EMBODIEDEVO: CONTINUAL ROBOT LEARNING THROUGH HYPOTHESIS-GUIDED CO-EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot foundation models provide strong visuomotor control, yet their performance can degrade when object positions or task instructions change. Further improvements often require post-training on substantial robot data, which can be costly to collect through methods such as teleoperation. Agentic harnesses can adapt around the model, but current self-evolving harnesses use robot trials inefficiently when deciding which code and skill changes to pursue. We introduce **EmbodiedEvo**, a self-evolving agentic harness that autonomously decides where to explore next and turns the resulting physical interaction into improved code and skills. EmbodiedEvo realizes this through a Fast-Slow Dual-System Architecture, in which competing code and skill hypotheses are maintained in a Hypothesis Graph. Value-of-Information Experiment Selection chooses physical experiments that can distinguish these hypotheses. Their outcomes guide Code-Skill Co-Evolution. The Slow System builds Hierarchical Memory, and Reward-Grounded Memory Learning selects effective memory according to their value for later Fast-System improvement. On RoboCasa365, EmbodiedEvo reaches **77.0%** overall success and **71.3%** on Composite-Unseen, compared with **40.1%** for the best baseline. EmbodiedEvo also reaches **86.8%** overall success on LIBERO-Pro. Beyond benchmark performance, EmbodiedEvo transfers zero-shot to real-world robot, achieving **71.3%** overall success across multiple challenging tasks.

1 INTRODUCTION

Robot foundation models have substantially advanced general-purpose robot learning, enabling transferable behavior across diverse tasks and environments. However, their performance can degrade when deployment conditions depart from prior experience, such as changes in object positions or task instructions (Zhou et al., 2025). Adapting the underlying model typically requires additional robot data collection and post-training, while gathering sufficient new interaction data for finetuning remains costly. Agentic robot harnesses provide an alternative by adapting robot behavior at the execution layer through a small number of trial executions (Liang et al., 2023; Zhang et al., 2026c; Fu et al., 2026; Ding et al., 2026; Lu et al., 2026). These harnesses use high-level reasoning to organize robot execution, and update themselves based on the outcomes of a few attempts. Some synthesize and revise executable code after observing execution failures, while others coordinate pretrained robot policies through planning, memory, and feedback. More recent systems further enable the harness to evolve from interaction, allowing skills to improve as experience accumulates.

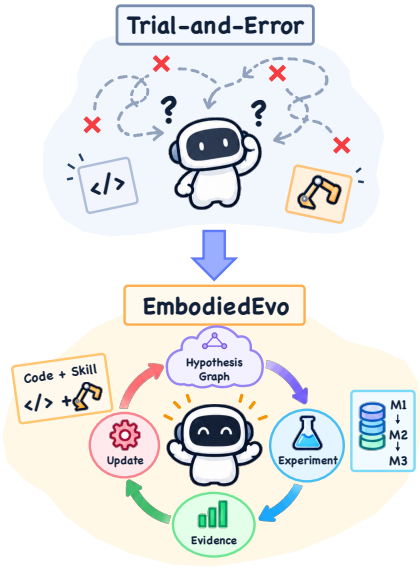


Figure 1: **The EmbodiedEvo Self-Evolution Loop.** Physical outcomes guide code and skill updates while the robot foundation model remains frozen.

Despite these advances, current self-evolving robotics harnesses still learn inefficiently from physical interaction. As the space of executable programs,

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.

behavioral skills, and their compositions grows, iterative proposal-and-execution loops can repeatedly pursue closely related improvement directions without explicitly deciding *which physical experiment would provide the most useful evidence for the next update*. Physical trials remain costly, yet existing harnesses treat them as an undifferentiated resource to consume rather than a scarce resource to actively allocate. This inefficiency is compounded by two further gaps. Code and skills are also commonly evolved through separate processes, even though long-horizon manipulation often depends on their interaction, and memory is typically accumulated for retrieval and reuse rather than learned according to whether it improves later self-evolution. As a result, existing harnesses lack a principled way to direct physical exploration toward the most informative experiments, while still coordinating code-skill improvement and retaining experience useful for future adaptation. We therefore ask how an agentic robotics harness can decide what experiment to run next, so as to efficiently transform physical interaction into continual Code-Skill Co-Evolution and reusable knowledge for adaptation to unseen scenes and long-horizon tasks.

We therefore present **EmbodiedEvo**, an agentic harness in which each physical episode informs which code or skill hypothesis to pursue next. A Fast System proposes and executes candidate experiments, each stating which hypotheses its outcome can distinguish, and revises its confidence in response to every physical outcome, while a Hypothesis Graph preserves the refinement history to inform later decisions. Within this loop, Code-Skill Co-Evolution compares each candidate pair with its individual components from the same initial states, showing that code and skills reinforce each other beyond what either achieves alone and using this evidence to choose the next physical experiment. Hierarchical Memory further distills the refinement history into reusable cognition, managed according to downstream value to the Fast System (measured by later task improvement, exploration-uncertainty reduction, and physical-experiment efficiency) through reinforcement learning rather than fixed heuristics.

Our experiments establish that a self-evolving agentic harness around a frozen robot foundation model turns physical interaction directly into reusable code and skill improvements that generalize to unseen scenes and tasks, while preserving the underlying model's local control competence.

Our main contributions share a common goal, enabling the harness to autonomously decide where to explore so that it learns from physical interaction more efficiently. **(i)** A **Fast-Slow Dual-System Architecture** realizes this principle directly, letting the Fast System select the next physical experiment using current hypothesis weights and cost and rapidly iterate code and skill updates, while the Slow System builds Hierarchical Memory that abstracts past experience to inform this choice. **(ii) Code-Skill Co-Evolution** lets code and skill updates reinforce each other rather than evolve separately, comparing each candidate pair against its individual components from the same initial states, with a Hypothesis Graph keeping these comparisons traceable to the evidence behind them. **(iii) Reward-Grounded Memory Learning** completes the loop, training the selection of memory actions according to downstream value to the Fast System. **(iv)** On RoboCasa365, EmbodiedEvo reaches state-of-the-art **77.0%** overall success, including **71.3%** on Composite-Unseen, and **86.8%** overall success on LIBERO-Pro. Beyond benchmark performance, EmbodiedEvo evolves its agent pipeline entirely in simulation and transfers it zero-shot to real-world experiments, achieving **71.3%** overall success across multiple challenging tasks.

2 Related Work

Agentic Execution Around Frozen Robot Foundation Models. Generalist vision-language-action (VLA) policies improve transfer through scale and vision-language pretraining (Brohan et al., 2022; 2023; Kim et al., 2024). Frozen-VLA agents add planning, memory, and tools around the policy. Harness VLA uses a frozen VLA as a contact-rich primitive (Zhang et al., 2026c), while Zetta evolves runtime critics, response skills, and tools at multiple time scales (Ding et al., 2026). EmbodiedEvo links physical attempts, candidate code or skill updates, and outcomes in a Hypothesis Graph before Code-Skill Co-Evolution constructs an update.

