Revision

We deploy the simulation-evolved harness on an AgileX PiPER platform and continue the execution–revision loop. After each attempt, the meta-agent analyzes the trajectory and revises skills; the task agent uses those revisions in the next attempt. A small number of these cycles yields successful executions of block stacking, bowl stacking, pen placement, liquid pouring, and writing "PhysEvo" with a pen (figure 6). Two revisions illustrate how physical discrepancies change the inherited skills.

Compensating for liquid motion. The liquid stream's forward momentum causes it to overshoot the cup, an effect insufficiently captured in our simulated setup. The meta-agent revises the pouring skill to shift the bottle opening backward relative to the cup, compensating for the observed displacement.

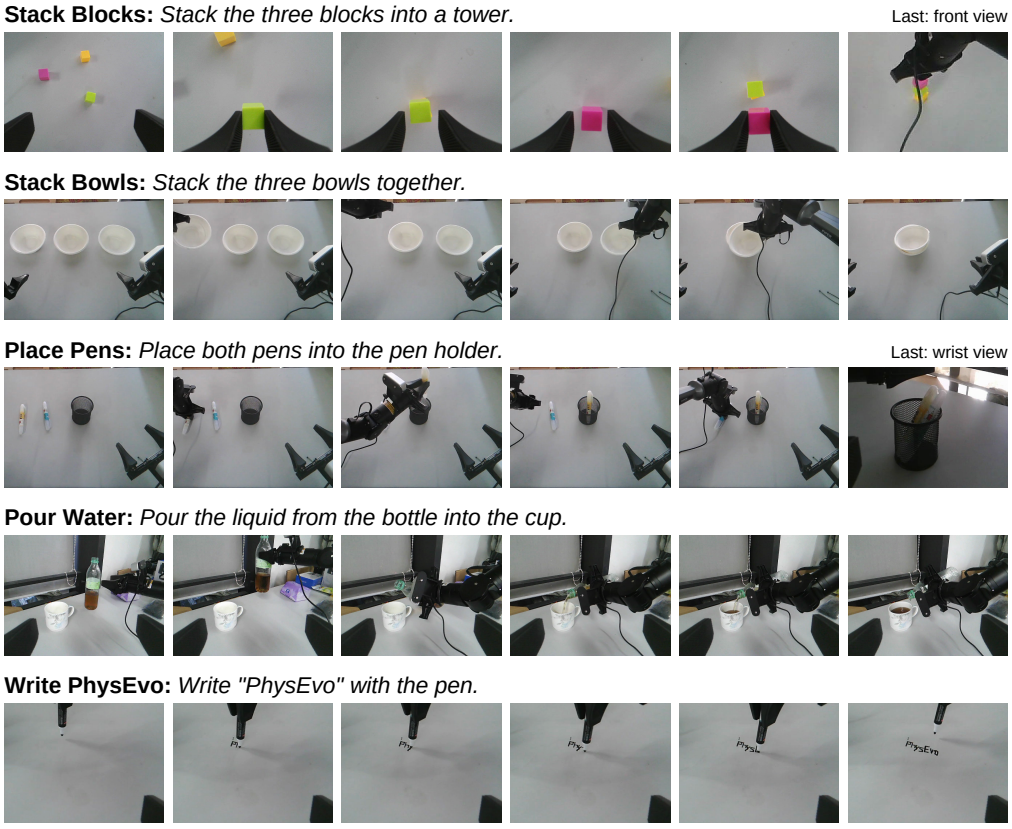


Figure 6: Five AgileX PiPER tasks after trajectory-driven skill revision. Each row shows one execution in time; alternate final views are marked.

Where the simulation revision preserves a view of the pour, the hardware revision changes the action to account for the observed flow.

Changing when geometry is used. Our physical arm has lower positioning accuracy than its simulated counterpart, and localization from the initial distant view is often unreliable. The revised skill reverses the geometry-first sequence: visual feedback guides a coarse approach, then geometry provides close-range localization for fine positioning. Geometry is not discarded; it is invoked under more reliable viewing conditions. This changes the order of perception and action, not just a motion parameter.

These cases demonstrate continued post-deployment adaptation. Additional simulation executions appear in section E.

6 Discussion and Conclusion

PhysEvo uses physical consequences to revise the tools, skills, and diagnostic resources of a frozen Astra. Its retained deployment versions achieve 62.00% success under five-dimension weighting across 42 RoboDojo tasks. The cases identify three ways experience changes the system: exposing posture control, maintaining useful observations, and inheriting tools for later diagnosis. On AgileX PiPER, the same loop adapts skills to liquid motion and localization discrepancies. The central lesson is that experience can improve not only the next action, but the interface and diagnostic resources available to future actions.

Limitations. Our hardware evaluation remains qualitative, and observation and model calls introduce additional latency.

AI USE STATEMENT

Generative AI tools were used to refine the language and presentation of the manuscript. The authors led the development of PhysEvo and its experimental evaluation, verified the reported measurements and analyses, and reviewed all AI-assisted material. Responsibility for the complete paper—including its text, claims, figures, and other artifacts—rests with the authors.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025. URL https://arxiv.org/abs/2507.19457.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Yanzhe Chen, Zechen Bai, Zhijun Cao, Wenzheng Zeng, Kevin Qinghong Lin, Yiqi Lin, Guoqiang Liang, Kevin Yuchen Ma, Qiming Huang, and Mike Zheng Shou. Show-Harness: Just a VLM agent can play robots. *arXiv preprint arXiv:2609.10522*, 2026. URL https://arxiv.org/abs/2609.10522.

Xin Ding et al. Zetta: An efficient closed-loop embodied harness for self-evolving physical intelligence. *arXiv preprint arXiv:2608.16590*, 2026. URL https://arxiv.org/abs/2608.16590.

Karim Elmaaroufi, Justin Svegliato, Sarunas Kalade, Graham Schelle, Sanjit A. Seshia, and Matei Zaharia. RHO: Your coding agent is secretly a roboticist. *arXiv preprint arXiv:2606.16458*, 2026. URL https://arxiv.org/abs/2606.16458.

Letian Fu et al. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026. URL https://arxiv.org/abs/2603.22435.

