ROBOHARNESS: MEMORY-DRIVEN ORCHESTRATION OF HETEROGENEOUS ROBOT POLICIES FOR LONG-HORIZON PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon robotic tasks require a breadth of capabilities beyond what any single existing robot control policy can reliably provide. Combining heterogeneous policies with complementary strengths offers a promising solution, but introduces two key challenges: uncertain capability boundaries and distribution mismatches during policy handoffs. These challenges remain largely unaddressed by existing planning methods, which typically assume homogeneous, predefined skills with fixed applicability. We propose RoboHarness, a unified framework that encapsulates independently developed heterogeneous policies, including vision-language-action models (VLAs), world-action models (WAMs), reinforcement learning (RL) policies, and task and motion planners (TAMP), as reusable agentic skills. RoboHarness integrates understanding, memory, and evolution skills to reason about policy capabilities and support capability-aware task decomposition and policy routing. To mitigate distribution mismatches during policy handoffs, we introduce Memory Bridge, a plug-in policy-chaining mechanism that enables reliable transitions between heterogeneous policies without joint retraining. Extensive experiments across five public benchmarks, 500 customized tasks across 10 classes, and 135 real-robot trials demonstrate substantial gains in long-horizon and memory-dependent tasks, as well as robustness to out-of-distribution conditions.

1 INTRODUCTION

Long-horizon robotic tasks require instruction alignment, generalization across environmental variations, logical consistency, and noise robustness (Garrett et al., 2020; Yang et al., 2025). Despite substantial progress (Huang et al., 2026b; Black et al., 2025b; Silver et al., 2024; Ye et al., 2025), no single policy reliably meets all these requirements. Vision-language-action models (VLAs) provide semantic grounding but struggle with long-horizon consistency (Black et al., 2025b; Kim et al., 2025; Zheng et al., 2025). World-action models (WAMs) support predictive control but face prediction errors and inference latency (Zhang et al., 2026b). Reinforcement learning (RL) post-training improves closed-loop control but depends on reward design and training coverage (Zang et al., 2026; Physical Intelligence et al., 2025). Task and motion planning (TAMP) offers logical and geometric reasoning but relies on predefined abstractions and skills (Silver et al., 2023; Kumar et al., 2023; Liu et al., 2025; Huang et al., 2026c). These complementary strengths motivate coordinating heterogeneous policies for long-horizon execution.

Such coordination presents two challenges. First, task decomposition and policy selection must account for uncertain, context-dependent capability boundaries. Conventional planners typically assume predefined skills with fixed applicability (Liang et al., 2025; Han et al., 2024; Kaelbling & Lozano-Pérez, 2011), whereas independently developed policies differ in architecture, interfaces, and assumptions (Zhang et al., 2026b; Garrett et al., 2021). Their applicability may overlap, and relative performance varies with object configurations, observations, instructions, and execution history. Planners must therefore jointly determine task decomposition and policy assignment under current conditions. Second, successful policy selection does not ensure compatible handoffs: one policy's terminal state may fall outside the next policy's feasible execution region or input distribution (Lee et al., 2022; 2019), causing failures and cascading errors. Effective orchestration thus requires both capability-aware planning and reliable policy handoffs.

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.

We propose RoboHarness, a unified framework that exposes independently trained or hand-designed policies as reusable agentic skills through a common interface accommodating policy-specific inputs and outputs. It uses contextual and historical execution evidence to guide policy assignment, and the incoming policy's retrieved experience to construct compatible handoffs. Our main contributions are: **(1)** We introduce RoboHarness, a unified framework that orchestrates heterogeneous robot policies as reusable agentic skills for long-horizon tasks. **(2)** We integrate understanding, memory, and self-evolution skills to characterize and adapt context-dependent policy capabilities, supporting capability-aware task decomposition and routing. **(3)** We introduce Memory Bridge, a policy-chaining algorithm that learns a local progress ranking from retrieved trajectories and selects reachable, memory-supported handoff configurations without joint policy retraining. **(4)** We evaluate RoboHarness across five public benchmarks, 500 customized tasks, and 135 real-robot trials, demonstrating substantial improvements in long-horizon and memory-dependent task success and out-of-distribution robustness.

Our contribution lies in translating contextual information and execution experience into effective task decomposition, policy selection, and handoff coordination, enabling heterogeneous policies to collectively accomplish tasks that no constituent policy can reliably complete alone.

2 Related Work

Harness Systems for Agentic and Robotic Execution. Harness systems embed language models in execution loops with tools, memory, and feedback for state tracking and iterative refinement (Yao et al., 2023; Schick et al., 2023; Shinn et al., 2023). Extensions include executable skill libraries, multi-agent collaboration, and software-engineering agents (Wang et al., 2024a; Wu et al., 2024; Hong et al., 2024; Wang et al., 2024b; Yang et al., 2024; Wang et al., 2025; Xia et al., 2025). Concurrent robotic harnesses support VLA data collection, execution, and self-improvement (Li et al., 2026b; Zhang et al., 2026a), incorporating physical execution constraints (Xiao et al., 2026). Heterogeneous policies additionally differ in capabilities, assumptions, and representations (Zhang et al., 2026b; Dalal et al., 2023; Huang et al., 2026a; 2023b), complicating selection and handoffs. Existing robotic harnesses orchestrate learned primitives (Li et al., 2026b) or combine a VLA with analytic primitives (Zhang et al., 2026a), but do not explicitly address handoffs among independently developed policy families. RoboHarness combines capability-aware task decomposition and routing with explicit bridging between heterogeneous policies.

Long-Horizon Planning and Policy Chaining. Long-horizon robot planning includes hierarchical, task and motion (TAMP), skill-based, and language-model-based approaches (Silver et al., 2024; Liang et al., 2025; Athalye et al., 2026; Garrett et al., 2020; Liu et al., 2025; Oswald et al., 2024; Huang et al., 2026c). TAMP combines symbolic reasoning with geometric feasibility checks over predefined skills (Garrett et al., 2021; Kaelbling & Lozano-Pérez, 2011), while language-model-based planners generate and adapt plans from natural-language goals (Zhao et al., 2023; Yang et al., 2025; Guan et al., 2023; Chen et al., 2024; Hu et al., 2024). At the planning level, these approaches typically operate over low-level skills with specified applicability conditions, while compatibility between successive skills is assumed. Such abstractions do not by themselves resolve uncertain capability boundaries or context-dependent routing among independently developed policies. Beyond high-level planning, policy chaining poses a further challenge for hierarchical execution. Skill chaining aligns one policy's terminal states with the next policy's executable region through learned initiation sets (Konidaris & Barto, 2009; Huang et al., 2023a), transition mechanisms (Lee et al., 2019; Sahni et al., 2017; Mishra et al., 2023), or terminal-state regularization (Lee et al., 2022), typically requiring dedicated training or policy regularization for a given skill set. RoboHarness jointly addresses task decomposition, policy assignment, and handoff adaptation across independently developed policy families through capability-aware planning and memory-guided bridging without joint retraining.

3 Problem Definition

We consider long-horizon planning with heterogeneous robot policies $\Pi = \{\pi_1, \ldots, \pi_N\}$, without task-specific pretraining of the orchestration framework or joint retraining of constituent policies, while allowing online adaptation and experience accumulation during evaluation. Let $\mathcal{X}$, $\mathcal{O}$, and $\mathcal{G}$ denote the system state, global observation, and subtask instruction spaces, respectively. Each policy π_i has an achievable subtask space $\mathcal{G}_i \subseteq \mathcal{G}$ and a context-dependent applicability region $\mathcal{A}_i \subseteq \mathcal{O} \times \mathcal{G}_i$,

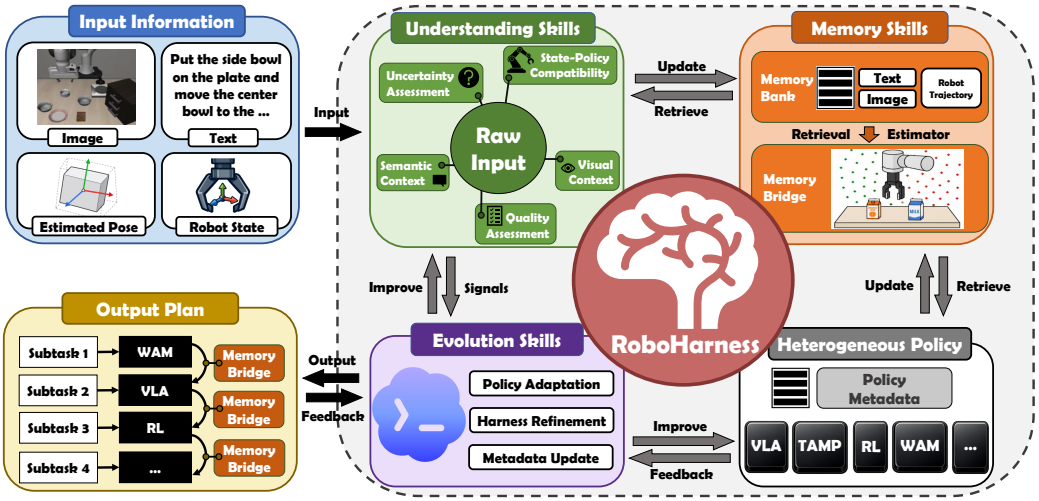


Figure 1: Overall Framework. RoboHarness integrates heterogeneous robot policies with three auxiliary skill libraries: Understanding Skills, Memory Skills, and Evolution Skills. Understanding Skills transform multimodal observations into task-relevant information for planning and policy selection. Memory Skills retrieve execution evidence for routing and construct Memory Bridge for policy transitions. Evolution Skills use online execution evidence to update individual policies and the harness, forming a closed information loop for capability-aware planning and adaptive orchestration.

containing observation–subtask pairs under which it can reliably execute. These regions are not explicitly known; policy metadata and execution evidence inform applicability judgments rather than provide an exact representation of $\mathcal{A}_i$. RoboHarness wraps each policy as an executable skill and maintains policy-specific episodic memory banks $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_N\}$, where $\mathcal{M}_i$ stores successful trajectories of π_i. It additionally maintains an execution log $\mathcal{H}$ that records the complete agent-level interaction trace for both successful and failed executions.