Skill Libraries and Code-Mediated Improvement. Code as Policies writes robot-control programs from language and perception APIs (Liang et al., 2023). CaP-X and Playful Agentic Robot Learning build code-skill libraries from successful executions (Fu et al., 2026; Zhang et al., 2026b), while Voyager accumulates executable skills from environment feedback (Wang et al., 2023). Other self-evolution methods revise skills or optimize an external skill and harness layer (Ju et al., 2026;

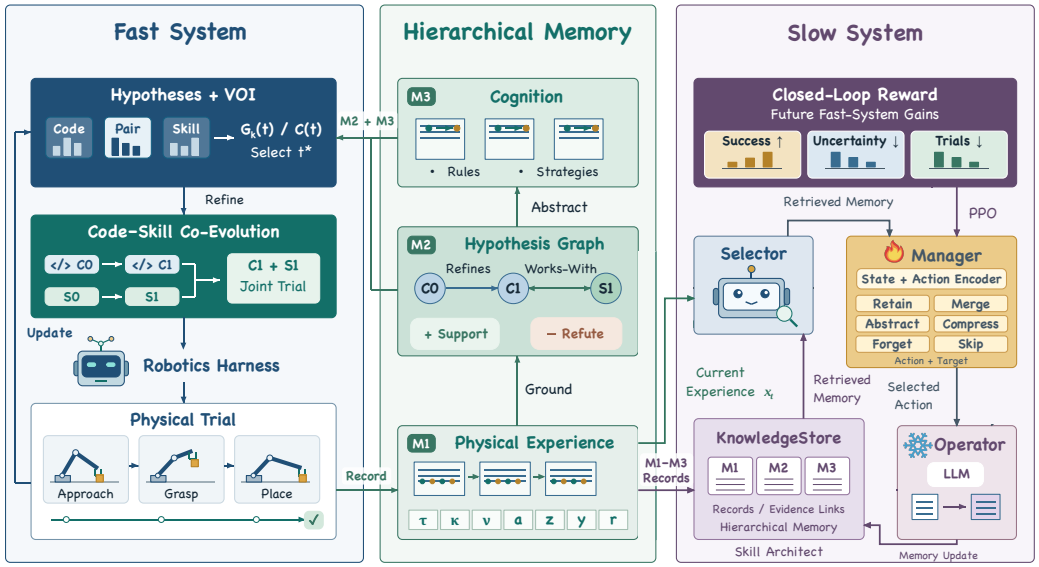


Figure 2: **Overview of EmbodiedEvo.** The Fast System uses a Hypothesis Graph to choose each physical experiment. Each result guides a joint code-skill update. The Slow System retains memories that improve later adaptation.

Wang et al., 2026), and Eureka evolves reward programs (Ma et al., 2023). EmbodiedEvo represents competing explanations in a Hypothesis Graph, uses Value-of-Information Experiment Selection to select physical experiments, and retains code and skill updates after trials from the same initial states.

Embodied Memory and Continual Learning. Embodied-RAG develops hierarchical memory for embodied retrieval and generation (Xie et al., 2024), and MemSkill learns memory-writing skills (Zhang et al., 2026a). EmbodiedEvo uses physical transitions to build reusable memory and learns from later task outcomes which memory actions support self-improvement.

3 Method

3.1 Problem Formulation

We formulate EmbodiedEvo as an agentic robotics harness that continually evolves through physical interaction. After k physical episodes, $\mathcal{A}_k$ denotes the current harness, including its executable code and skills, hypothesis set $\mathcal{H}_k$, and accumulated memory $\mathcal{M}_k$.

At each episode, EmbodiedEvo determines the next physical experiment from the candidate set $\mathcal{T}_k$ based on its current hypotheses and memory:

$$t_k^* = f(\mathcal{T}_k, \mathcal{H}_k, \mathcal{M}_k), \tag{1}$$

where f represents the experiment-selection process and t_k^* is the physical experiment performed at episode k. Executing t_k^* produces physical evidence e_k, including the execution trajectory and task outcome. The harness then evolves according to

$$\mathcal{A}_{k+1} = g(\mathcal{A}_k, t_k^*, e_k), \tag{2}$$

where g represents the evolution process that incorporates new physical evidence into the harness.

Given a budget of B physical episodes, this process yields $\mathcal{A}_B$. For N evaluation episodes with binary task outcomes $b_i(\mathcal{A}_B) \in \{0, 1\}$, the task success rate $R_{\text{task}}(\mathcal{A}_B)$ is the fraction of successful evaluations. Our objective is to maximize this task success rate:

$$R_{\text{task}}(\mathcal{A}_B) = \frac{1}{N} \sum_{i=1}^{N} b_i(\mathcal{A}_B), \qquad \max_{\mathcal{A}_B} R_{\text{task}}(\mathcal{A}_B). \tag{3}$$

3.2 FAST SYSTEM: RAPID CODE-SKILL CO-EVOLUTION

Hypothesis Graph and Confidence. Each selected experiment produces evidence about possible code and skill changes. To keep that evidence tied to hypotheses as they evolve, the Hypothesis Graph maintains code and skill hypotheses as nodes, with a directed REFINES edge linking each hypothesis to its revision. The Fast System uses each trajectory and outcome to generate new code or skill hypotheses and to refine existing hypotheses when the evidence supports a more specific version. To track how strongly physical experiments support each hypothesis h_i, all hypotheses begin with the same $\mathrm{Beta}(\alpha, \beta)$ prior on their support rate (Gelman et al., 2013), where α and β are prior support and refutation counts. Let $n_{\mathrm{sup},i}$ count physical experiments supporting h_i and $n_{\mathrm{ref},i}$ count outcomes inconsistent with h_i. Matched trials update each tested hypothesis from its success-rate difference, holding initial states and all other calls and their order constant. The posterior mean confidence of h_i is:

$$c_i = \frac{n_{\mathrm{sup},i} + \alpha}{n_{\mathrm{sup},i} + n_{\mathrm{ref},i} + \alpha + \beta}. \tag{4}$$

The mean confidence captures support for each hypothesis; choosing the next experiment also requires knowing where that support remains uncertain. Let D_k be the matched outcomes observed after k physical episodes and $q_i = P(\theta_i > 1/2 \mid D_k)$, where θ_i is the support rate of h_i. Multiple hypotheses can jointly contribute to task success. We measure uncertainty over their support as:

$$\mathcal{H}_{\mathrm{exp}}(D_k) = \sum_{h_i \in \mathcal{H}} h_2(q_i), \quad h_2(q) = -q \log_2 q - (1-q) \log_2 (1-q). \tag{5}$$

A matched trial provides initial evidence for each new hypothesis before its subsequent experiments are selected adaptively.

Value-of-Information Experiment Selection. Physical outcomes update the confidence of each hypothesis, revealing where evidence is still needed. To decide which uncertainty to address within its limited interaction budget, the Fast System constructs experiments $t \in \mathcal{T}$ that evaluate code and skill hypotheses through matched trials. Each experiment declares the subset $S_t \subseteq \mathcal{H}$ whose confidence its outcome can update. Given the evidence D_k, Value-of-Information Experiment Selection measures the expected reduction in exploration uncertainty from the possible outcomes e of experiment t:

$$G_k(t) = \mathcal{H}_{\mathrm{exp}}(D_k) - \mathbb{E}_{e \sim P(e \mid D_k, t)} \big[\mathcal{H}_{\mathrm{exp}}(D_k \cup \{e\}) \big]. \tag{6}$$

Appendix D gives the outcome probabilities. Let T_t, $N_{\mathrm{step},t}$, $C_{\mathrm{GPU},t}$, and $C_{\mathrm{reset},t}$ denote elapsed time, simulator steps, GPU cost, and reset cost, with weights w_τ, w_s, w_g, and w_r. The physical-experiment cost is $C(t) = w_\tau T_t + w_s N_{\mathrm{step},t} + w_g C_{\mathrm{GPU},t} + w_r C_{\mathrm{reset},t}$, the selection score is $\mathrm{VOI}_k(t) = G_k(t)/C(t)$, and the next experiment is $t_k^* = \arg\max_{t \in \mathcal{T}} \mathrm{VOI}_k(t)$. After executing t_k^*, the Fast System updates the evaluated hypotheses from matched outcomes. The resulting posteriors determine both exploration uncertainty and the next experiment ranking. Algorithm 1 in Appendix D gives the complete procedure. The experimental configuration is provided in Appendix E. With thresholds ϵ_H and ϵ_{VOI}, broad exploration ends when $\mathcal{H}_{\mathrm{exp}} < \epsilon_H$, $\max_t \mathrm{VOI}_k(t) < \epsilon_{\mathrm{VOI}}$, the rollout budget is exhausted, or every hypothesis reaches the evaluation cap $n_{\max}$.