Meng-Hao Guo, Zhe-Han Mo, Jia-Jun Wang, Yi Zhang, Kejin Wang, Yi-Xuan Deng, Jia-Peng Zhang, Yongming Rao, and Shi-Min Hu. Transferring the intelligence of VLMs to robotic control. *arXiv preprint arXiv:2609.22966*, 2026a. URL https://arxiv.org/abs/2609.22966.

Meng-Hao Guo, Zhe-Han Mo, Jia-Jun Wang, Yi Zhang, Kejin Wang, Yi-Xuan Deng, Jia-Peng Zhang, Yongming Rao, and Shi-Min Hu. RoboDawn: Public task-level evaluation results, 2026b. URL https://robodawn.top/results/. RoboDojo, GPT-6 Astra, one-shot setting; five runs per task. Recorded 26 September 2026.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024. URL https://arxiv.org/abs/2408.08435.

Jinbang Huang, Yuanzhao Hu, Zhiyuan Li, Ran Qi, Yixin Xiao, Zhanguang Zhang, Mark Coates, Tongtong Cao, and Yingxue Zhang. RoboHarness: Memory-driven orchestration of heterogeneous robot policies for long-horizon planning. *arXiv preprint arXiv:2607.18060*, 2026. URL https://arxiv.org/abs/2607.18060.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. URL https://arxiv.org/abs/2207.05608.

Ruofei Ju, Xinrui Wang, Xin Ding, Yifan Yang, Hao Wu, Shiqi Jiang, Qianxi Zhang, Hao Wen, Xiangyu Li, Weijun Wang, Kun Li, Yunxin Liu, Haipeng Dai, Wei Wang, and Ting Cao. EmbodiSkill: Skill-aware reflection for self-evolving embodied agents. *arXiv preprint arXiv:2605.10332*, 2026. URL https://arxiv.org/abs/2605.10332.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement. *arXiv preprint arXiv:2607.15524*, 2026a. URL https://arxiv.org/abs/2607.15524.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026b. URL https://arxiv.org/abs/2603.28052.

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. *arXiv preprint arXiv:2209.07753*, 2022. URL https://arxiv.org/abs/2209.07753.

Haowen Liu, Xirui Li, Shaoxiong Yao, Peng Shi, Tianyi Zhou, Jia-Bin Huang, Furong Huang, and Jiayuan Mao. Guava: An effective and universal harness for embodied manipulation. *arXiv preprint arXiv:2606.18363*, 2026a. URL https://arxiv.org/abs/2606.18363.

Yang Liu, Weixing Chen, Xinshuai Song, Tao Pu, Siwen Mo, Yongjie Bai, Zihao Chen, Qianran Sun, Liruo Zhong, Ying Shen, and Liang Lin. PhyAgentOS: A self-evolving operating system for embodied agents with decoupled cognitive planning and physical execution. *arXiv preprint arXiv:2607.16636*, 2026b. URL https://arxiv.org/abs/2607.16636.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*, 2023. URL https://arxiv.org/abs/2306.15724.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. AutoHarness: Improving LLM agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026. URL https://arxiv.org/abs/2603.03329.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi Fan, and Guanzhi Wang. ASPIRE: Agentic /skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026. URL https://arxiv.org/abs/2607.00272.

Noematrix Team. RoboRSI: Stable, efficient, and reusable robot self-evolution in complex real-world environments. Noematrix Lab research blog, September 2026. URL https://lab.noematrix.ai/blog/2-roborsi/. Accessed 21 September 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025. URL https://arxiv.org/abs/2504.15228.

RoboDojo Team. RoboDojo Simulation Leaderboard, 2026. URL https://robodojo-benchmark.com/leaderboard. Data updated 23 September 2026; task-level results recorded 26 September 2026.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Jiayi Su, Yixin Zheng, Mi Yan, Li Yi, Zhizheng Zhang, and He Wang. GPT 6 Astra as an embodied policy. Technical report and code, 2026. URL https://github.com/anonymous-report-421/GPT-as-Policy. Accessed 21 September 2026.

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. URL https://arxiv.org/abs/2305.16291.

Peidong Wang, Zhiming Ma, Ying Chang, Xufang Luo, Yiqun Zhang, Zihan Wang, Xiaocui Yang, Shi Feng, Yuqing Yang, and Dongsheng Li. Self-evolving embodied agents via skill-harness evolution. *arXiv preprint arXiv:2608.11350*, 2026a. URL https://arxiv.org/abs/2608.11350.

Qi Wang, Tianyi Wang, Chengyang Li, Shikun Ban, Yurun Chen, Yizhong Ge, Jason Qin, Chengtai Li, and Wentao Zhu. Towards the harness of embodied agents. *arXiv preprint arXiv:2608.11246*, 2026b. URL https://arxiv.org/abs/2608.11246.

Xin Wang, Wenhao Wu, Menghao Zhang, Zhi Wang, Kun Shao, Jian Luan, Yang Li, Qing Li, Shangding Gu, Huichi Zhou, Shuqing Shi, Fei Ni, Shuo Lu, Weicheng Meng, Kang Li, Jin Wu, Kang Zhao, Shangmin Guo, Gen Li, Yongqiang Tang, Zhizhong Zhang, Yuan Xie, and Heng Qu. HarnessPAI: An evolving harness for physical AI. *arXiv preprint arXiv:2609.29166*, 2026c. URL https://arxiv.org/abs/2609.29166.

Peng Xia, Rujun Han, Zifeng Wang, Yanfei Chen, Yufan Zhuang, Yoonho Lee, Chengsong Huang, Han Yu, Zhongying CuiZhu, Yifei Ming, Huaxiu Yao, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. RRSI: Regularized recursive self-improvement of agent harnesses. *arXiv preprint arXiv:2609.24972*, 2026. URL https://arxiv.org/abs/2609.24972.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026. URL https://arxiv.org/abs/2606.19980.

Bingxin Xu, Yuzhang Shang, Zhen Dong, and Emilio Ferrara. Coding agents with an obstacle-aware harness for safe robot manipulation. *arXiv preprint arXiv:2609.20822*, 2026. URL https://arxiv.org/abs/2609.20822.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*, 2023. URL https://arxiv.org/abs/2310.02304.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026a. URL https://arxiv.org/abs/2606.09498.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents. *arXiv preprint arXiv:2603.19461*, 2026b. URL https://arxiv.org/abs/2603.19461.