Given a high-level instruction I, an initial state $\mathbf{x}_0 \in \mathcal{X}$, the policy library Π, episodic memory $\mathcal{M}$, and execution log $\mathcal{H}$, RoboHarness seeks subtasks $\tau = (g_1, \ldots, g_T)$, policy assignments $\rho = (\pi_{k_1}, \ldots, \pi_{k_T})$, and bridge trajectories $B = (b_1, \ldots, b_{T-1})$ whose joint execution completes I. Here, $g_t \in \mathcal{G}$, $k_t \in \{1, \ldots, N\}$, and t indexes subtask stages. Let $o_t \in \mathcal{O}$ denote the observation at the start of stage t, from which the selected policy's native inputs are constructed. A reliable assignment requires $(o_t, g_t) \in \mathcal{A}_{k_t}$. Let $\bar{o}_t$ denote the terminal observation of π_{k_t}. When needed, b_t starts from the corresponding post-execution state and produces o_{t+1}, aiming to establish $(o_{t+1}, g_{t+1}) \in \mathcal{A}_{k_{t+1}}$. The problem thus jointly addresses task decomposition, capability-aware policy assignment, and inter-policy bridging to complete I with achievable subtasks and compatible execution conditions.

4 Methodology

RoboHarness uses a coding agent to decompose long-horizon tasks and orchestrate heterogeneous robot policies (Figure 1). Three auxiliary skill libraries support orchestration: understanding guides policy selection; memory retrieves experience and constructs Memory Bridge for policy handoffs; evolution refines policies and orchestration using execution feedback. Policies retain their native interfaces, requiring neither joint retraining nor a shared action representation. Implementation and policy details appear in Sections C and C.3.1.

4.1 Understanding Skills

Understanding skills extract decision-relevant information from raw inputs to assess policy applicability in the current context. These inputs include task instructions, visual observations, estimated object poses, robot states, and intermediate execution results. RoboHarness currently includes five types of understanding skills and supports extensions as needed. Uncertainty assessment computes the mean and variance of estimated object poses within a sliding time window to measure temporal

stability. Visual-context assessment compares visual embeddings of the current observation with those stored in policy-specific memory. Semantic-context assessment compares text embeddings of task instructions and candidate subtasks with those stored in policy-specific memory. Input-quality assessment evaluates exposure, sharpness, contrast, and image entropy to assess observation quality. State-policy compatibility assessment uses the memory-supported region and local progress estimator constructed by Memory Bridge (Section 4.2.2) to assess robot-configuration compatibility. The coding agent invokes these skills on demand and interprets their structured outputs alongside policy cards and the execution log to propose and revise task decompositions and policy assignments. The outputs provide complementary evidence about applicability, rather than a calibrated estimate of policy success. Encoders and assessment procedures are detailed in Section C.1.

4.2 Memory Skills and Memory Bridge

Memory skills serve two roles. First, policy-specific episodic memory supports history-dependent decisions and supplies successful execution experience for Memory Bridge construction. Second, the execution log maintains a complete chronological record of agent and skill interactions and execution feedback across both successful and failed executions. These records provide evidence for capability assessment, failure diagnosis, and evolution.

4.2.1 Memory Management

Each policy-specific episodic memory bank $\mathcal{M}_i$ stores successful trajectories $\xi = (n_1, \ldots, n_L)$ of temporally linked nodes. Each node n records a subtask instruction $g(n) \in \mathcal{G}$, observation $o(n) \in \mathcal{O}$, robot state $\mathbf{s}(n) \in \mathbb{R}^{d_s}$, gripper state $z(n)$, and text and visual embeddings from frozen encoders. The d_s-dimensional robot state comprises the end-effector pose and joint configuration, extracted from the full system state via $\mathbf{s} = r(\mathbf{x})$.

Given a subtask g and observation o, RoboHarness retrieves nodes hierarchically, first by text similarity and then by visual similarity:

$$\widetilde{\mathcal{N}} = \underset{\xi \in \mathcal{M}_i,\, n \in \xi}{\mathrm{TopK}_{K_{\text{text}}}} \mathrm{sim}_{\text{text}}(g, g(n)), \qquad \mathcal{N} = \underset{n \in \widetilde{\mathcal{N}}}{\mathrm{TopK}_{K_{\text{vis}}}} \mathrm{sim}_{\text{vis}}(o, o(n)),$$

where TopK_K selects the K highest-scoring nodes, and both similarity functions use cosine similarity between encoded representations (Section C.1). We denote this operation by $\mathrm{Retrieve}_{\mathcal{M}_i}(g, o) = \mathcal{N}$. Retrieval may span multiple banks or be restricted to one, such as $\mathcal{M}_{k_{t+1}}$. During execution, the task and scene context, agent and skill interactions, and execution feedback from both successful and failed executions are recorded chronologically in $\mathcal{H}$. These complete traces provide evidence for capability assessment, failure diagnosis, and subsequent evolution. Successful policy executions additionally update the corresponding episodic memory bank $\mathcal{M}_i$ with linked-node trajectories, whereas Memory Bridge retrieves only successful trajectories from $\mathcal{M}$. Node contents and update procedures are detailed in Section C.2.

4.2.2 Memory Bridge

Memory Bridge handles transitions where the terminal robot configuration of π_{k_t} is poorly aligned with states supported by $\pi_{k_{t+1}}$. Its role is to select a handoff configuration using the incoming policy's experience; a motion planner then determines how to reach it. Given the upcoming subtask g_{t+1} and terminal observation $\bar{o}_t$, RoboHarness retrieves anchors $\mathcal{N}_t = \mathrm{Retrieve}_{\mathcal{M}_{k_{t+1}}}(g_{t+1}, \bar{o}_t)$ and generates a bridge trajectory b_t through three stages.

Local progress estimation. For each anchor in $\mathcal{N}_t = \{n_{1,0}, \ldots, n_{K,0}\}$, RoboHarness collects neighboring nodes $n_{i,j}$ by expanding its trajectory bidirectionally while preserving the same subtask. This produces training pairs $(\mathbf{s}_{i,j}, y_{i,j}) = \big(\mathbf{s}(n_{i,j}), \Delta t_{i,j}\big)$, where $\Delta t_{i,j}$ is the signed temporal offset from the anchor: zero at the anchor, positive for subsequent nodes, and negative for preceding nodes. We train a lightweight progress estimator $f_{\text{score},t}$ to favor local forward progress, with details of its pairwise ranking SVM implementation provided in Section C.2.

Spatial state scoring. Candidate handoff states are scored by $f_{\text{score},t}$ within the supported region

$$\mathcal{R}_{\text{conf},t} = \left\{ \mathbf{s} \in \mathbb{R}^{d_s} \;\middle|\; \min_{\bar{\mathbf{s}} \in \mathcal{S}_{\text{ret},t}} \|\mathbf{s} - \bar{\mathbf{s}}\|_2 \le \epsilon \right\},$$

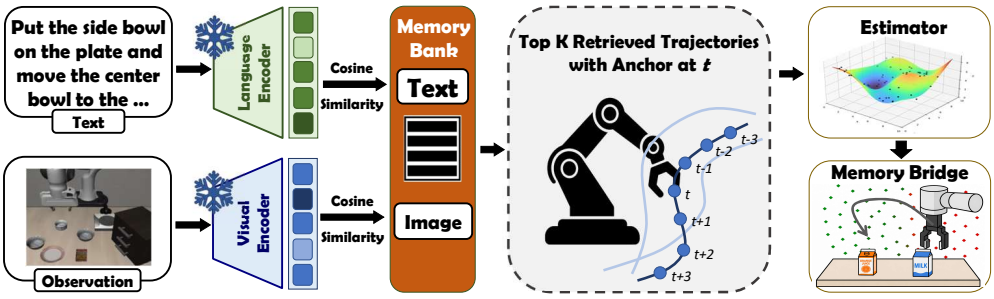


Figure 2: Memory Bridge. (1) Retrieve top-K anchors from the next policy's memory using the current observation and upcoming subtask instruction. (2) Fit a local progress estimator and construct a memory-supported neighborhood from robot states around these anchors. (3) Generate a bridge trajectory by balancing predicted progress score against motion cost.

where $\mathcal{S}_{\mathrm{ret},t} = \{\mathbf{s}_{i,j}\}$ contains the collected robot states and $\epsilon > 0$ bounds the distance to the nearest sample. We constrain handoff targets to this locally supported region to limit unsupported score extrapolation and guide transitions to the next policy.