Code-Skill Co-Evolution. The selected experiment provides evidence for deciding what to update. An improvement may come from code, skills, or their interaction. Code-Skill Co-Evolution separates code, skill, and joint effects by evaluating code hypothesis C_i, skill hypothesis S_j, and their joint execution from the same initial states. Let $R(C_i, S_j)$, $R(C_i)$, and $R(S_j)$ denote the measured task success rates of joint, code-only, and skill-only execution, respectively. Their measured joint gain is:

$$w_{ij} = R(C_i, S_j) - \max\{R(C_i), R(S_j)\}. \tag{7}$$

Positive joint gain indicates a higher success rate for joint execution in the matched trials. The Slow System records this relation as a bidirectional WORKS-WITH edge in the Hypothesis Graph, prioritizing the strongest relations while directed revision edges retain the refinement history. The Fast System evaluates new and revised hypotheses in subsequent experiments and retains executable improvements for the next round. Each outcome enters Hierarchical Memory and changes what the Fast System will explore next. Figure 3 shows how physical trials turn candidate code-skill connections into retained WORKS-WITH relations.

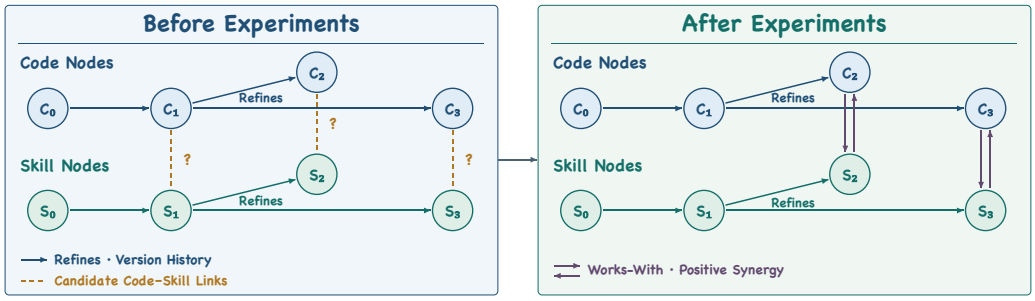


Figure 3: **Hypothesis Graph Updates from Physical Trials. Before Experiments (left).** Directed REFINES edges show how code and skill hypotheses have evolved, while dashed links mark code-skill relations awaiting evidence. **After Experiments (right).** Solid bidirectional WORKS-WITH edges identify relations supported by measured positive joint gain.

3.3 SLOW SYSTEM: HIERARCHICAL MEMORY AND REWARD-GROUNDED MEMORY LEARNING

Hierarchical Memory. Each experiment produces evidence that later decisions need at different levels of detail. Hierarchical Memory carries this evidence forward at three levels: M1 records physical execution, M2 maintains code and skill hypotheses in the Hypothesis Graph, and M3 retains reusable cognition distilled from repeated evidence.

For an episode, let τ be the robot execution trajectory, κ the invoked tools, ν the time steps at which the VLA is invoked, a the executed actions, z the telemetry, y the physical outcome, and r the task reward. The M1 experience record is:

$$\mathcal{M}_1 = \{(\tau, \kappa, \nu, a, z, y, r)\}. \tag{8}$$

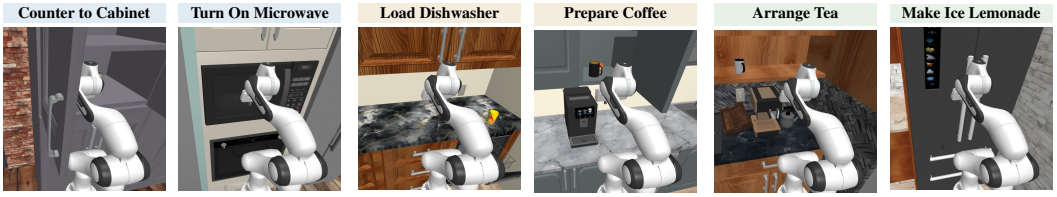
M1 automatically records every episode and supplies matched evidence for M2 count and relation updates under the Fast-System rules. Across episodes, the Slow System abstracts recurring M1 and M2 evidence into M3. The hierarchy is:

$$\mathcal{M}_1 \to \mathcal{M}_2 \to \mathcal{M}_3. \tag{9}$$

Before the next physical episode, the Fast System retrieves M2 hypotheses and M3 cognition to guide its candidate experiments.

Reward-Grounded Memory Learning. For the memory hierarchy to guide later experiments, the Slow System learns how to transform incoming evidence into reusable knowledge. Reward-Grounded Memory Learning bases that decision on later Fast-System improvement. For current experience x_t, the Slow System retrieves N_{ret} relevant records $\{m_{t,\ell}\}_{\ell=1}^{N_{\text{ret}}}$ from M1, M2, and M3, including hypothesis confidence and evidence links. We encode experience, retrieved records, and action descriptions with the frozen Qwen3-Embedding-0.6B encoder $e(\cdot)$ (Zhang et al., 2025). The trainable context function f_θ^{ctx} aggregates these embeddings into Slow-System state h_t. Each candidate in $\mathcal{S} = \{s_1, \ldots, s_K\}$ specifies an applicable action and target record or evidence group. The action space comprises Retain, Merge, Abstract, Compress, Forget, and Skip. Let u_i represent candidate s_i, and let $z_{t,i}$ measure compatibility between h_t and u_i. The Slow-System policy is $\pi_\theta(i \mid h_t) = \text{softmax}(z_t)_i$. Here $z_t = (z_{t,1}, \ldots, z_{t,K})$ collects the compatibility scores and $[\cdot,\cdot]$ denotes concatenation. $\text{desc}(s_i)$ describes the action and target of candidate s_i. The trainable skill function f_θ^{skill} maps each encoded description to u_i, and the trainable scoring function f_θ^{score} maps $[h_t, u_i]$ to $z_{t,i}$. The operator uses frozen Qwen3.5-122B-A10B (Qwen Team, 2026) to update the retrieval index or M3 according to the selected candidate, preserving raw M1 records and M2 evidence updates. Appendix C specifies the actions.