Jenny Zhang et al. Darwin Godel Machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025. URL https://arxiv.org/abs/2505.22954.

Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong Niu, Zirui Wang, Roei Herzig, Ken Goldberg, Yutong Bai, David M. Chan, Ion Stoica, Angjoo Kanazawa, Jiahui Lei, Haiwen Feng, and Trevor Darrell. Playful agentic robot learning. *arXiv preprint arXiv:2606.19419*, 2026c. URL https://arxiv.org/abs/2606.19419.

Wenbo Zhang, Kaixuan Wang, Yutao Ouyang, Xiaoyu Huang, Liyang Li, Kailun Su, Weiyang Jin, Wenhao Chai, Haotian Liang, Zhiyang Dou, Yue Chen, and Tianxing Chen. An unexpected robot policy: Early evaluations of gpt-6 astra on robodojo and beyond, 2026d. URL https://robodojo-benchmark.com/report/gpt-6-astra-eval. Equal contribution: Wenbo Zhang, Kaixuan Wang, Yutao Ouyang. Project lead: Wenbo Zhang. Corresponding: Yue Chen, Tianxing Chen.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Yi Nie, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Jiaji Rao, Zhengru Fang, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness VLA: Steering frozen VLAs into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026e. URL https://arxiv.org/abs/2607.08448.

Tong Zheng, Xidong Wu, Zheng Zhang, Zhankui He, Chaoyi Zhang, Benjamin Coleman, Ruoqiao Wei, Di Bai, Haolin Liu, Rui Liu, Xue Wang, Yue Zhuan, Wang-Cheng Kang, Renkai Xiang, Heng Huang, Xinwu Cheng, and Yunsong Guo. Dream-RSI: Recursive self-improvement through evolving worlds. *arXiv preprint arXiv:2609.14858*, 2026. URL https://arxiv.org/abs/2609.14858.

A PHYSICAL RSI PROCEDURE

At iteration r, the persistent system is $H_r = (H_r^{\text{task}}, H_r^{\text{meta}})$. Both roles use the same frozen model M_θ. The task agent selects tools from a context assembled by the task harness:

$$a_t \sim M_\theta(\cdot \mid P_{H_r^{\text{task}}}(\ell, o_{\le t}, m_t)), \qquad (o_{t+1}, f_t) = \text{RunTool}_{H_r^{\text{task}}}(a_t). \tag{2}$$

Here ℓ is the instruction, $o_{\le t}$ contains observations, m_t is episode memory, and f_t is execution feedback. The index t counts tool interactions. A motion call can execute several native steps; a review or geometry call need not advance the environment.

Let E_r contain feedback available to the meta-agent. One improvement round is summarized by

$$\widetilde{H}_r \leftarrow \text{Revise}_{M_\theta, H_r^{\text{meta}}}(H_r, E_r), \tag{3}$$

$$y_r \leftarrow \text{Test}(\widetilde{H}_r, H_r), \tag{4}$$

$$H_{r+1} \leftarrow \text{Select}_{M_\theta, H_r^{\text{meta}}}(H_r, \widetilde{H}_r, y_r). \tag{5}$$

$\widetilde{H}_r$ is a candidate revision and y_r contains its test results. The selection step either adopts the candidate or preserves the current harness. Robot dynamics, actuator restrictions, model configuration, and evaluation rules are outside the editable system.

Table 4: Physical RSI procedure in PhysEvo.

Inputs: frozen Astra M_θ, initial task and meta harnesses H_0, and an interaction budget.

1 Execute a task using the current task harness.
2 Give the meta-agent the task outcome, relevant images and robot states, action feedback, and current code.
3 Let the meta-agent inspect the failure, retrieve evidence, and run diagnostic experiments.
4 Propose a candidate revision to tools, skills, execution logic, or meta resources.
5 Run the edited code and check its outputs. For task-behavior changes, compare the candidate with the current harness on the same task and initial scene.
6 Adopt a candidate with a score gain, or progress beyond the failure without a score decrease. Reject a candidate that lowers the score. Test diagnostic changes against the error or measurement they address.
7 Load accepted tools and their associated skills together in subsequent episodes. Reuse accepted meta resources in subsequent improvement rounds.
8 Repeat within the interaction budget, then retain a deployment version for that task and evaluate held-out layouts without further updates.

Initialization and persistence. The initial task harness supplies instructions, camera observations, proprioception, and a TCP/local-IK action interface. The initial meta harness supplies a code-editing and execution workspace. Retained tools and skills form a compatible system: an updated tool and the guidance for using its API are loaded together. Updates take effect between episodes rather than changing an active episode's tools. Selected skills are loaded at episode start. The task agent uses its runtime tools; the meta-agent uses the code-editing workspace. Development resources are shared across tasks. At the task-agent level, tools and task-agnostic skills are shared, while task-specific skills remain task-local. Each task retains its own deployment version before scoring.

B EVALUATION DETAILS

B.1 METRICS AND AGGREGATION

For task t, let $s_{t\ell}$ be the native score on layout $\ell \in \{1, \ldots, 5\}$. We compute

$$M_t = \frac{1}{5}\sum_{\ell=1}^{5} s_{t\ell}, \qquad \text{SR}_t(\%) = \frac{100}{5}\sum_{\ell=1}^{5} \mathbf{1}[s_{t\ell} = 100]. \tag{6}$$

The 42-task average gives equal weight to each task, yielding 65.00 score and 58.57% SR. Because each task contributes five layout scores, this task-macro average also equals the pooled-layout average. The five-dimension average first averages tasks within each dimension and then weights the dimensions equally, yielding 68.14 score and 62.00% SR.

The ten-task GPT-as-Policy set and the eight-task manipulation subset overlap on tower building and making kong. Their union contains 16 distinct tasks and 80 attempts, giving 75.06 score and 65.00% SR. Each task is counted once in this union; the two subset averages are not averaged together.