Bridge trajectory generation. Let $\mathbf{s}_t$ be the robot state after executing π_{k_t}, and $\mathcal{S}_{\mathrm{plan},t}$ the states reachable from it through feasible motion plans. RoboHarness selects a target by balancing predicted progress against motion cost:

$$\mathbf{s}_t^* = \arg\max_{\mathbf{s}} \left[f_{\mathrm{score},t}(\mathbf{s}) - \lambda_{\mathrm{motion}} C_{\mathrm{motion}}(\mathbf{s}_t, \mathbf{s}) \right],$$
$$\text{subject to} \quad \mathbf{s} \in \mathcal{R}_{\mathrm{conf},t} \cap \mathcal{S}_{\mathrm{plan},t}, \quad f_{\mathrm{score},t}(\mathbf{s}) \geq \theta_{\mathrm{score},t}.$$

where C_{motion} measures the control steps required to reach a candidate, $\lambda_{\mathrm{motion}} \geq 0$ weights this cost, and $\theta_{\mathrm{score},t}$ is the minimum acceptable score. In practice, candidates are sampled, filtered by these constraints, and ranked by the objective. An off-the-shelf motion controller generates b_t from $\mathbf{s}_t$ to $\mathbf{s}_t^*$. Because robot-configuration support alone does not establish task and scene compatibility, the coding agent reassesses the observation and task context after execution before invoking the next policy or revising the routing decision. If the subsequent policy execution succeeds, its linked-node trajectory is additionally written to the incoming policy's episodic memory bank. We initialize $\theta_{\mathrm{score},t}$ to a preset value and iteratively lower it when no candidate is found, as detailed in Section C.2.

4.3 Heterogeneous Policies as Agentic Skills

RoboHarness wraps each robot policy as a callable agentic skill comprising the policy and an associated policy card. Each policy retains its native implementation, inputs, and outputs without requiring a shared action representation. The policy card records its capabilities, interface requirements, constraints, assumptions, training tasks, and execution statistics. The coding agent uses this metadata and can inspect the implementation code to assess policy applicability and compatibility. As execution evidence accumulates, policy cards are updated to support subsequent planning and assignment. This interface accommodates independently developed policies and supports extension to additional policy families. Our current implementation includes VLAs, WAMs, RL policies, and a TAMP planner. Implementation details and available checkpoint identifiers are provided in Section C.3.1.

4.4 Evolution Skills

Evolution skills read evidence from the execution log $\mathcal{H}$ to update policies, orchestration, and capability descriptions through three mechanisms. **Policy Adaptation** adjusts the task models and execution parameters of underlying policies. Our TAMP implementation uses SIMPACT (Liu et al., 2026a) to refine grasp poses and PDDLLM (Huang et al., 2026c) to learn new logical representations from observed interactions. **Harness Refinement** uses the coding agent to revise policy interfaces, routing, and inter-skill coordination based on execution traces and runtime errors. It also adjusts orchestration parameters, including retrieval counts and Memory Bridge thresholds, to improve policy selection and handoffs. **Metadata Update** revises policy cards using observed successes

Table 1: LIBERO task following and LIBERO-Plus perturbation robustness. The Original column reports performance on the original LIBERO benchmark. Each entry reports the success rate (%). The Overall column aggregates the seven perturbation categories, weighted by their official evaluation instance counts. The best and second-best results are shown in **bold** and underlined, respectively.

Method	Original	Robot State	Language	Layout	Background	Sensor	Camera	Light	Overall
$\pi_{0.5}$	96.9	77.5	85.6	85.7	94.6	89.7	75.4	96.9	85.7
π_0	91.3	40.8	63.5	76.4	84.1	80.1	61.0	89.3	69.4
UniVLA	95.2	46.2	69.6	31.9	81.0	21.2	1.8	69.0	42.9
OpenVLA-OFT$_m$	97.6	21.7	81.0	68.7	91.0	78.6	55.6	92.7	67.9
X-VLA	98.1	89.7	75.7	71.8	96.0	62.7	23.4	88.2	71.4
Cosmos-Policy	98.5	63.3	81.7	82.2	88.9	**92.7**	75.8	96.5	82.2
FAST-WAM	97.6	44.5	68.9	60.7	53.7	37.7	16.4	78.2	51.5
LingBot-VA	98.5	83.0	86.4	76.2	53.1	64.4	40.9	82.3	69.5
RoboHarness (ours)	**98.7**	**90.4**	**97.0**	**86.8**	**97.1**	90.3	**87.6**	**97.4**	**91.9**

and failures, recording the objects, tasks, and execution conditions associated with reliable or unsuccessful execution. These updated capability descriptions inform subsequent task decomposition and policy assignment. Persistent execution failures prompt the coding agent to retrieve relevant log traces and select appropriate updates. The resulting adaptation actions, validation outcomes, and retained changes are written back to $\mathcal{H}$. Validated changes persist across evaluation episodes, allowing accumulated experience to guide subsequent execution without task-specific pretraining of the orchestration framework. Implementation details, including parameter-tuning procedures, are provided in Section C.3.

5 Experimental Setup and Baselines

We evaluate RoboHarness against diverse baselines on five public benchmarks and in 135 real-robot trials. The coding agent uses Codex powered by GPT-5.5. Policy checkpoints and skill implementations are detailed in Section C.

Heterogeneous Policies Implementations. RoboHarness incorporates four independently developed policies: (1) the VLA $\pi_{0.5}$ (Black et al., 2025a); (2) Faster-WAM (Ma et al., 2026); (3) an open-source OpenVLA-OFT checkpoint (Kim et al., 2025), post-trained with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) using RLinf (Zang et al., 2026); and (4) a TAMP planner designed primarily for geometrically constrained pick-and-place tasks involving predefined objects, with support for multiple motion planners. Implementation details are provided in Section C.3.1. We inject 5–10% noise into simulator-provided object poses to approximate real-world estimation errors. Learned policies use either publicly released checkpoints or checkpoints trained or fine-tuned by us for the corresponding benchmarks. Policy inclusion depends on the availability of compatible checkpoints and the requirements of each benchmark. Each policy's episodic memory bank is initialized with successful trajectories from its corresponding dataset. Benchmark-specific policy configurations are detailed in Section D.1.

Evaluation on Benchmarks. The five public benchmarks probe task following, routing behavior and robustness, extended execution horizons, dependence on past observations, and mobile manipulation. **LIBERO** measures task-following performance (Liu et al., 2023). **LIBERO-Plus** assesses out-of-distribution robustness and routing behavior under perturbations (Fei et al., 2025). **LIBERO-LoHo** assesses extended-horizon execution on tasks averaging approximately four times the original LIBERO horizon (Huang et al., 2026a). **RMBench** evaluates memory-dependent long-horizon manipulation at different memory complexity levels (Chen et al., 2026). **BEHAVIOR-1K** evaluates long-horizon mobile manipulation, for which we conduct a partial evaluation on 6 selected tasks (Li et al., 2024). Selected tasks are detailed in Section D.3. Existing benchmarks do not explicitly require heterogeneous policy switching or evaluate policy handoffs. We introduce **LIBERO-Complex**, a LIBERO-based benchmark with 500 randomized instances across 10 long-horizon task classes requiring heterogeneous policy coordination, for ablation study (Section A). During evaluation, execution logs are continuously updated, successful executions update episodic memory banks, and experience is only retained across trials of the same task.

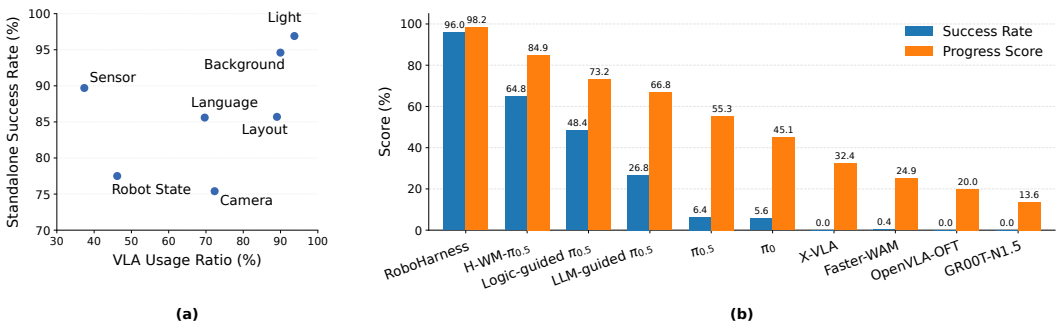


Figure 3: (a) Relationship between the VLA invocation ratio and its standalone performance across perturbation categories. (b) LIBERO-LoHo long-horizon evaluation results.

Real Robot Experiments. We conduct 135 real-robot trials across five assembly task classes. Each task involves 20–30 subtasks, with a 10-minute execution limit per trial. Given language instructions, RoboHarness combines TAMP-based assembly planning with $\pi_{0.5}$ for contact-rich cabinet opening and exploration. Tasks include constructing short and tall bridges, a Chinese character, and a boat, with some required blocks hidden inside the cabinet to necessitate exploration and retrieval. We test four execution-time disturbances: re-hiding required blocks, partially dismantling structures, perturbing estimated object poses, and introducing distractors.