Closed-Loop Reward and RL Training. The value of each memory action becomes visible through its effect on later Fast-System experiments. At memory decision step t, let R_t estimate R_{task} for the current harness, H_t denote the exploration uncertainty $\mathcal{H}_{\text{exp}}$, and C_{t+1} be the realized cost of the next physical experiment under $C(\cdot)$. We compare success rates under the same evaluation protocol and compute entropy changes over the same hypothesis set. With reward weights λ_1,



(a) Atomic-Seen **(b)** Composite-Seen **(c)** Composite-Unseen

Figure 4: **RoboCasa365 Task Suites. (a) Atomic-Seen** contains individual manipulation tasks observed during pretraining. **(b) Composite-Seen** combines multiple skills into tasks represented in pretraining. **(c) Composite-Unseen** combines skills into tasks absent from pretraining.

λ_2, and λ_3, the reward following the next experiment is:

$$r_t^{\text{mem}} = \lambda_1(R_{t+1} - R_t) + \lambda_2(H_t - H_{t+1}) - \lambda_3 C_{t+1}. \tag{10}$$

For a horizon of L memory decisions and discount factor $\gamma \in (0, 1]$, the discounted return is $J_t^{\text{mem}} = \sum_{j=0}^{L-1} \gamma^j r_{t+j}^{\text{mem}}$. The Slow-System policy is trained with PPO (Schulman et al., 2017) to maximize expected discounted return. Training settings are given in Appendix F. An architect can refine action instructions from recurring experience. Thus each episode can change both the hypotheses considered by the Fast System and the memories available when the next experiment is selected.

4 Experiments

Our experiments address three questions about generalization, zero-shot sim-to-real transfer, and learning efficiency. **Q1**: How does EmbodiedEvo perform on RoboCasa365 and LIBERO-Pro, especially under cross-task, cross-scene, and out-of-distribution generalization? **Q2**: Can EmbodiedEvo's agentic harness transfer zero-shot from simulation to a real robot? **Q3**: Which components of EmbodiedEvo improve success and learning efficiency?

4.1 Experimental Setup

Benchmark and Backbone. We evaluate on two simulated benchmarks. We select RoboCasa365 to evaluate generalization across household scenes and task compositions, and LIBERO-Pro to examine changes in task goals and object layouts. RoboCasa365 (Nasiriany et al., 2026) extends RoboCasa's diverse household scenes and compositional manipulation tasks (Nasiriany et al., 2024) to large-scale mobile manipulation, with Atomic-Seen, Composite-Seen, and Composite-Unseen task suites. LIBERO-Pro (Zhou et al., 2025) extends LIBERO (Liu et al., 2023) with controlled perturbations. We evaluate instruction-redirection (T) and position-swap (S) perturbations. Backbone settings are given in Appendix I. On both benchmarks, we evaluate each task with 10 random seeds and 10 trials per seed. Evolution uses 10 separate seeds, while all methods are evaluated on another 10 mutually disjoint seeds.

Baselines. We compare against end-to-end frozen-VLA policies (OpenVLA (Kim et al., 2024), π_0 (Black et al., 2025), $\pi_{0.5}$ (Physical Intelligence et al., 2025), π_{RLinf} (Yu et al., 2026), RLDX-1 (Kim et al., 2026), NORA (Hung et al., 2025), MolmoAct (Lee et al., 2025), X-VLA (Zheng et al., 2026), AtomVLA (Sun et al., 2026)), a world-model baseline (WorldDreamer (Wang et al., 2024)), a leaderboard entry (Xiaomi-Robotics-1), and code-as-policy and harness agents built on a frozen VLA (CaP-X (Fu et al., 2026), RATS (Zhang et al., 2026b), Harness VLA (Zhang et al., 2026c), and Zetta (Ding et al., 2026)).

Metrics. On **RoboCasa365**, we report success on the Atomic-Seen, Composite-Seen, and Composite-Unseen (out-of-distribution) suites together with the overall success rate. On **LIBERO-Pro**, we report success on each task-family/perturbation cell together with the overall success rate.

4.2 Generalization on RoboCasa365 and LIBERO-Pro

Composite-Unseen provides the most stringent RoboCasa365 evaluation: familiar skills must support task compositions absent from pretraining. With the robot foundation model frozen, Embod-

Table 1: **RoboCasa365 Success Rates.** Values are percentages. "–" denotes a suite without a reported result for the corresponding method.

Method	Atomic-Seen	Composite-Seen	Composite-Unseen	Overall
π_0	34.6	6.1	1.1	14.8
$\pi_{0.5}$	39.6	7.1	1.2	16.9
RLDX-1	60.0	21.3	5.0	30.0
WorldDreamer	66.3	26.7	9.0	35.3
Xiaomi-Robotics-1	80.2	57.1	32.1	57.4
Harness VLA	91.3	55.8	40.1	63.6
EmbodiedEvo (Ours)	**94.4**	**63.1**	**71.3**	**77.0**

iedEvo leads the baselines on all three suites and reaches **77.0%** overall success (Table 1). Figure 4 illustrates the three task suites.

On Composite-Unseen, EmbodiedEvo reaches **71.3%** success, compared with 40.1% for the best baseline (Harness VLA). This separation shows where the harness matters most: when previously learned behaviors must be reorganized for a new task.

LIBERO-Pro evaluates generalization under changed instructions and object positions. EmbodiedEvo reaches **86.8%** overall success versus 72.1% for Harness VLA and leads Harness VLA in all eight perturbation cells (Table 2). The largest gaps occur under position-swap (S) in SPATIAL and LIBERO-10, indicating that EmbodiedEvo adapts execution when object layouts change. Appendix A provides examples of both perturbations. *Together, these results answer **Q1**: EmbodiedEvo generalizes to unseen task compositions and to changed instructions and object positions.*

Table 2: **LIBERO-Pro Success Rates.** The table aggregates success rates (%) across SPATIAL, OBJECT, GOAL, and LIBERO-10 under instruction-redirection (T) and position-swap (S) perturbations. "–" denotes a cell not applicable to the corresponding method.

Method	Spat-T	Spat-S	Obj-T	Obj-S	Goal-T	Goal-S	L10-T	L10-S	Overall
OpenVLA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
π_0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.3
$\pi_{0.5}$	1.0	20.0	1.0	17.0	2.0	38.0	1.0	8.0	11.0
MolmoAct	0.0	0.0	0.0	6.0	0.0	0.0	6.0	0.0	1.5
NORA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
X-VLA	0.0	0.0	8.0	2.0	9.0	1.0	10.0	0.0	3.8
AtomVLA	1.0	16.0	0.0	10.0	11.0	2.0	9.0	1.0	6.3
π_{RLinf}	42.0	59.0	71.0	78.0	45.0	42.0	49.0	14.0	50.0
CaP-X	14.0	12.0	18.0	22.0	17.0	26.0	–	–	18.2
RATS	31.0	29.0	63.0	61.0	36.0	43.0	–	–	43.8
Harness VLA	81.0	69.0	94.0	91.0	75.0	66.0	52.0	49.0	72.1
Zetta	–	–	–	–	**92.5**	89.0	63.0	40.0	71.1
EmbodiedEvo (Ours)	**100.0**	**93.0**	**100.0**	**96.0**	78.0	**90.0**	**66.0**	**71.0**	**86.8**

4.3 Zero-Shot Sim-to-Real Transfer of the Agentic Harness

We next examine whether the gains in simulation carry into physical manipulation by directly deploying EmbodiedEvo's agentic harness, including code and skills evolved in simulation, on a physical SO-101 arm (Cadene et al., 2026). SmolVLA (Shukor et al., 2025) serves as the robot foundation model. We compare SmolVLA alone with the same backbone under EmbodiedEvo's agentic harness. For each task, we collect 50 human-teleoperated demonstrations and fully fine-tune SmolVLA for 20,000 steps on 4× NVIDIA H200 GPUs. The model remains frozen during all trials. We run 30 trials per condition on each task. The remaining training settings are given in Appendix B.