Development and held-out scoring. Self-improvement takes place during simulation development. For each task, a retained deployment version is fixed across its five scored layouts; these layouts are not used for harness revision or version selection. Development resources may carry experience across tasks, but task-agent sharing is restricted to tools and task-agnostic skills. Task-specific skills remain task-local.

Episode state and information access. Before every scored episode, session context, historical-frame indices, and writable memory are reset. The task agent loads its retained tools and selected skills, then may accumulate and revisit observations from that episode and correct its actions online. This within-episode feedback does not change persistent tools, skills, or the deployment version. Scored instances do not supply privileged information to the meta-agent or contribute to revision or version selection. The hardware experiments use a separate adaptation protocol: skills are revised between real-world attempts rather than frozen for a quantitative evaluation.

Interpreting the aggregate difference. Relative to RoboDawn, the five dimension-wise SR differences are -5.00, $+15.00$, 0.00, $+46.67$, and $+17.50$ percentage points for generalization, precision, long-horizon, memory, and open-ended tasks, respectively. Under equal dimension weighting, the memory difference contributes approximately 9.33 points to the 14.83-point overall margin, or about 63% of the net difference. This is an arithmetic decomposition, not evidence that a particular memory or observation tool caused that share of the gain. The eight-task SR lead is also a reported mean: its 40 PhysEvo attempts imply a 2.5-point change in mean SR for a single changed outcome. We do not infer statistical superiority from the 4.17-point margin over Simate-beta.

B.2 Comparison Sources and Aggregation

Five-dimension comparison. The robot and direct-language-model results in table 1 are taken from the RoboDojo leaderboard and Astra report (RoboDojo Team, 2026; Zhang et al., 2026d). The published generalization dimension averages the Gen-Std and Gen-Rand splits. The final average weights the five dimensions equally. RoboDawn's five-dimension Score and SR are computed from its public evaluation records: GPT-6 Astra, 1-shot, with five runs per task (Guo et al., 2026b). The dimension-weighted means, 54.63 score and 47.17% SR, agree with its paper (Guo et al., 2026a).

Direct and hybrid Astra policies. The ten-task set follows GPT-as-Policy (Su et al., 2026). PhysEvo is evaluated on five layouts per task, with one attempt per layout, for 50 attempts in total. The source reports 50 selected cases for the hybrid and 48 valid scores for Direct. We recompute Direct's score by averaging task means, giving 38.50 rather than its pooled-case score of 37.81. VLA task means use the source report's standard/random scene-mixture weights. The three VLA methods displayed in table 2 have the highest ten-task average scores among the source report's VLA references.

Manipulation-challenge subset. The eight tasks in table 3 correspond to Table F2 of the RoboDojo Astra report (Zhang et al., 2026d). Its selection criterion is VLA SR $\geq 20\%$ and Astra SR $< 5\%$. PhysEvo is evaluated on five scenes per task on this subset, with one attempt per scene. The main table uses the same six methods as figure 1. Simate-beta and Liber-0 Preview follow the 23 September 2026 leaderboard update, recorded on 26 September. For liquid pouring and laptop/headphone storage, Std/Rand Score and SR are averaged within task before computing the eight-task mean. RoboDawn uses GPT-6 Astra, 1-shot, with five runs per task (Guo et al., 2026b). Both Score and SR are shown in the task rows. Section D retains the other robot-policy references.

C Additional Framework Analysis

C.1 Control-Interface Comparison

A focused comparison on language-based object classification evaluates TCP, Joint, and Both interfaces on two scene layouts, for six episodes in total. The model, task skill, initialization for each layout, and budget are held fixed. Mean scores are 40 for TCP, 100 for Joint, and 55 for Both. Joint is retained as the default control interface. This comparison concerns the choice of action interface; the final framework uses the Joint interface described in section 4.

The control sequences in figure 3 illustrate two independent executions. The first develops an obstructive posture under TCP/local IK. In the second, Astra adjusts joints after a TCP rejection and then completes a motion to the same target pose. The mechanism is direct access to the arm configuration, which an end-effector target alone leaves to the IK solver.

C.2 Observation Tools

Historical-frame retrieval returns images and aligned robot states from earlier in the same episode. It does not advance the environment. Active viewing can place the free wrist camera to the side of a manipulation. Geometric tools combine model-selected image features with camera calibration and robot geometry.

A synthetic-marker reconstruction test accepted 8 of 12 points, with a median error of 0.615 mm and a 95th-percentile error of 1.437 mm. Four points were rejected by the validity checks. A separate wrist-camera translation of approximately 57.525 mm yielded a marker error of about 2.14 mm. These component tests measure the calibration and reconstruction paths. Temporal triangulation additionally requires correspondence to the same stationary point across views.

C.3 Skill-Guided Observation and Diagnosis

The pouring example changes the skill guidance while keeping the other loaded skills fixed. The revised skill maintains a free-arm side view during pouring, draining, and upright recovery. The task agent uses the observation arm while the pouring arm makes corrections, and the revised episode reaches task success.

The packing example reuses a trajectory-review tool that presents RGB, robot states, and policy decisions. The meta-agent inspects the box displacement and revises the packing skill to require clearance above the flaps before a turn. With the revised skill, the task agent avoids pushing the box away. The reused diagnostic tool thus supports a subsequent skill revision.

D Robot and Agent Averages on the Manipulation Subset

The original 40 robot-policy means are reaggregated from the RoboDojo Astra report's task matrix (Zhang et al., 2026d), weighting all eight tasks equally. The Simate-beta, Liber-0 Preview, and RoboDawn rows summarize the task-level Score and SR entries in table 3 (RoboDojo Team, 2026; Guo et al., 2026b). Each method is evaluated on the same task set and averaged with equal task weights. G0.5 is the strongest robot policy in the original 40-method set (43.09% SR); Simate-beta is the strongest displayed robot-policy entry in the extended comparison (50.83%). PhysEvo reaches 55.00%.

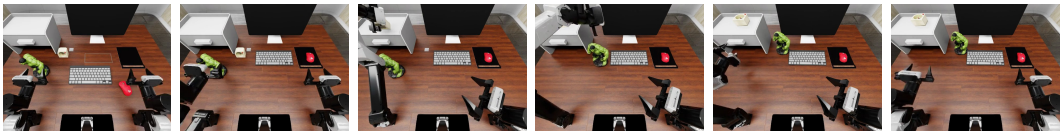
Table 5: Eight-task means for the original references and the extended comparison. Bold and underlining mark the best and second-best values.