Baselines We compare RoboHarness with two families of baselines: end-to-end policies and hierarchical methods. End-to-end policies include six VLAs, $\pi_{0.5}$ (Black et al., 2025a), π_0 (Black et al., 2025b), UniVLA (Bu et al., 2025), X-VLA (Zheng et al., 2025), OpenVLA-OFT (Kim et al., 2025), and GR00T-N1.5 (NVIDIA Research, 2025); four world-action models, LingBot-VA (Li et al., 2026a), Fast-WAM (Yuan et al., 2026), Faster-WAM (Ma et al., 2026), and Cosmos Policy (Kim et al., 2026); and Mem-0 (Chen et al., 2026), RLC (Larchenko et al., 2025), and G0.5 (Liu et al., 2026b). Hierarchical methods include H-WM (Huang et al., 2026a), LLM-guided VLA (Shi et al., 2025), and logic-guided VLA (Huang et al., 2026c), which provide intermediate guidance for long-horizon execution. Baseline results are taken from published evaluations (Huang et al., 2026a; Zhang et al., 2026b; Chen et al., 2026; Liu et al., 2026b) or obtained by evaluating publicly available checkpoints.

6 Results

In this section, we present evaluation results addressing the following six research questions: **RQ1:** Does policy usage adapt to context-dependent changes in execution reliability? **RQ2:** Can RoboHarness improve robustness to out-of-distribution perturbations? **RQ3:** Can RoboHarness coordinate heterogeneous policies for long-horizon tasks without task-specific orchestration co-training? **RQ4:** Can RoboHarness solve memory-dependent tasks requiring information beyond the current observation? **RQ5:** Can RoboHarness be extended to long-horizon mobile manipulation tasks? **RQ6:** How does each design component contribute to overall performance?

RQ1: Capability-aware policy routing. To evaluate whether RoboHarness can adapt policy routing to context-dependent capability changes, we use $\pi_{0.5}$ as the default base policy and pair it with a TAMP policy that provides complementary structured execution. On LIBERO-Plus, we compare the VLA invocation ratio with its standalone success rate across perturbation categories (Figure 3.(a)). VLA usage generally increases in categories with stronger standalone performance, providing behavioral evidence that routing responds to context-dependent reliability. The VLA is used more frequently under lighting, background, and layout changes than under robot-state perturbations. Sensor noise is an exception as it blurs observations, impairing the understanding skill's assessment.

RQ2: Robustness to perturbations. RoboHarness achieves the highest average success rate of 91.9%, ranking first in six of seven perturbation categories and outperforming every underlying policy overall (Table 1). These results show that RoboHarness effectively routes tasks according to the relative strengths of different policies, substantially improving robustness. Sensor noise remains the main bottleneck: degraded observations impair the understanding skill's assessment of policy suitability, leaving RoboHarness at 90.3%.

Table 2: RMBench and BEHAVIOR-1K evaluations. RMBench reports success rates (%) under different memory complexity levels, $M(1)$ and $M(n)$. BEHAVIOR-1K reports the average success rate (SR) and progress score (PS); a dash indicates unavailable SR. The best and second-best results for each metric are shown in **bold** and underlined, respectively.

(a) RMBench results.

Method	$M(1)$	$M(n)$
Diffusion Policy (Chi et al., 2025)	6.4	5.0
$\pi_{0.5}$ (Black et al., 2025a)	14.4	5.5
X-VLA (Zheng et al., 2025)	11.8	7.3
Mem-0 (Chen et al., 2026)	<u>52.8</u>	<u>28.5</u>
RoboHarness (ours)	**77.0**	**77.5**

(b) BEHAVIOR-1K results.

Method	SR	PS
$\pi_{0.5}$ (Black et al., 2025a)	0.05	0.26
RLC (Larchenko et al., 2025)	<u>0.23</u>	0.42
X-VLA (Zheng et al., 2025)	0.05	0.15
G0.5 (Liu et al., 2026b)	–	<u>0.54</u>
RoboHarness (ours)	**0.43**	**0.66**

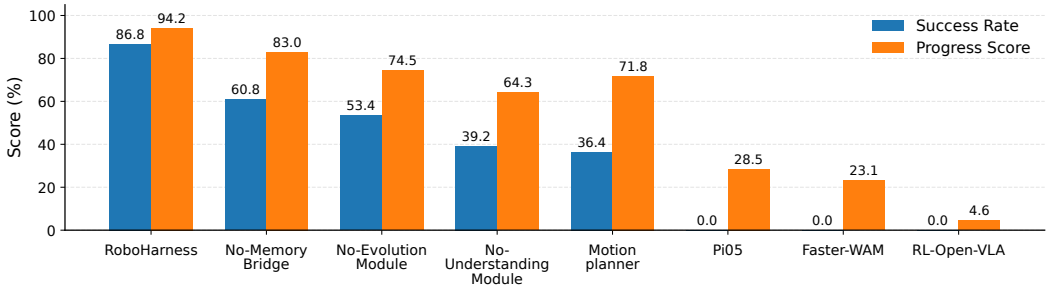


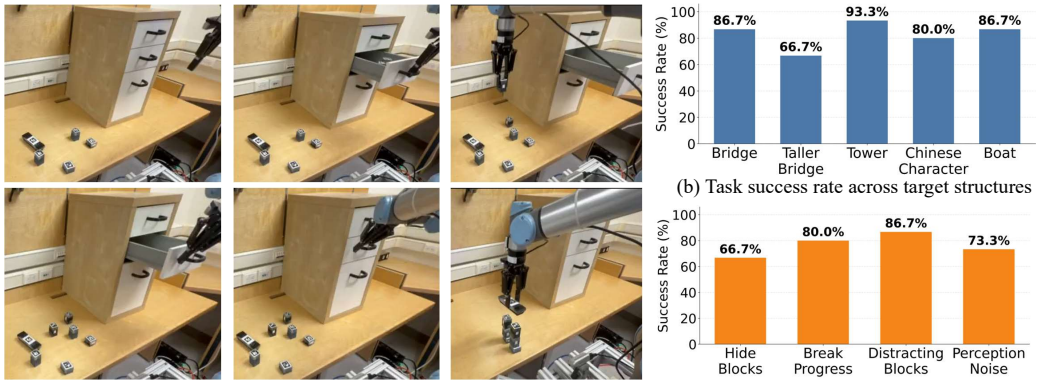
Figure 4: Ablation study of RoboHarness on LIBERO-Complex. We compare average success rates and progress scores for the full system and variants with individual components removed.

RQ3: Coordination of heterogeneous policies for long-horizon tasks. On LIBERO-LoHo, RoboHarness achieves the highest progress score and success rate on every task, averaging 98.2% and 96.0%, respectively (Figure 3.(b)). Individual policies often make partial progress but rarely complete the full task, while hierarchical guidance alone remains substantially below RoboHarness. This gap shows that long-horizon execution requires more than task decomposition: policy capabilities must be matched to individual subtasks, and intermediate states must remain compatible at policy transitions. RoboHarness addresses both requirements via capability-aware routing and Memory Bridge.

RQ4: Memory-dependent task execution. RMBench distinguishes between $M(1)$ tasks requiring a single key historical observation and $M(n)$ tasks requiring information accumulated across multiple past observations. RoboHarness achieves success rates of 77.0% and 77.5%, respectively, outperforming Mem-0 at both levels, with a larger advantage on $M(n)$ tasks (Table 2a). Across all nine tasks, it succeeds in 139 of 180 episodes, yielding an overall success rate of 77.2%. Its comparable performance across the two levels suggests that it can retain and use multiple historical observations without the degradation observed in the baselines. Task decomposition confines execution to tractable subproblems, while skill-relevant histories provide observations needed for later decisions. Together, these mechanisms support memory-dependent execution beyond immediate visual feedback. Complete task-level results are provided in Table 4.

RQ5: Long-horizon mobile manipulation. We evaluate RoboHarness on six selected BEHAVIOR-1K tasks involving long-horizon mobile manipulation (Table 2b). RoboHarness achieves a success rate of 0.433 and a progress score of 0.658, exceeding RLC (Larchenko et al., 2025) in success and G0.5 (Liu et al., 2026b) in progress. These tasks require coordinating navigation, grasping, placement, and precise interactions such as button pressing across spatially distributed locations. The results support the value of combining complementary capabilities within a single execution framework, while the remaining failures highlight the challenge of reliable coordination over long task horizons.

RQ6: Ablation Study. RoboHarness outperforms each constituent policy on 500 randomized instances across 10 long-horizon task classes designed for heterogeneous policy coordination (Figure 4). Among the three component ablations, removing understanding skills causes the largest performance drop, highlighting the value of contextual assessment for task decomposition and policy routing. Removing evolution skills also reduces performance, supporting the contribution of online refinement



(a) Real Robot Experiment Example

(b) Task success rate across target structures

(c) Task success under different disturbances

Figure 5: (a) Real robot experiment examples. (b) Overall success Rate of various target structure. (c) Overall success rate under different disturbances

as experience accumulates. Without Memory Bridge, the system retains substantial partial progress but completes fewer tasks. This pattern is consistent with Memory Bridge helping successful subtasks carry through to full-task completion by improving policy handoffs. Together, these results support the complementary roles of contextual assessment, online adaptation, and memory-guided handoffs.