These five highly challenging real-world tasks span three categories: *(1) Multi-Step Manipulation (cake stacking and towel folding), (2) Semantic and Arithmetic Reasoning (Oreo placement and poker-chip selection), and (3) Precision Grasping (lifting glasses at the bridge).*

The tasks require multi-step execution, grounding object descriptions in the scene, and precise control at a narrow grasp point.

In the poker-chip task, the robot must push out the chip whose value multiplied by five equals 100. The target is the chip marked 20 because $20 \times 5 = 100$. We shuffle the chips' order and orientation before every trial. Figure 5 illustrates the first three tasks. Table 3 reports success on all five. The largest gain occurs on poker-chip selection, where the agent pipeline identifies the correct chip from the arithmetic instruction before directing SmolVLA in a randomized layout.

*With SmolVLA frozen, zero-shot transfer of EmbodiedEvo's agentic harness raises overall real-robot success from 46.0% to 71.3%, answering **Q2**: improvements learned by the harness in simulation carry over to physical manipulation.*

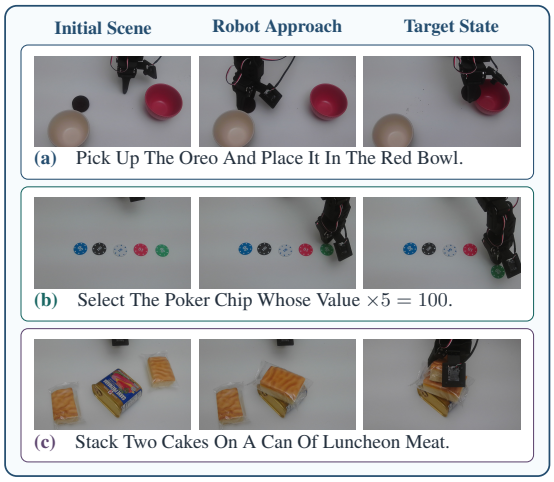


Figure 5: **Representative Real-World Evaluation Tasks.** Initial and target states.

Table 3: **Real-Robot Success Rates.** Success over 30 trials per task on the real robot.

Task	SmolVLA	**EmbodiedEvo**
Oreo to Red Bowl	70.0%	86.7%
Poker Chip Selection ($\times 5 = 100$)	10.0%	90.0%
Stack Two Cakes on a Can of Luncheon Meat	63.3%	76.7%
Glasses Bridge Grasp	60.0%	56.7%
Towel Folding	26.7%	46.7%
Overall	**46.0%**	**71.3%**

4.4 Ablation Study

We further explore what drives EmbodiedEvo's gains from physical interaction (**Q3**). The ablations isolate how Code-Skill Co-Evolution, Value-of-Information Experiment Selection, and Reward-Grounded Memory Learning affect success and learning efficiency. For the code-skill comparison, we hold the frozen VLA, task suites, physical-experiment budget, and evaluation protocol fixed. Code-only Evolution updates executable code while keeping skills fixed, Skill-only Evolution updates skills while keeping executable code fixed, and Code-Skill Co-Evolution updates both code and skills.

Figure 6 shows that Code-Skill Co-Evolution improves success across RoboCasa365, with the clearest advantage on Composite-Unseen tasks. On this suite, Code-Skill Co-Evolution reaches 71.3% success versus 55.6% for Skill-only Evolution. Notably, Code-only Evolution outperforms Skill-only Evolution on Atomic-Seen, but the order reverses on Composite-Unseen. This reversal suggests that a change effective for familiar atomic tasks may not carry over to unseen compositions. Joint evolution leads on both suites, indicating that code and skills are more useful when updated together.

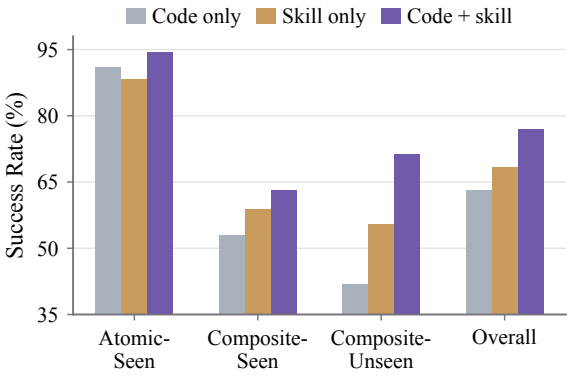


Figure 6: **Code-Skill Co-Evolution Ablation.** Bars show RoboCasa365 success rates.

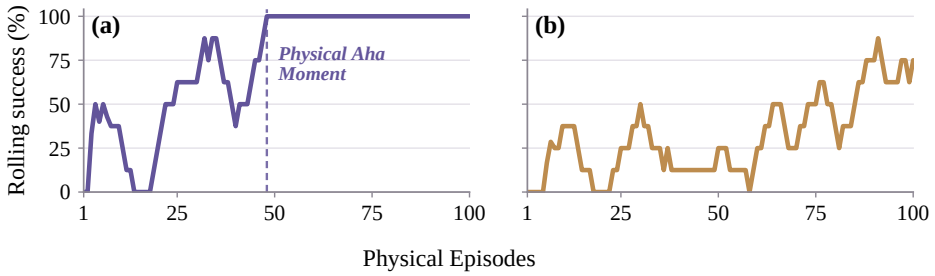


Figure 7: **Representative LIBERO-Pro Learning Curves.** Eight-episode rolling success on SPAT-T over 100 physical episodes. **(a)** Value-of-Information Experiment Selection reaches sustained 100% at episode 48 (dashed line). **(b)** Random Selection continues to fluctuate.

Table 4: **Learning Efficiency.** EmbodiedEvo is compared with Random Exploration.

Method	Overall Success @ 50 Ep. ↑	Learning AUC ↑
Random Exploration	27.1	0.309
EmbodiedEvo (Ours)	**72.9**	**0.635**

Learning Efficiency. Table 4 compares EmbodiedEvo with Random Exploration after 50 physical episodes. EmbodiedEvo reaches 72.9% overall success versus 27.1%, and its learning AUC is 0.635 versus 0.309. A ***Physical Aha Moment*** for agentic robotics occurs when an agent identifies the critical physical state variable needed to return a VLA to its in-domain region (Ding et al., 2026). In a separate representative LIBERO-Pro SPAT-T case, Figure 7 shows the observable marker of this moment: Value-of-Information Experiment Selection reaches 100% eight-episode rolling success within 50 physical episodes and remains there. Random Selection of code and skill experiments continues to fluctuate through episode 100 and ends around 75.0% rolling success, suggesting that more physical episodes are needed for a comparable ***Physical Aha Moment***.

Reward-Grounded Memory Learning. Table 5 compares retrieval-based selection, PPO with task reward alone, and the full method. The ablation settings are detailed in Appendix C. Composite-Unseen success rises from 61.7% and 65.2% to 71.3%. Learning AUC rises from 0.556 and 0.589 to 0.635. *Together, the ablations resolve **Q3**: Code-Skill Co-Evolution, hypothesis-guided exploration, and Reward-Grounded Memory Learning improve success and learning efficiency.*

Table 5: **Reward-Grounded Memory Learning Ablation.** Composite-Unseen and learning AUC on RoboCasa365.