Method	Mean score	SR (%)
PhysEvo (ours)	**62.88**	**55.00**
Simate-beta	<u>61.97</u>	<u>50.83</u>
Liber-0 Preview	54.22	46.44
RoboDawn (1-shot)	52.50	45.00
GalaxeaVLA (G0.5)	53.70	43.09
Xiaomi-Robotics-1	44.73	31.83
DM0.5	37.80	26.42
OpenWAM-α	34.10	26.25
VLAct	28.26	22.67
Meituan-Robotics-0	26.76	16.25
StarVLA-PI_v3	24.38	18.50
InternVLA-A1.5	23.84	16.00
Spatial Forcing	23.78	16.42
Hy-Embodied-0.5-VLA	23.61	15.17
StarVLA-OFT	22.68	15.08
X-VLA	22.12	15.25
Pi-05	19.73	12.00
StarVLA-GR00T	15.95	9.00
StarVLA	13.84	8.33
Xiaomi-Robotics-0	12.93	8.33
MolmoAct2	11.62	5.17
EventVLA	11.42	5.92
GalaxeaVLA (G0)	10.48	5.25
GigaWorld-Policy-0	7.87	3.09
LingBot-VLA	7.49	3.58
AHA-WAM	7.29	2.50
X-WAM	7.11	2.17
Pi-0	5.31	2.42
ABot-M0	4.61	1.67
SmolVLA (Single Task)	3.22	1.17
GROOT-N1.7	3.22	1.25
InternVLA-A1	2.96	1.33
LDA-1B	2.64	0.58
ACT (Single Task)	2.58	1.75
GO-1	2.35	1.25
Fast-WAM	1.58	0.00
RDT	0.60	0.08
Dexora-1B	0.48	0.00
H-RDT	0.42	0.00
DM0	0.31	0.08
A1	0.11	0.00
TinyVLA	0.09	0.00
OpenVLA-OFT	0.09	0.00
Spirit v1.5	0.07	0.00
Official Astra	5.96	1.25

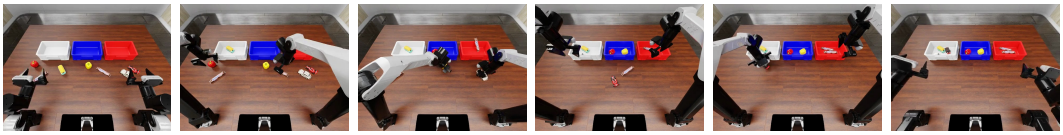
E Additional Simulation Executions

Figures 7 to 12 present execution sequences for all 42 RoboDojo simulation tasks. Each strip retains the task name and instruction and shows six frames in temporal order from left to right. The gallery follows the S01–S42 case order.

Organize Table: *Place the mouse on the mouse pad, push the keyboard into the frame, put the figurine on the stand, place the alarm clock on the drawer, then open the drawer and put all remaining miscellaneous items inside.*



Classify Objects By Language: *Put category$_1$ objects into the left basket, category$_2$ objects into the middle basket, and category$_3$ objects into the right basket, then reset the robot arm.*



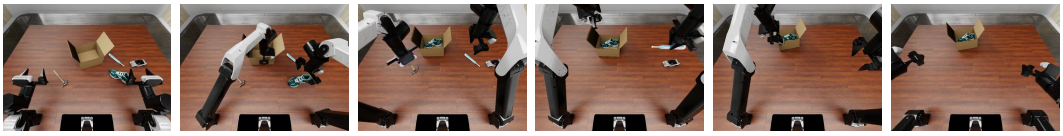
Imitate Sorting Sequence: *Observe the object placement order, remember it, then place the corresponding objects into the basket in the same order.*



Arrange Largest Number: *Arrange the numbers from left to right to form the largest possible number, and place them on the pad.*



Pack Objects Into Box: *Place all the objects into the box with their front sides facing left.*



Classify Objects: *Sort the objects by category into the three baskets.*



Build Tower: *Build a stable multi-layer tower with the wooden blocks and boards.*

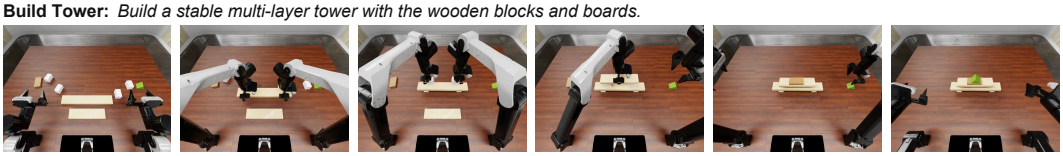
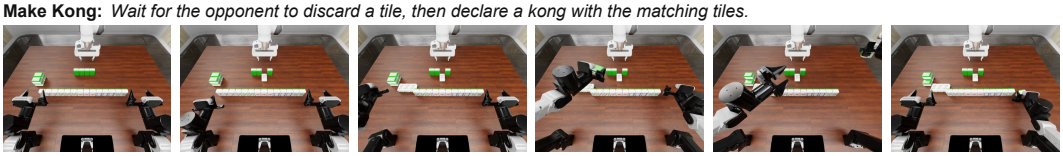
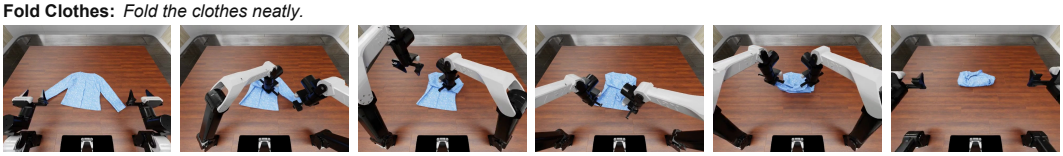


Figure 7: Simulation executions, cases S01–S07 from top to bottom. Task names and instructions appear above each strip; frames progress from left to right.