7 Real Robot Experiments

We conducted 135 real-robot trials: 15 for each of five structures and four disturbance settings. In each construction trial, one to three required blocks were randomly hidden in a cabinet, requiring VLA-based identification and retrieval before TAMP assembly. As shown in Figure 5, performance was consistent across structures, with lower success on the Taller Bridge due to its complexity and physical instability. RoboHarness tolerated partial dismantling, pose estimation errors, and distractors, while re-hiding blocks caused the largest degradation by requiring repeated exploration and replanning. Timeouts were the primary failure mode. Under our setting, reasoning and planning API calls incurred an average latency of 4.6 seconds each, contributing 67.4 seconds of additional latency per trial on average across all 135 trials. This cumulative overhead reflects the multiple reasoning calls required for tasks comprising 20–30 subtasks.

8 Limitations and Future Work

Although RoboHarness effectively coordinates heterogeneous policies, it remains bounded by their capabilities and cannot perform tasks requiring capabilities absent from the entire policy library. Its reasoning and planning stages also add to overall execution time. Future work will broaden the policy library and use RoboHarness to collect execution trajectories for further training of robot policies. Incorporating additional modalities, such as tactile feedback, may improve contextual assessment.

9 Conclusion

We presented RoboHarness, a unified framework that couples capability-aware policy assignment with memory-guided handoffs for long-horizon manipulation. Contextual assessments, policy metadata, and the execution log support decomposition and routing, while Memory Bridge uses the incoming policy's retrieved experience to select reachable, memory-supported handoff configurations. The framework requires no task-specific orchestration pretraining or joint policy retraining, while permitting online adaptation and experience accumulation during evaluation. Results across five public benchmarks, including selected BEHAVIOR-1K tasks, 500 randomized instances across 10 task templates, and 135 real-robot trials demonstrate the effectiveness of the complete system. Component ablations further support the complementary value of contextual assessment, online refinement, and bridging for heterogeneous policy orchestration.

AI USE STATEMENT

Generative AI models are components of RoboHarness and are described explicitly as part of the methodology and experimental implementation. In preparing this paper, we used generative AI tools to assist with editing text, improving grammar and readability. We also used these tools to prepare figure components, including digitizing and refining hand-drawn sketches, removing image backgrounds, and converting graphics to vector formats. The authors reviewed the AI-assisted material, verified the reported results, and take responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

We provide the experimental setup in Section 5, implementation details in Section C, evaluation details in Section D, and additional experimental design details and results in the appendix. These materials document the policy interfaces, auxiliary skills, Memory Bridge construction, and evaluation procedures. Reproduction should account for online memory updates, experience accumulation across trials, and variability in the coding agent's outputs.

REFERENCES

Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Jiuguang Wang, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. From pixels to predicates: Learning symbolic world models via pretrained VLMs. IEEE Robotics and Automation Letters, 11(4):4002–4009, 2026. doi: 10.1109/LRA.2026.3662533.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In 9th Annual Conference on Robot Learning, 2025a.

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 XXI, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. In Robotics: Science and Systems XXI, 2025. doi: 10.15607/RSS.2025.XXI.014.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Junjie Wang, Tianhang Yang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. arXiv preprint arXiv:2603.01229, 2026.

Yongchao Chen, Jacob Arkin, Yilun Hao, Yang Zhang, Nicholas Roy, and Chuchu Fan. PRompt optimization in multi-step tasks (PROMST): Integrating human feedback and heuristic-based sampling. In Proc. Conf. Empirical Methods in Natural Language Processing, 2024.

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

Murtaza Dalal, Ajay Mandlekar, Caelan Reed Garrett, Ankur Handa, Ruslan Salakhutdinov, and Dieter Fox. Imitating task and motion planning with visuomotor transformers. In Proceedings of The 7th Conference on Robot Learning, pp. 2565–2593, 2023.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. arXiv preprint arXiv:2510.13626, 2025.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. Proceedings of the International Conference on Automated Planning and Scheduling, 30:440–448, 2020. doi: 10.1609/icaps.v30i1.6739.

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. Annual review of control, robotics, and autonomous systems, 4(1):265–293, 2021. doi: 10.1146/annurev-control-091420-084139.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. In Proc. Adv. Neural Inf. Proc. Systems, 2023.

Weihang Guo, Theodoros Tyrovouzis, Emiliano Flores, Clayton W Ramsey, Zachary K Kingston, Ioan A Şucan, Mark Moll, and Lydia E Kavraki. The open motion planning library 2.0. arXiv preprint arXiv:2605.29301, 2026.

Muzhi Han, Yifeng Zhu, Song-Chun Zhu, Ying Nian Wu, and Yuke Zhu. Interpret: Interactive predicate learning from language feedback for generalizable task planning. In Robotics: Science and Systems (RSS), 2024.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research, 14:253–302, 2001. doi: 10.1613/jair.855.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In International Conference on Learning Representations, 2024.

Mengkang Hu, Yao Mu, Xinmiao Yu, Mingyu Ding, Shiguang Wu, Wenqi Shao, Qiguang Chen, Bin Wang, Yu Qiao, and Ping Luo. Tree-planner: Efficient close-loop task planning with large language models. In International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=Glcsog6zOe.

Jinbang Huang, Wenyuan Chen, Zhiyuan Li, Oscar Pang, Xiao Hu, Lingfeng Zhang, Yuanzhao Hu, Zhanguang Zhang, Mark Coates, Tongtong Cao, et al. H-wm: Robotic task and motion planning guided by hierarchical world model. arXiv preprint arXiv:2602.11291, 2026a.

Jinbang Huang, Zhiyuan Li, Yuanzhao Hu, Zhanguang Zhang, Mark Coates, Xingyue Quan, and Yingxue Zhang. Self-criteach: Llm self-teaching and self-critiquing for improving robotic planning via automated domain generation. In International Conference on Machine Learning, 2026b. URL https://arxiv.org/abs/2509.21543.

Jinbang Huang, Yixin Xiao, Zhanguang Zhang, Mark Coates, Jianye Hao, and Yingxue Zhang. One demo is all it takes: Planning domain derivation with LLMs from a single demonstration. In International Conference on Learning Representations, 2026c.

Tao Huang, Kai Chen, Wang Wei, Jianan Li, Yonghao Long, and Qi Dou. Value-informed skill chaining for policy learning of long-horizon tasks with surgical robot. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8495–8501. IEEE, 2023a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In Proceedings of the 6th Conference on Robot Learning, volume 205 of Proceedings of Machine Learning Research, pp. 1769–1782. PMLR, 2023b.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In 2011 IEEE international conference on robotics and automation. IEEE, 2011.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In Robotics: Science and Systems XXI, 2025. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. arXiv preprint arXiv:2601.16163, 2026.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. Advances in neural information processing systems, 22, 2009.

Nishanth Kumar, Willie McClinton, Rohan Chitnis, Tom Silver, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning efficient abstract planning models that choose what to predict. In Proc. Conf. on Robot Learning, 2023.

Ilia Larchenko, Gleb Zarin, and Akash Karnatak. Task adaptation of vision-language-action model: 1st place solution for the 2025 BEHAVIOR challenge, 2025. URL https://arxiv.org/abs/2512.06951.

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In International conference on learning representations, 2019.

Youngwoon Lee, Joseph J Lim, Anima Anandkumar, and Yuke Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. In Proceedings of the 5th Conference on Robot Learning, volume 164 of Proceedings of Machine Learning Research, pp. 406–416. PMLR, 2022.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. arXiv preprint arXiv:2403.09227, 2024.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Luyao Zhang, Mingrui Yu, Zelin Gao, Nan Xue, Boyu Zhou, Xing Zhu, Mingyu Ding, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. In Robotics: Science and Systems XXII, 2026a. doi: 10.15607/RSS.2026.XXII.016.

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

Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B Tenenbaum, Tom Silver, João F Henriques, and Kevin Ellis. VisualPredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. In International Conference on Learning Representations, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/975db59bfa6eba2175a410f6afcecd99-Abstract-Conference.html.

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, 2023.

Haowen Liu, Shaoxiong Yao, Haonan Chen, Jiawei Gao, Jiayuan Mao, Jia-Bin Huang, and Yilun Du. Simpact: Simulation-enabled action planning using vision-language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2026a.

Weiyu Liu, Neil Nie, Ruohan Zhang, Jiayuan Mao, and Jiajun Wu. Learning compositional behaviors from demonstration and language. In Proceedings of the 8th Conference on Robot Learning, volume 270 of Proceedings of Machine Learning Research, pp. 1992–2028. PMLR, 2025.

Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, Anqi Yang, Shicheng Cao, Haonan Liu, Yue Sun, Zihan Guo, Xiao Liu, Dong Ke, Changxun Pan, Chenru Wu, Tailai Cheng, Xiaoshu Ren, Xinlei Zhang, Jianning Cui, Zijie Zhao, Haoyu Zhang, Kaiming Xu, Haodong Yang, Bowen Zhang, Jiahui Niu, Shaoting Zhu, Shiduo Zhang, and Hang Zhao. G0.5: One autoregressive stream for robot reasoning and action, 2026b. URL https://arxiv.org/abs/2608.11739.