Variant	Composite-Unseen ↑	Learning AUC ↑
Retrieval-Based	61.7%	0.556
PPO + Task Reward Only	65.2%	0.589
EmbodiedEvo (Ours)	**71.3%**	**0.635**

5 Conclusion

EmbodiedEvo shows that a self-evolving agentic harness can gain lasting value from each physical trial by using the outcome to select the next experiment, refine code and skills together, and retain experience for later adaptation. Across simulated benchmarks and real-robot tasks, EmbodiedEvo improves generalization and learning efficiency, and transfers the simulation-evolved harness to physical execution. More broadly, continual robot learning can take place in the harness around a frozen robot foundation model when physical evidence shapes both the next experiment and the knowledge carried into future tasks.

Limitations. Code-as-policy and agentic harness methods learn executable behavior from physical trajectories without updating the robot foundation model's parameters. Future work could use trajectories selected by the harness for online model updates, allowing code, skills, and the model to improve together.

AI USE STATEMENT

We used GPT to assist with drafting and polishing manuscript text and checking LaTeX formatting. The authors reviewed the resulting text and take responsibility for the final manuscript.

ETHICS STATEMENT

This work evaluates EmbodiedEvo on RoboCasa365 and LIBERO-Pro in simulation and on a tabletop SO-101 arm across five object-manipulation tasks. Simulation allows code and skill updates to be examined before transfer to hardware. The experiment-selection cost accounts for simulator steps and GPU use, and physical outcomes are reported separately from simulation results. Human teleoperation supplies demonstrations for fine-tuning SmolVLA, while the experiments evaluate robot behavior rather than people. EmbodiedEvo updates the agentic harness around a robot foundation model that remains frozen during evaluation. The paper specifies the tasks, training hardware, and physical trial counts to make the study's scope and resource use clear.

Before any public release, demonstration videos and execution logs will be reviewed for incidental identifying information and shared only with appropriate permissions. The reported evaluation concerns the specified robot and task settings, so reuse of evolved code and skills on another platform calls for validation in that setting.

REPRODUCIBILITY STATEMENT

We provide implementation and evaluation details throughout the paper and appendix. The Fast-System procedure is specified in Algorithm 1, with the numerical configuration, interaction budget, and online reporting procedure reported in Appendix E. The Slow-System memory hierarchy, memory actions, and controlled ablation protocol are detailed in Appendix C. PPO training hyperparameters are reported in Appendix F. Real-robot training settings are provided in Appendix B. The experimental setup describes the benchmarks and random seeds. Appendix H specifies the matched initial-state protocol used for code, skill, and joint comparisons. We will provide anonymized source code, configuration files, and experiment scripts as supplementary material.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Remi Cadene, Simon Alibert, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, et al. Lerobot: An open-source library for end-to-end robot learning. In *International Conference on Learning Representations*, volume 2026, pp. 122398–122417, 2026.

Xin Ding, Liang Mi, Mingzhe Huang, Zixuan Wang, Chao Zhang, Zixu Hao, Fu Chen, Xiangyu Li, Yikai Zheng, Yaoyu Guo, et al. Zetta ζ: An efficient closed-loop embodied harness for self-evolving physical intelligence. *arXiv preprint arXiv:2608.16590*, 2026.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 3 edition, 2013. ISBN 9781439840955.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

Ruofei Ju, Xinrui Wang, Xin Ding, Yifan Yang, Hao Wu, Shiqi Jiang, Qianxi Zhang, Hao Wen, Xiangyu Li, Weijun Wang, et al. Embodiskill: Skill-aware reflection for self-evolving embodied agents. *arXiv preprint arXiv:2605.10332*, 2026.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026.

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

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

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)*, pp. 9493–9500. IEEE, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. Aspire: Agentic/skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023. URL https://arxiv.org/abs/2310.12931.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 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.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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

Xiaoquan Sun, Zetian Xu, Chen Cao, Zonghe Liu, Yihan Sun, Jingrui Pang, Ruijian Zhang, Zhen Yang, Kang Pang, Dingxin He, et al. Atomvla: Scalable post-training for robotic manipulation via predictive latent world models. *arXiv preprint arXiv:2603.08519*, 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.

Peidong Wang, Zhiming Ma, Ying Chang, Xufang Luo, Xiaocui Yang, Shi Feng, Yuqing Yang, and Dongsheng Li. Self-evolving embodied agents via skill-harness evolution. *arXiv preprint arXiv:2608.11350*, 2026.

Xiaofeng Wang, Zheng Zhu, Guan Huang, Boyuan Wang, Xinze Chen, and Jiwen Lu. Worlddreamer: Towards general world models for video generation via predicting masked tokens. *arXiv preprint arXiv:2401.09985*, 2024.

Quanting Xie, So Yeon Min, Pengliang Ji, Yue Yang, Tianyi Zhang, Kedi Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-rag: General non-parametric embodied memory for retrieval and generation. *arXiv preprint arXiv:2409.18313*, 2024.

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

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026a.

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*, 2026b.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

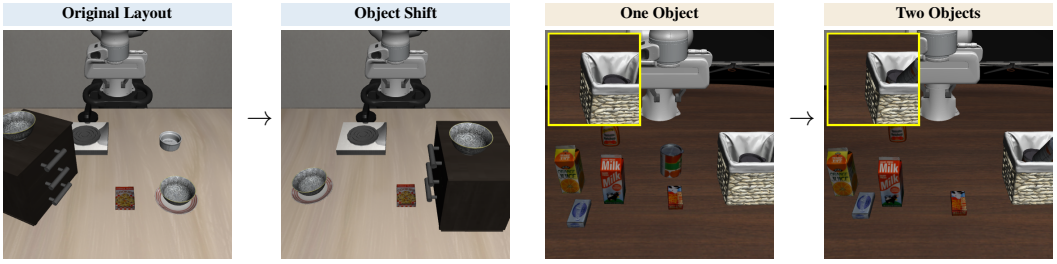
Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026c.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

APPENDIX

A LIBERO-PRO PERTURBATION EXAMPLES



(a) Position Perturbation **(b)** Task Perturbation

Figure A.1: **Representative LIBERO-Pro Perturbation Settings. (a) Position Perturbation** relocates manipulable objects within feasible spatial bounds, illustrated by moving the cabinet, bowls, and cookie box. **(b) Task Perturbation** redefines task logic and target states: the depicted task requires two objects in the goal basket rather than one. Yellow boxes indicate the goal basket (Zhou et al., 2025).

B REAL-ROBOT TRAINING CONFIGURATION

Table B.1 lists the complete SmolVLA fine-tuning configuration used for the real-robot comparison.

Table B.1: **SmolVLA Fine-Tuning Configuration.** Settings shared by both real-robot conditions before the agent pipeline is applied.

Parameter	Setting
Hardware	4× NVIDIA H200
Parallelism	Distributed data parallel (`--multi_gpu --num_processes=4`)
Precision	bfloat16
Fine-tuning scope	Full model
Freeze vision encoder	`freeze_vision_encoder=false`
Train expert only	`train_expert_only=false`
Batch size per GPU	64
Effective batch size	256
Training steps	20,000
Learning-rate schedule	Cosine decay
Warmup steps	1,000
Decay steps	20,000

The training loss decreases from 0.65 to 0.013, the gradient norm stabilizes near 0.2, and wall-clock training time is approximately 4 hours 7 minutes. Both real-robot conditions use this backbone. EmbodiedEvo adds its agent pipeline during evaluation.