Make Kong: *Wait for the opponent to discard a tile, then declare a kong with the matching tiles.*



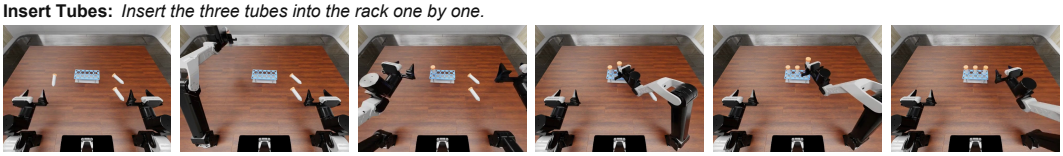
Fold Clothes: *Fold the clothes neatly.*



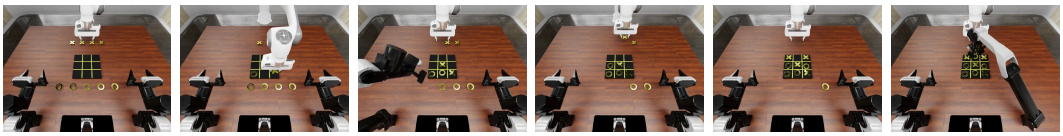
Put Bottles Into Dustbin: *Pick up the bottles and throw them into the dustbin, using handover when needed.*



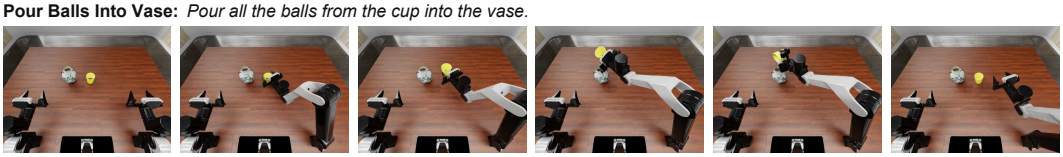
Insert Tubes: *Insert the three tubes into the rack one by one.*



Play Tic-Tac-Toe: *Play tic-tac-toe as the first player and fill the board with the opponent.*



Pour Balls Into Vase: *Pour all the balls from the cup into the vase.*



Pour Liquid Into Cup: *Pour the liquid from the bottle into the cup.*



Figure 8: Simulation executions, cases S08–S14 from top to bottom. Task names and instructions appear above each strip; frames progress from left to right.

Store Laptop And Headphones: *Hang the headphones on the headphone stand, close the laptop, then place it into the vertical laptop stand.*



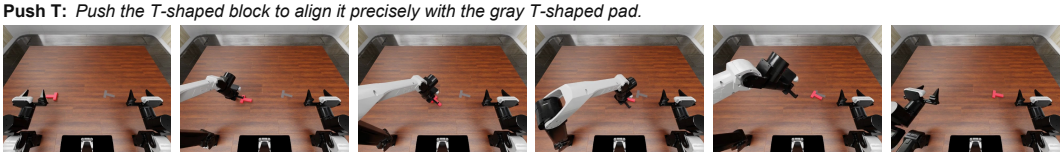
Fill Pen Holder: *Hold the pen holder with one hand, place all pens into it with the other hand, then put it back down.*



Stack Bowls: *Stack the three bowls together.*



Push T: *Push the T-shaped block to align it precisely with the gray T-shaped pad.*



Hang Mugs: *Hang all the mugs on the mug rack.*



Sweep Blocks: *Pick up the broom, hand it over to the right hand, then use the dustpan to sweep the blocks.*



Make Toast: *Pick up two slices of bread, place them into the toaster, and press the lever down.*



Figure 9: Simulation executions, cases S15–S21 from top to bottom. Task names and instructions appear above each strip; frames progress from left to right.

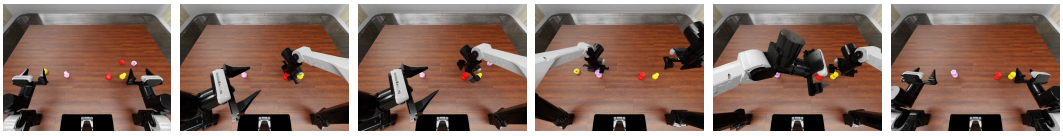
Sort Nesting Dolls By Size: *Arrange the five nesting dolls in a row from left to right, from smallest to largest.*



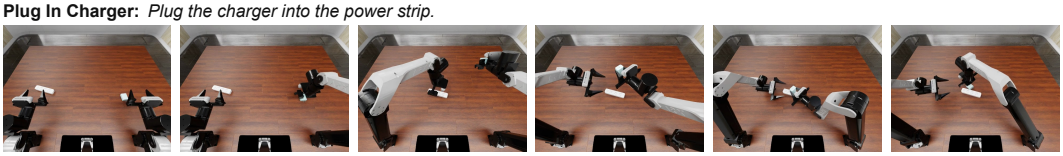
Stack Blocks: *Stack the three blocks with different textures.*



Fasten Screws: *Insert and tighten each screw into the nut of the same color.*



Plug In Charger: *Plug the charger into the power strip.*



Play Xylophone: *Pick up the mallet and strike all xylophone keys from left to right.*



Deposit Coin: *Pick up the coin from the holder and insert it precisely into the coin bank.*



Insert Key: *Pick up the key, hand it over to the other hand, insert it into the keyhole, then turn it.*

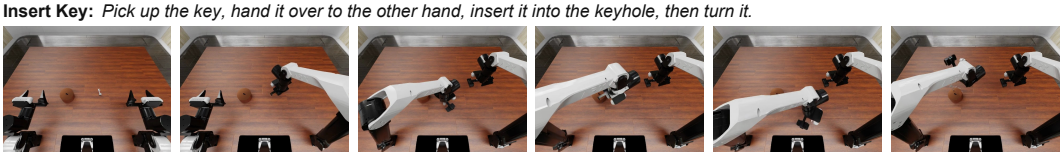
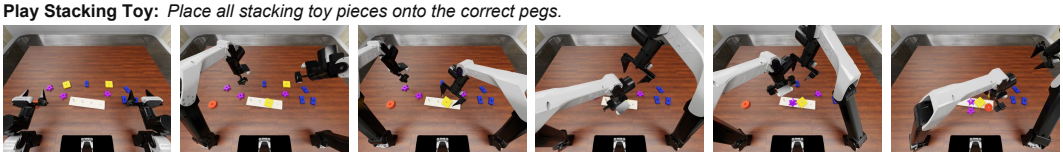


Figure 10: Simulation executions, cases S22–S28 from top to bottom. Task names and instructions appear above each strip; frames progress from left to right.