Liheng Ma, Rui Heng Yang, Zhanguang Zhang, Mateo Clemente, Ziwen Hu, Tongtong Cao, and Yingxue Zhang. Faster-wam: Do world action models need deep action modules? arXiv preprint arXiv:2608.02365, 2026.

Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In Conference on Robot Learning, pp. 2905–2925. PMLR, 2023.

NVIDIA Research. GR00T N1.5: An improved open foundation model for generalist humanoid robots. NVIDIA Research, 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_5/.

James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. Large language models as planning domain generators. In Proc. Int. Conf. on Automated Planning and Scheduling, 2024.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. arXiv preprint arXiv:2511.14759, 2025.

Himanshu Sahni, Saurabh Kumar, Farhan Tejani, and Charles Isbell. Learning to compose skills. arXiv preprint arXiv:1711.11289, 2017.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. Advances in neural information processing systems, 36:68539–68551, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi Robot: Open-ended instruction following with hierarchical vision-language-action models. In Proceedings of the 42nd International Conference on Machine Learning, volume 267 of Proceedings of Machine Learning Research, pp. 54919–54933. PMLR, 2025. URL https://proceedings.mlr.press/v267/shi25d.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. Advances in neural information processing systems, 36:8634–8652, 2023.

Tom Silver, Rohan Chitnis, Nishanth Kumar, Willie McClinton, Tomás Lozano-Pérez, Leslie Kaelbling, and Joshua B Tenenbaum. Predicate invention for bilevel planning. In Proc. AAAI Conf. on Artificial Intelligence, 2023.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Kaelbling, and Michael Katz. Generalized planning in pddl domains with pretrained large language models. In Proc. AAAI Conf. on Artificial Intelligence, 2024.

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. Transactions on Machine Learning Research, 2024a. URL https://openreview.net/forum?id=ehfRiF0R3a.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In Forty-first International Conference on Machine Learning, 2024b.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In International Conference on Learning Representations, volume 2025, pp. 65882–65919, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In Proceedings of the First Conference on Language Modeling, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. Proceedings of the ACM on Software Engineering, 2(FSE):801–824, 2025. doi: 10.1145/3715754.

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

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. Advances in Neural Information Processing Systems, 37:50528–50652, 2024.

Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding long-horizon task and motion planning with vision language models. In 2025 IEEE International Conference on Robotics and Automation (ICRA), pp. 16847–16853, 2025. doi: 10.1109/ICRA55743.2025.11128705.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Haoming Ye, Yunxiao Xiao, Cewu Lu, and Panpan Cai. Unidomain: Pretraining a unified pddl domain from real-world demonstrations for generalizable robot task planning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), Advances in Neural Information Processing Systems, 2025.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? arXiv preprint arXiv:2603.16666, 2026.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Peihong Wang, Hua Yuan, Yixian Zhang, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. RLux-VLA: A unified and efficient framework for reinforcement learning of vision-language-action models. In Robotics: Science and Systems XXII, 2026. doi: 10.15607/RSS.2026.XXII.089.

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, 2026a.

Zhanguang Zhang, Zhiyuan Li, Behnam Rahmati, Rui Heng Yang, Yintao Ma, Amir Rasouli, Sajjad Pakdamansavoji, Yangzheng Wu, Lingfeng Zhang, Tongtong Cao, et al. Do world action models generalize better than vlas? a robustness study. arXiv preprint arXiv:2603.22078, 2026b.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

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. arXiv preprint arXiv:2510.10274, 2025.

A LIBERO-COMPLEX CUSTOMIZED TASKS

In this section, we provide full details of the LIBERO-Complex benchmark used in this paper, including both simulated and real-robot tasks.

A.1 SIMULATED TASKS

We construct 500 simulated instances from 10 long-horizon task templates, each evaluated under 50 randomly initialized tabletop configurations. The templates combine ambiguous or state-dependent instructions with object rearrangement, articulated-object manipulation, spatial reasoning, multi-object handling, and state-changing interactions. Approximately 20% of the ground-truth subtasks can be completed by only one of the four policy types—TAMP, VLA, WAM, or RL—given their respective capabilities. The remaining subtasks have overlapping policy coverage, but the policies differ in their success rates on them. The templates therefore test both whether RoboHarness selects a policy with the required capability and whether it selects an effective policy when multiple options are available. Results are reported in Figure 4. The ten templates are listed below:

1. Open the top drawer of the cabinet, place the middle bowl inside it, and place the remaining bowls on the plate.
2. Place the butter and chocolate pudding in the pan, turn on the stove, and finally place the pan on the stove.
3. Stack the bowl on the cabinet onto the ramekin, place the bowl on the stove onto the plate, and place the cookies on the cabinet.
4. Place both bowls on the microwave and close the microwave.
5. Place the bowl in the top drawer, close the drawer, and place the ketchup on the plate.
6. Place the ramekin on the stove, retrieve the Akita black bowl from the open top drawer of the wooden cabinet and place it on the plate, and place the cookies in the open drawer.
7. Place the bowl on the stove, turn on the stove, place the cream cheese on the bowl, and place the wine on the rack.
8. Collect the cookies, alphabet soup, cream cheese, and butter, and place them in the tray.
9. Place the side bowl on the cabinet, the center bowl on the plate, and the ramekin on the stove.
10. Switch the position of the left bowl and popcorn, and collect the other bowl on the cabinet.

These tasks test several challenges beyond standard single-stage manipulation. First, expressions such as "the middle bowl," "the remaining bowls," "the side bowl," and "the already-open top drawer" require grounding objects through their spatial relations and current scene state rather than fixed object identities. Second, the mixed and long-horizon requirement require the system to coordinate different underlying heterogeneous policies to accomplish the full task. Third, many instructions contain ordering dependencies: containers must be opened before objects are inserted, appliances must be activated at the appropriate stage, and intermediate placements alter the scene for subsequent subtasks. Finally, the combination of semantic interaction, geometric placement, and multi-step task reasoning requires RoboHarness to alternate between policies with complementary capabilities and maintain execution consistency across their handoffs.

B REAL ROBOT EXPERIMENT

Five different target structures: Using UR5e, We conducted 135 real-robot trials across five structure-construction tasks and four execution-time disturbance settings, with 15 trials for each setting. The full process of the experiment is shown in Figure 6. The five task classes are Bridge, Taller Bridge, Tower, Chinese Character, and Boat (Figure 7). The Bridge uses short blocks as lower piers, long blocks as vertical supports, and flat blocks as the bridge deck. The Taller Bridge increases the structural complexity by stacking two layers of long blocks to support an elevated deck. The Tower is a three-level structure constructed by arranging short and long blocks in a staggered configuration. For the Chinese Character task, one of four characters, is randomly selected as the

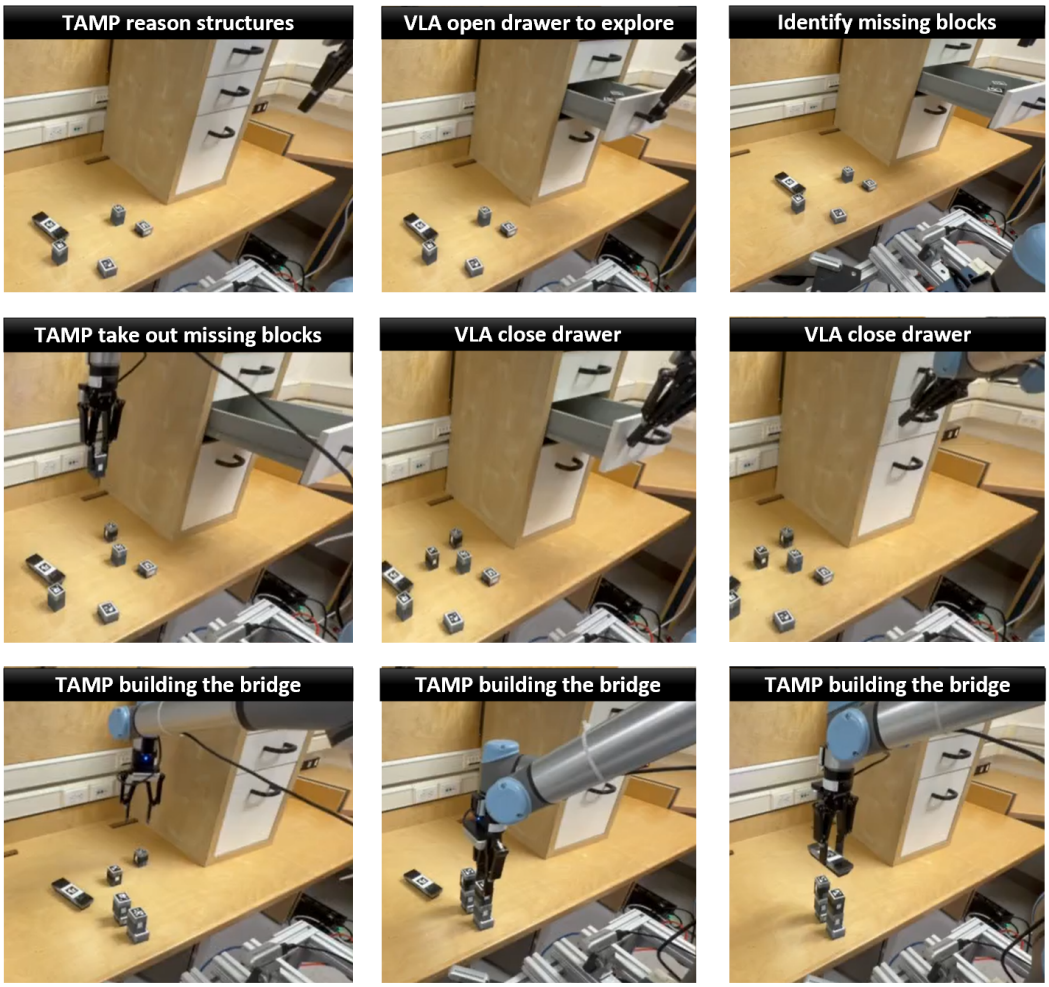


Figure 6: Full process of the real-robot experiment

target structure for each trial. The Boat is constructed by stacking blocks from short to long to form the hull and cabin, with a long block placed vertically on top as the mast.