C HIERARCHICAL MEMORY SCHEMA

EmbodiedEvo records an episode as a sequence of physical transitions. Hierarchical Memory stores each raw transition as M1 experience, organizes executable code and skill hypotheses as M2, and periodically extracts reusable cognition as M3. The M1 record contains tool arguments, a pre-state, a post-state, execution latency, raw tool output, a physical-outcome label, and task reward. A state can contain end-effector pose, gripper state, task progress, VLA history, and simulator-only fields. The planner accesses tool-interface fields, while simulator-only fields remain in the memory record. The resulting hierarchy links episodes, physical states, hypotheses, outcomes, and content-addressed artifacts. Table C.1 summarizes the component responsibilities, and Table C.2 defines the Slow-System action space.

Table C.1: **Hierarchical Memory Responsibilities.** Components, inputs, and responsibilities in the Fast-Slow loop.

Component	Input	Responsibility
Manager	Current experience and retrieved M1, M2, M3 records	Select an applicable action and target using the PPO-trained policy.
Selector	Current experience and indexed records	Retrieve relevant M1, M2, and M3 records with evidence links.
Operator	Selected action, target, and source records	Generate and apply a structured update using a frozen LLM.
KnowledgeStore	Experience, hypothesis states, and cognition	Store M1–M3 records, evidence links, and retrieval indices.
M1: Raw Experience	Execution trajectory and outcome	Automatically append every trial record and preserve the raw evidence.
M2: Hypothesis State	Matched M1 evidence	Maintain hypothesis counts and graph relations under the Fast-System update rules.
M3: Reusable Cognition	Selected action and source evidence	Store reusable knowledge with conditions, strategies, scope, and evidence references.

Table C.2: **Slow-System Memory Actions.** Each action specifies how experience contributes to reusable knowledge.

Action	Purpose	Target
Retain	Add valuable experience to the long-term retrieval index.	M1 retrieval index
Merge	Incorporate new evidence into existing cognition while preserving applicable conditions.	Existing M3 item
Abstract	Create new reusable cognition from recurring experience and hypothesis evidence.	M1 and M2 evidence group $\rightarrow$ M3
Compress	Shorten the representation while preserving meaning and evidence links.	Indexed representation or M3 item
Forget	Remove an outdated or superseded entry from active retrieval while preserving source evidence.	Retrieval index or active M3 entry
Skip	Leave the retrieval index and M3 unchanged.	No update

Each candidate s_i identifies an applicable action and target record or evidence group. The manager scores candidates using the current experience and retrieved records, including task context, hypothesis confidence, and evidence links. Skip is always available. The operator applies the selected action while preserving raw M1 records and evidence-driven M2 updates. After the operator applies a memory action, the Fast System selects and executes the next experiment using the updated memory. The resulting success-rate change, entropy reduction, and realized experiment cost provide the reward. The manager is trained with PPO on discounted returns over L memory decisions. Here H_t denotes $\mathcal{H}_{\text{exp}}$ at decision t, with both entropy estimates evaluated over the same hypotheses. C_{t+1} uses the elapsed time, simulator steps, GPU cost, and reset cost in $C(\cdot)$.

Controlled Memory Ablation Protocol. Table C.3 documents the three memory-selection variants used in Table 5. This comparison separates learned action selection from retrieval-based selection and evaluates the joint contribution of entropy reduction and physical-experiment cost in the reward.

Table C.3: **Controlled Memory Ablation Protocol.** Selection rules and training rewards used in the three ablation variants.

Variant	Selection Rule	Training Reward
Retrieval-Based	Select the applicable action and target with the highest cosine similarity between the current experience and candidate description under the frozen encoder.	No PPO training
PPO + Task Reward Only	PPO Manager selects an applicable action and target.	$\lambda_1(R_{t+1} - R_t)$
EmbodiedEvo	The same PPO Manager selects an applicable action and target.	$r_t^{\text{mem}} = \lambda_1(R_{t+1} - R_t) + \lambda_2(H_t - H_{t+1}) - \lambda_3 C_{t+1}$

Across these experiments, we hold the remaining components constant. Each Manager receives the current experience and retrieved M1, M2, and M3 records, including hypothesis confidence and evidence links. All variants share the six-action space, target construction, applicability rules, and Skip availability. The encoder, frozen Qwen operator, operator prompts, and generation settings are identical. Action instructions remain unchanged during this comparison. Fast-System hypothesis updates, Code-Skill Co-Evolution, and Value-of-Information Experiment Selection follow the same rules. All variants start from the same memory and hypotheses and receive equal physical-interaction budgets, task distributions, and evaluation protocols. Only the selection rule or training reward differs.

D Value-of-Information Experiment Selection

Notation. Let $\mathcal{G}$ be the Hypothesis Graph, $\mathcal{H}$ its code and skill hypotheses, and D_k the matched physical outcomes observed after k episodes. An experiment t may invoke any ordered sequence of code and skill nodes. Its tested set $S_t \subseteq \mathcal{H}$ identifies the nodes evaluated by matched comparisons. For each tested node, the comparison preserves the initial state and all other calls in their original order. A positive success-rate difference adds one to $n_{\text{sup},i}$, a negative difference adds one to $n_{\text{ref},i}$, and a tie adds one to $n_{\text{tie},i}$. The evaluation count is $n_{\text{eval},i} = n_{\text{sup},i} + n_{\text{ref},i} + n_{\text{tie},i}$. Each tested node is compared once per experiment. Comparisons of several tested nodes use independent resets, and their physical episodes are included in $q(t)$.

For hypothesis h_i, let θ_i be the probability of a positive difference when the matched outcomes differ. A shared $\text{Beta}(\alpha, \beta)$ prior yields $\theta_i \mid D_k \sim \text{Beta}(a_i, b_i)$, where $a_i = \alpha + n_{\text{sup},i}$ and $b_i = \beta + n_{\text{ref},i}$. Let ρ_i be the probability that the matched outcomes differ, with shared prior $\text{Beta}(\gamma, \delta)$ and posterior parameters $u_i = \gamma + n_{\text{sup},i} + n_{\text{ref},i}$ and $v_i = \delta + n_{\text{tie},i}$. The probabilities for the next matched comparison are:

$$
\begin{aligned}
P(e_i = + \mid D_k) &= \frac{u_i}{u_i + v_i} \frac{a_i}{a_i + b_i}, \quad P(e_i = - \mid D_k) = \frac{u_i}{u_i + v_i} \frac{b_i}{a_i + b_i}, \\
P(e_i = 0 \mid D_k) &= \frac{v_i}{u_i + v_i}.
\end{aligned}
\tag{11}
$$

For several tested nodes, $P(e \mid D_k, t)$ is the product of these probabilities across their independently reset comparisons. Both entropy terms in $G_k(t)$ use the current hypothesis set. New hypotheses enter the next round with the same priors and receive an initial matched comparison before $G_k(t)$ is used to select their next experiment. The budget is B physical episodes, $n_{\max}$ is the evaluation cap per hypothesis, and $\epsilon_H, \epsilon_{\text{VOI}}$ are the stopping thresholds. Candidates returns experiments with nonempty tested sets that fit the remaining budget and cap. UpdateGraph updates the three outcome counts and supported WORKS-WITH edges. Propose returns new code or skill hypotheses and revisions from the recorded evidence. Insert gives every proposal the shared priors and adds a REFINES edge only for a revision.