Fill Egg Holder: *Place the four eggs from the basket into the egg holder, then close the lid.*



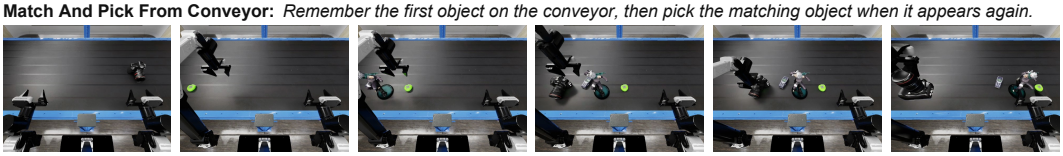
Play Stacking Toy: *Place all stacking toy pieces onto the correct pegs.*



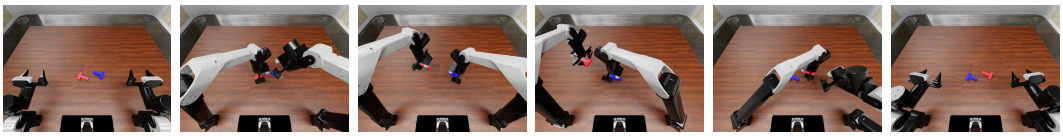
Cover Blocks: *Cover the blocks from left to right, remember their colors, then uncover them in the order: red, green, and blue.*



Match And Pick From Conveyor: *Remember the first object on the conveyor, then pick the matching object when it appears again.*



Swap T: *Pick up the two T-shaped blocks, swap their positions, and place them back with the correct orientations.*



Press By Number: *Press the two red buttons the required number of times according to the number cards, then press the blue button to confirm.*



Swap Blocks: *Swap the two blocks using the empty mat, pressing the button after each move.*

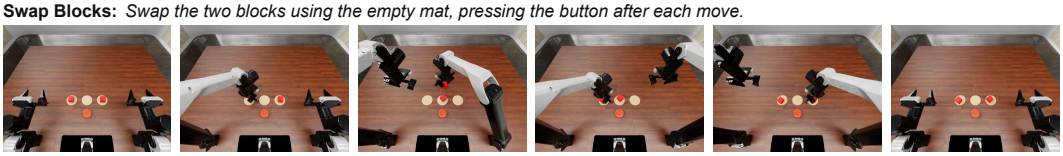
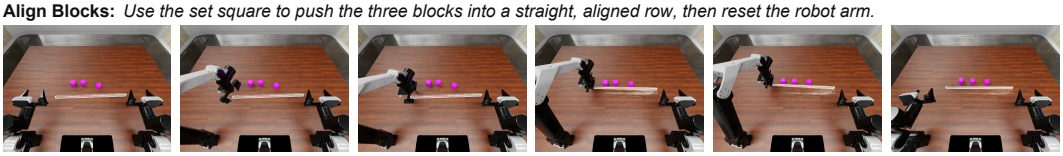


Figure 11: Simulation executions, cases S29–S35 from top to bottom. Task names and instructions appear above each strip; frames progress from left to right.

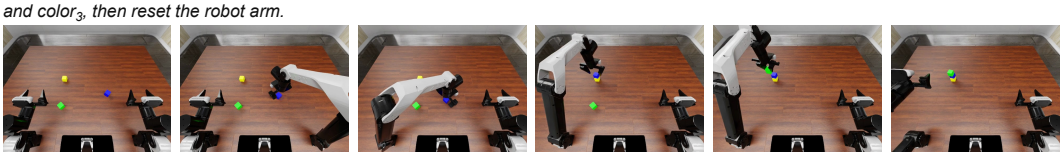
Align Blocks: *Use the set square to push the three blocks into a straight, aligned row, then reset the robot arm.*



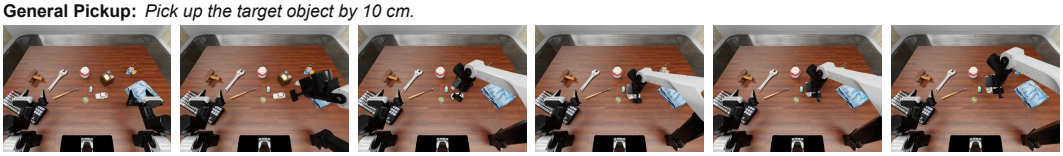
Solve Equation: *Complete the equation by selecting the correct missing number or operator and placing it on the pad, then reset the robot arm.*



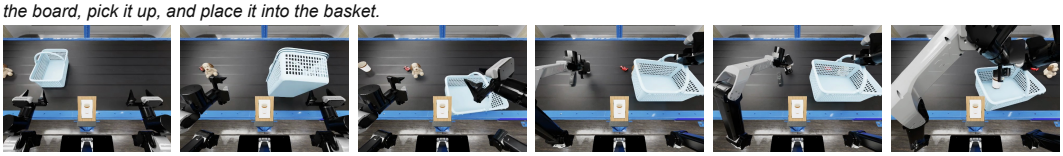
Stack Blocks By Language: *Stack the three blocks on top of each other in the order of $color_1$, $color_2$, and $color_3$, then reset the robot arm.*



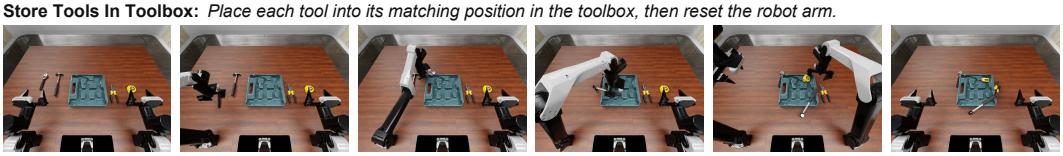
General Pickup: *Pick up the target object by 10 cm.*