In each trial, RoboHarness receives a language instruction describing the target structure and observes the blocks available in the workspace. It must infer the required block configuration, identify missing components, and generate a long-horizon assembly plan. Some required blocks are initially hidden inside a storage cabinet, requiring the robot to open the cabinet, actively explore its contents, and retrieve the missing blocks before completing the structure.

Our real-robot system integrates a TAMP planner with a VLA policy, $\pi_{0.5}$, fine-tuned on 50 human-demonstrated trajectories each for opening and closing cabinet drawers. The TAMP planner performs structural reasoning and generates the geometrically precise sequence of pick-and-place operations required for assembly. When RoboHarness determines that a required block is hidden, it transfers control to $\pi_{0.5}$ through the Memory Bridge. The VLA handles contact-rich interactions that are difficult to model explicitly within TAMP, including opening, closing the cabinet, and exploring its contents. Control is then transferred back to TAMP, which updates the object state and resumes the remaining assembly sequence. This workflow requires repeated capability-aware routing and stable handoffs between semantic, contact-rich exploration and precise long-horizon construction.

Execution-Time Disturbances: We further evaluate robustness by introducing four disturbances during execution: (1) re-hiding required blocks after the initially hidden blocks have been retrieved, forcing RoboHarness to detect the newly missing components and repeat the exploration procedure;

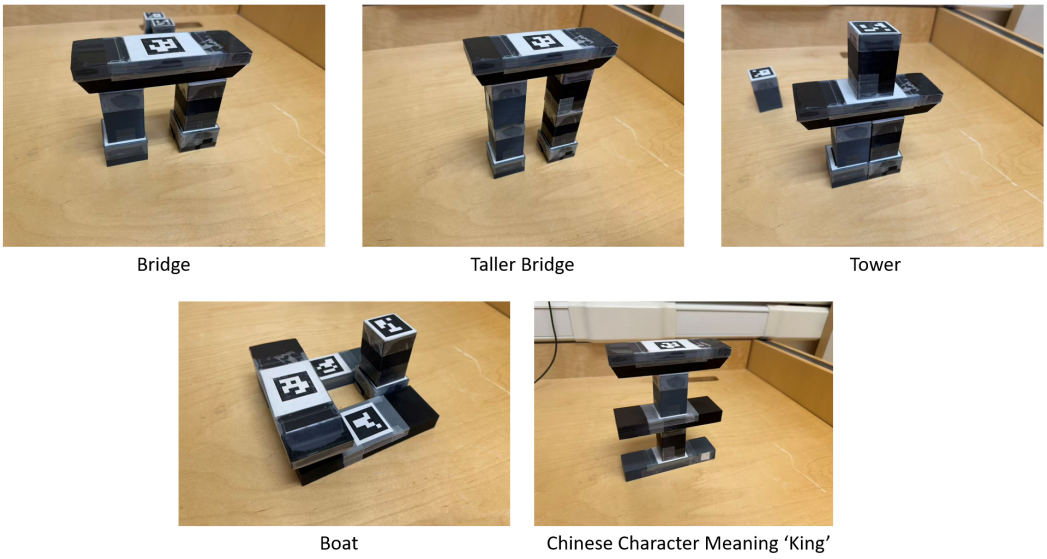


Figure 7: Five different target structure

(2) dismantling a partially completed structure, requiring the system to reassess task progress and replan the assembly; (3) injecting 5%-10% relative percentage noise into object-pose estimates to test robustness to perception errors; and (4) introducing distracting objects into the workspace to evaluate task-relevant reasoning and object selection. These disturbances test whether RoboHarness can recognize unexpected state changes, update its plan, reroute subtasks to appropriate policies, and recover without restarting the entire task.

Implementation Details. Our real-robot system is implemented on a UR5e manipulator and integrates a TAMP policy with a $\pi_{0.5}$ VLA policy. Two RealSense D435 cameras provide visual observations: one captures the agent view, and the other is mounted on the robot wrist. The TAMP policy uses the UR5e low-level control API for robot motion, PDDLStream (Garrett et al., 2020) for symbolic task reasoning and continuous motion-parameter sampling, and an ArUco-marker-based perception pipeline for object-pose estimation. It reasons about the required blocks and executes geometrically constrained pick-and-place operations during structure construction. The VLA policy is initialized from the official `pi05_libero` checkpoint and fine-tuned using 50 human-demonstrated trajectories for each of drawer opening and drawer closing. It handles the contact-rich cabinet interactions required to explore the drawer.

C Details of Auxiliary Skill Modules

This section provides implementation details corresponding to the understanding skills in Section 4.1, memory management and Memory Bridge in Section 4.2, self-evolution skills in Section 4.4, and heterogeneous policy interfaces in Section 4.3.

C.1 Understanding Skills

- **Uncertainty Assessment.** Perception outputs are aggregated over a temporal window to estimate the mean and variance of object poses. These statistics characterize temporal stability and are provided to the coding agent to support uncertainty-aware planning and policy selection.
- **Visual-Context Assessment.** We use two frozen visual encoders from Hugging Face: `google/siglip2-base-patch16-224` and `facebook/dinov2-base`. Let e_{sig}

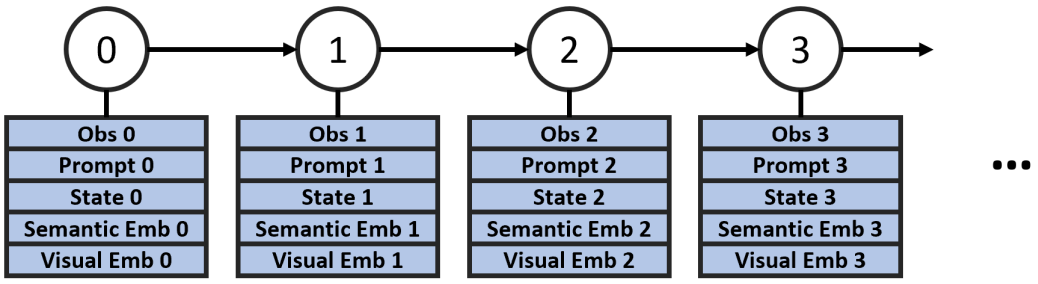


Figure 8: Memory Data Structure. Successful policy executions are stored in policy-specific episodic memory banks as temporally linked nodes containing observations, active subtask instructions, robot states, gripper states, and visual and semantic embeddings. Links support retrieval of execution context before and after an anchor node.

and e_{dino} denote their embedding functions. The visual similarity used in Section 4.2.1 is

$$\begin{aligned}\text{sim}_{\text{vis}}(o, o') = \alpha \cos(e_{\text{sig}}(o), e_{\text{sig}}(o')) \\ + (1-\alpha)\cos(e_{\text{dino}}(o), e_{\text{dino}}(o'))\,,\end{aligned}$$

where $\alpha \in [0, 1]$ controls the relative contribution of the encoders. This score is used for visual retrieval and provides evidence about the similarity between the current observation and policy-specific execution contexts.

- **Semantic-Context Assessment.** We use the frozen Hugging Face language encoder `BAAI/bge-large-en-v1.5` as e_{text}. Task instructions and candidate subtasks are embedded and compared with stored descriptions using cosine similarity. The resulting scores support semantic retrieval and assessment of a policy's task relevance.
- **Input-Quality Assessment.** Observation quality is evaluated using conventional image-processing algorithms. Exposure is measured from the mean grayscale intensity and the proportions of pixels below and above predefined intensity thresholds. Sharpness is quantified by the variance of the Laplacian response, while contrast is measured by the standard deviation of grayscale intensities. Image entropy is computed from the intensity histogram to characterize intensity variation. The normalized metrics are combined into an overall quality score and provided to the coding agent for observation selection and reliability-aware decision-making.
- **State-Policy Compatibility Assessment.** For a candidate policy and subtask, RoboHarness retrieves relevant policy-specific memory and constructs the local support region and progress estimator described in Section 4.2.2. The current robot state is assessed using its distance to retrieved states and its local progress score. These outputs provide evidence about robot-configuration compatibility and possible bridging requirements; they are interpreted jointly with visual, semantic, and task-state information by the coding agent.