Algorithm 1 Hypothesis Graph Confidence and Value-of-Information Experiment Selection

Require: Initial graph $\mathcal{G}$; shared priors $\alpha, \beta, \gamma, \delta$; budget B; cap $n_{\max}$; thresholds $\epsilon_H, \epsilon_{\text{VOI}}$
$D, \mathcal{M}_1 \leftarrow \emptyset$; $b \leftarrow 0$; $n_{\text{sup},i}, n_{\text{ref},i}, n_{\text{tie},i} \leftarrow 0$ for all initial h_i
while $b < B$ **do**
  $\mathcal{H} \leftarrow \text{nodes}(\mathcal{G})$; $\mathcal{U} \leftarrow \{h_i \in \mathcal{H} : n_{\text{eval},i} < n_{\max}\}$
  $\mathcal{T} \leftarrow \text{Candidates}(\mathcal{G}, \mathcal{U}, B - b, n_{\max})$
  if $\mathcal{T} = \emptyset$ **then**
    break
  end if
  $\mathcal{T}_0 \leftarrow \{t \in \mathcal{T} : S_t \cap \{h_i : n_{\text{eval},i} = 0\} \neq \emptyset\}$
  if $\mathcal{T}_0 \neq \emptyset$ **then**
    $t^* \sim \text{Uniform}(\mathcal{T}_0)$
  else
    $q_i \leftarrow P(\theta_i > 1/2 \mid D)$; $\mathcal{H}_{\text{exp}}(D) \leftarrow \sum_{h_i \in \mathcal{H}} h_2(q_i)$
    for each $t \in \mathcal{T}$ **do**
      $G_b(t) \leftarrow \mathcal{H}_{\text{exp}}(D) - \mathbb{E}_{e|D,t}[\mathcal{H}_{\text{exp}}(D \cup \{e\})]$
      $\text{VOI}_b(t) \leftarrow G_b(t)/C(t)$
    end for
    if $\mathcal{H}_{\text{exp}}(D) < \epsilon_H$ or $\max_{t \in \mathcal{T}} \text{VOI}_b(t) < \epsilon_{\text{VOI}}$ **then**
      break
    end if
    $t^* \leftarrow \arg\max_{t \in \mathcal{T}} \text{VOI}_b(t)$
  end if
  $e \leftarrow \text{Execute}(t^*)$; $b \leftarrow b + q(t^*)$; $D \leftarrow D \cup \{e\}$; $\mathcal{M}_1 \leftarrow \mathcal{M}_1 \cup \{e\}$
  $\mathcal{G} \leftarrow \text{UpdateGraph}(\mathcal{G}, t^*, e)$
  $(\mathcal{P}_{\text{new}}, \mathcal{P}_{\text{rev}}) \leftarrow \text{Propose}(\mathcal{G}, \mathcal{M}_1, e)$
  $\mathcal{G} \leftarrow \text{Insert}(\mathcal{G}, \mathcal{P}_{\text{new}}, \mathcal{P}_{\text{rev}})$
end while
$c_i \leftarrow a_i/(a_i + b_i)$; $q_i \leftarrow P(\theta_i > 1/2 \mid D)$ for all $h_i \in \text{nodes}(\mathcal{G})$
return $\mathcal{G}, \{c_i, q_i\}$

E Fast-System Experimental Configuration

Table E.1 specifies the configuration used in our experiments. The cost weights express time in seconds, simulator steps as counts, GPU use in GPU-seconds, and reset cost as the number of resets. The entropy threshold scales with the current hypothesis count, corresponding to an average posterior entropy below 0.1 bit per hypothesis.

Table E.1: **Fast-System Experimental Configuration.** Numerical settings and online evaluation budget used in our experiments.

Parameter	Value	Definition
α, β	$1, 1$	Beta prior for hypothesis support
γ, δ	$1, 1$	Beta prior for matched-outcome disagreement
w_τ	0.01	Weight on elapsed execution time
w_s	2.5×10^{-4}	Weight per simulator step
w_g	0.005	Weight on GPU-seconds
w_r	0.10	Weight per environment reset
ϵ_H	$0.1\lvert\mathcal{H}\rvert$ bits	Threshold for total hypothesis entropy
ϵ_{VOI}	10^{-3}	Minimum VOI for continuing exploration
$n_{\max}$	6	Maximum matched evaluations per hypothesis
B	200 episodes per task run	Maximum physical-interaction budget
Reporting window	Final 100 episodes	Online success while the harness evolves

The online metric averages the binary task outcomes of the final 100 physical episodes. Each outcome evaluates the harness available at that episode. Reaching a stopping criterion ends broad hypothesis exploration, while task execution continues to complete the reporting window. We expect additional iterations to offer further opportunities for improving task success rate.

F PPO TRAINING CONFIGURATION

The Manager is trained with PPO during evolution and remains frozen during evaluation. Table F.1 lists the settings used to train the Manager. Each rollout collects $L = 32$ memory decisions before four PPO optimization epochs. The encoder and operator remain frozen.

Table F.1: **PPO Training Configuration.** Hyperparameters used for Manager training.

Parameter	Value	Definition
Optimizer	AdamW	Policy and value optimization
Learning rate	2.5×10^{-4}	Policy and value learning rate
PPO clip ratio	0.18	Clipping range for the policy objective
Rollout horizon L	32	Memory decisions per PPO rollout
PPO epochs	4	Optimization passes per rollout
Mini-batch size	16	Samples per gradient step
Discount γ	0.97	Discount for future memory rewards
GAE λ	0.92	Advantage-estimation smoothing
Entropy coefficient	0.012	Weight on policy entropy
Value-loss coefficient	0.5	Weight on value-function loss
Maximum gradient norm	0.8	Gradient-clipping threshold
Training seeds	10	Number of independent training seeds

G HYPOTHESIS GRAPH RELATIONS AND COST MODEL

The Hypothesis Graph contains executable code and skill hypotheses as nodes. A directed REFINES edge records version evolution from a parent hypothesis to a more specific hypothesis. A bidirectional WORKS-WITH edge records measured positive joint gain between a code hypothesis and a skill hypothesis. The Hypothesis Graph keeps significant WORKS-WITH edges with the highest joint gain and retains REFINES edges as version history. Candidate physical experiments declare the hypothesis subsets they distinguish and are ranked by the cost model in subsection 3.2. The cost model accounts for elapsed time, simulator steps, GPU use, and reset cost, so the Fast System searches lower-cost experiments before more expensive physical trials.

H TRIALS FROM COMMON INITIAL STATES

For each code hypothesis, skill hypothesis, or code-skill pair, EmbodiedEvo runs trials from the same initial states. The resulting success rate defines $R(\cdot)$ in the WORKS-WITH score. Every trial outcome is written to M1, updates the corresponding M2 hypothesis confidence, and can contribute to M3 cognition after repeated evidence supports a reusable pattern.

I SHARED BACKBONES FOR BASELINE COMPARISONS

Table I.1 specifies the shared robot foundation models used in comparisons between agentic harnesses. All compared harnesses use identical backbone weights and evaluation settings on each benchmark. Only the harness differs.

Table I.1: **Shared Backbones for Baseline Comparisons.** Backbone settings shared across the compared harnesses.

Benchmark	Shared Backbone and Weights	Difference
RoboCasa365	Xiaomi-Robotics-1, identical weights	Agentic harness
LIBERO-Pro	$\pi_{0.5}$, official RLinf fine-tuned weights	Agentic harness