Pick From Conveyor By Image: *Lift the basket more than 8 cm, identify the target object on the conveyor according to the image on the board, pick it up, and place it into the basket.*



Store Tools In Toolbox: *Place each tool into its matching position in the toolbox, then reset the robot arm.*



Pour By Language: *Pour the liquid from the first $color_1$ bottle into the first $color_2$ bowl, from the second $color_3$ bottle into the second $color_4$ bowl, and from the third $color_5$ bottle into the third $color_6$ bowl. Then reset the robot arm.*

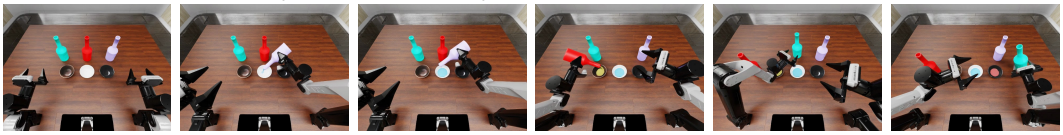


Figure 12: Simulation executions, cases S36–S42 from top to bottom. Task names and instructions appear above each strip; frames progress from left to right.

F TASK-AGENT INPUT, OUTPUT, AND TOOL FORMATS

The following blocks describe the task-agent interface in a joint-control configuration. Angle brackets and ellipses denote runtime data rather than literal argument values.

F.1 INPUT CONTEXT

The system message supplies the control contract and selected skills. Each skill is Markdown text with `name` and `description` fields. A request also includes the preceding completed tool cycle, current public context, and RGB observations. Key context fields are:

```
{
  "instruction": "<current task instruction>",
  "step": <native-step index>,
  "state": {
    "left_arm_joint_state": [<six measured joint angles>],
    "right_arm_joint_state": [<six measured joint angles>],
    "left_ee_pose": [<x>, <y>, <z>, <qw>, <qx>, <qy>, <qz>],
    "right_ee_pose": [<x>, <y>, <z>, <qw>, <qx>, <qy>, <qz>],
    "left_ee_joint_state": [<gripper command echo>],
    "right_ee_joint_state": [<gripper command echo>],
    "left_gripper_measured_joints": [<measured finger positions>],
    "right_gripper_measured_joints": [<measured finger positions>]
  },
  "memory": "<same-episode facts and unresolved questions>",
  "previous_result": <preceding tool feedback or null>,
  "budget": {
    "remaining_native_steps": <integer>,
    "model_calls_remaining": <integer>, ...
  }, ...
}
```

Named RGB inputs include `cam_head`, `cam_left_wrist`, and `cam_right_wrist`, together with selected previous-segment or requested historical views. Other context fields provide home poses, joint semantics, and camera geometry. Joint angles are in radians; positions are in metres. Gripper command echoes are distinct from measured finger positions, and neither alone establishes object retention.

F.2 MODEL OUTPUT AND FUNCTION CALLS

The model returns exactly one enabled function call. Every tool requires `assessment`, `plan`, `phase`, and updated episode `memory`, followed by tool-specific arguments. Below, `arguments` is displayed as a decoded object; the function-call transport serializes it as a JSON string.

```
{
  "type": "function_call", "call_id": "<call identifier>",
  "name": "control_arms",
  "arguments": {
    "assessment": "<observed evidence and uncertainty>",
    "plan": "<next operation>",
    "phase": "fine",
    "memory": "<updated same-episode memory>",
    "left": null,
    "right": {
      "control": "joint",
      "joints": [<q1>, <q2>, <q3>, <q4>, <q5>, <q6>],
      "gripper": null
    },
    "steps": <integer>, "settle": <integer>
  }
}
```

F.3 Tool Arguments and Execution Feedback

Table 6 lists tool-specific arguments in addition to the four common fields above. Only enabled tools can be invoked; task-agent calls do not provide shell access or hidden simulator-state queries.

Table 6: Task-agent tool arguments. Braced entries denote object fields, and square brackets denote lists.

Tool	Tool-specific arguments and semantics
`control_arms`	`left`, `right`, `steps`, `settle`. Each non-null arm has `control`, `joints`, and `gripper` fields, as shown above.
`review_frames`	`frames: [{step, camera}]`. Retrieves 1–6 same-episode RGB frames and their public poses without motion.
`project_pixels`	`points: [{pixel: [u,v], z}]`. Projects 1–32 head-view pixels onto model-specified height planes without motion.
`triangulate_point`	`step`, `feature`, `views: [{camera, pixel}]`. Matches one physical feature in 2–3 distinct current views; returns geometry estimates and diagnostics without motion.
`triangulate_temporal_point`	`step`, `camera`, `feature`, `stationary_target`, `camera_hand_empty`, `stationarity_evidence`, and `samples: [{step, pixel}]`. Uses 2–3 reviewed views from one wrist camera. Both Boolean fields must be `true`; sampling steps increase, end at the current step, and span at most 24 native steps.
`verify_completion`	No additional arguments. Advances six hold steps and checks native termination; it does not declare success.
`stop_episode`	`reason`. Stops the controller when blocked; this is not a native success signal.

Control parameters. Each non-null `joints` field contains six absolute targets in radians, not joint increments. `gripper` is a continuous opening command in $[0, 1]$. A null arm, joint vector, or gripper field holds its corresponding command. `phase` is `fine` or `transit`: motion segments use 1–6 or 1–20 native steps, respectively, with `settle` in $\{0, 1, 2\}$. Execution checks can stop a segment before the requested number of steps.

Tool-result format. The result is linked to the model call through the same `call_id`. For a motion tool, feedback includes segment status, the executed step count, and tracking measurements. The following abbreviated block displays the serialized `output` payload as a decoded object:

```
{
  "type": "function_call_output",
  "call_id": "<same call identifier>",
  "output": {
    "feedback": {
      "status": "<segment status>",
      "reason": "<status detail>",
      "executed_steps": <integer>,
      "tracking": [<per-arm tracking measurements>], ...
    },
    "step": <current native-step index>,
    "ended": <Boolean>
  }
}
```

The next request supplies updated context and images together with this feedback. Observation and geometry tools instead return their requested evidence or diagnostic information without executing native actions. Completion is read from the environment, not inferred from the model's `assessment` or declared by a tool argument.