C.2 Memory Skill Implementation Details

The memory skill library supports memory retrieval, memory update, and Memory Bridge construction. The retrieval and target-selection mechanisms are defined in Section 4.2. Here, we describe the stored data, the progress estimator used in our experiments, and a representative bridge execution.

Memory Data Structure. As illustrated in Figure 8, each linked-node trajectory records a successful policy execution from its initial observation to its terminal state. Nodes are connected in temporal order, allowing the system to recover preceding and subsequent execution context from a retrieved anchor. Each node stores the current visual observation $o(n)$, active subtask instruction $g(n)$, raw robot state $\widetilde{\mathbf{s}}(n)$, gripper state $z(n)$, and corresponding visual and semantic embeddings.

The raw robot state extracted from the full system state $\mathbf{x}$ is

$$\widetilde{\mathbf{s}}(\mathbf{x}) = \begin{bmatrix}\mathbf{p}^\top & \mathbf{q}^\top & \boldsymbol{\theta}^\top\end{bmatrix}^\top,$$

where $\mathbf{p} \in \mathbb{R}^3$ is the end-effector position in meters, $\mathbf{q} \in \mathbb{R}^4$ is its unit-quaternion orientation, and $\boldsymbol{\theta}$ contains the joint angles in radians. At bridge-construction time t, we normalize the raw state using the element-wise ranges of the corresponding policy-specific memory bank:

$$\mathbf{s}_t(\mathbf{x}) = \frac{\widetilde{\mathbf{s}}(\mathbf{x}) - \boldsymbol{\ell}_t}{\max(\mathbf{u}_t - \boldsymbol{\ell}_t, \epsilon)},$$

where $\boldsymbol{\ell}_t$ and $\mathbf{u}_t$ are the element-wise minimum and maximum of the raw states stored in that bank at time t, $\epsilon > 0$ prevents division by zero, and the division and maximum are applied element-wise. We use these same ranges to normalize the current state and all retrieved memory states for a given bridge construction. We write $\mathbf{s}(n)$ as shorthand for $\mathbf{s}_t(\mathbf{x}(n))$ when t is fixed. A current state outside the memory-bank range may have normalized components outside $[0, 1]$. Euclidean distances are computed between states normalized using the same ranges.

Visual embeddings from the frozen SigLIP2 and DINOv2 encoders and semantic embeddings from the frozen language encoder are stored alongside the node data to support efficient retrieval.

The execution log $\mathcal{H}$ maintains a complete chronological trace of the orchestration process. Each entry records the task and scene context, agent or skill invocation, exchanged messages and outputs, observations, actions, runtime feedback and errors, bridge executions, and intermediate or final outcomes. It also records evolution actions and their results, including parameter changes, metadata revisions, interface or routing updates, validation outcomes, and whether each change is retained. Evolution skills retrieve relevant log segments as evidence for capability assessment, failure diagnosis, and subsequent adaptation. If a policy execution succeeds, its recorded nodes are additionally linked chronologically and inserted as an episodic trajectory into the corresponding policy-specific bank $\mathcal{M}_i$. Contextual trajectory retrieval and Memory Bridge construction operate only on successful episodic trajectories in these banks.

Progress Estimator. We implement $f_{\text{score},t}$ using a linear pairwise ranking support vector machine (SVM). For each bridge construction, the retrieved anchor nodes are expanded into local trajectory segments as described in Section 4.2.2, preserving the same subtask and gripper state. Each sample consists of a normalized robot-state feature vector $\mathbf{s}_{i,j} = \mathbf{s}(n_{i,j})$ and a signed temporal offset $y_{i,j} = \Delta t_{i,j}$. Before training, the temporal offsets are normalized to $[-1, 1]$:

$$\hat{y}_{i,j} = \frac{y_{i,j}}{\Delta t_t^{\max}}, \qquad \Delta t_t^{\max} = \max_{i,j} |y_{i,j}|, \qquad \mathcal{D}_t = \{(\mathbf{s}_{i,j}, \hat{y}_{i,j})\}.$$

The anchor has $y_{i,0} = 0$, subsequent nodes have positive offsets, and preceding nodes have negative offsets.

The normalized temporal labels define relative preferences rather than regression targets. Because the anchors are retrieved using visual and semantic similarity, we treat them as approximately aligned points of task progress and use their relative temporal offsets to construct preferences across the retrieved segments. For two samples a and b in $\mathcal{D}_t$, $\hat{y}_a > \hat{y}_b$ indicates that $\mathbf{s}_a$ should receive a higher progress score than $\mathbf{s}_b$. We construct the pair set

$$\mathcal{P}_t = \{(a, b) \mid \hat{y}_a > \hat{y}_b\}$$

and fit the ranking SVM by solving

$$\begin{aligned} \min_{\mathbf{w}_t,\, \{\xi_{ab}\}} \quad & \frac{1}{2}\|\mathbf{w}_t\|_2^2 + C_{\text{svm}} \sum_{(a,b)\in\mathcal{P}_t} \xi_{ab}, \\ \text{subject to} \quad & \mathbf{w}_t^\top (\mathbf{s}_a - \mathbf{s}_b) \geq 1 - \xi_{ab}, \\ & \xi_{ab} \geq 0, \qquad \forall (a, b) \in \mathcal{P}_t, \end{aligned}$$

where C_{svm} controls the trade-off between regularization and ranking violations. The resulting progress estimator is

$$f_{\text{score},t}(\mathbf{s}) = \mathbf{w}_t^\top \mathbf{s}.$$

Thus, the output represents an ordering of local progress rather than elapsed time, a probability density, or a probability of successful execution. Although the temporal labels are normalized to $[-1, 1]$ before training, they impose only relative ordering constraints; consequently, the learned

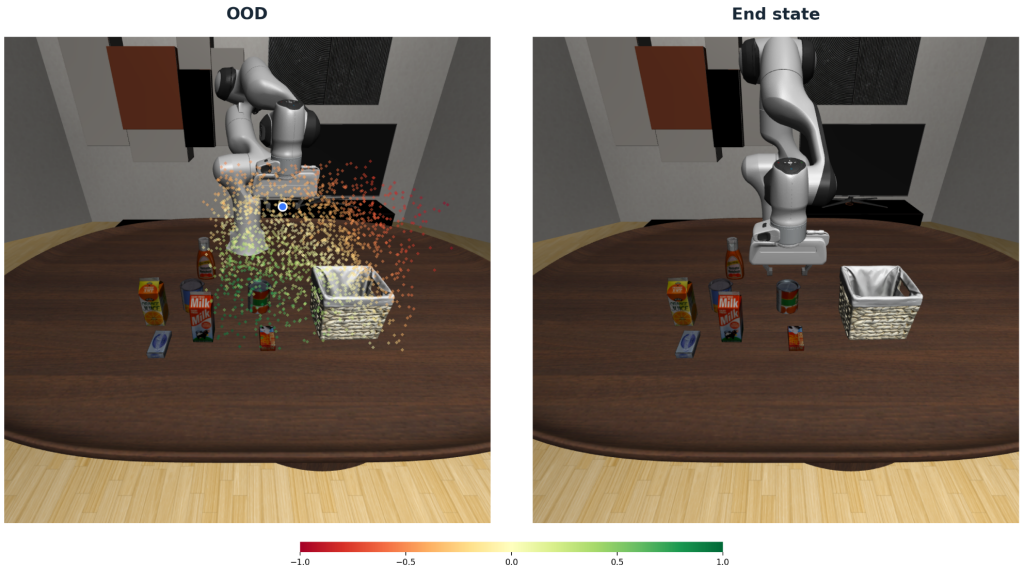


Figure 9: A Memory Bridge handoff. Left: an out-of-distribution (OOD) robot configuration and sampled candidate states, colored by their relative state scores (red: lower; green: higher). Right: the robot configuration goes back to distribution after executing the memory bridge trajectory.

SVM score is not itself restricted to this range. In particular, the zero label assigned to an anchor does not force its learned score to be zero.

Candidate targets are evaluated using the learned score together with the memory-support constraint, motion-planning feasibility, and motion cost defined in Section 4.2.2. We initialize the acceptance threshold as $\theta_{\text{score},t} = 0.9$. If candidate sampling produces no memory-supported, motion-feasible state whose score exceeds the current threshold, we reduce $\theta_{\text{score},t}$ by 0.02 and repeat the sampling and filtering procedure until an acceptable candidate is found or the 10-second time limit is reached. Fitting this auxiliary ranking model uses retrieved experience and does not jointly retrain the underlying control policies.

Memory Bridge Example. Figure 9 shows a representative policy handoff. The preceding policy leaves the robot in a configuration poorly aligned with the incoming policy's execution experience. Candidate states are filtered by memory support, progress score, and motion feasibility. RoboHarness selects a handoff target by balancing progress against motion cost, with $\lambda_{\text{motion}} = 0.2$, and plans a bridge to that target. After execution, the coding agent reassesses the observation and task context before invoking the next policy or revising the routing decision. This handoff requires no joint retraining of the constituent policies.

C.3 Evolution Skill Implementation Details

The self-evolution skill library contains three components: policy adaptation, harness refinement, and metadata update. These components use execution outcomes and retrieved experience to improve individual policies and their orchestration.

- **Policy Adaptation.** We implement SIMPACT (Liu et al., 2026a) and PDDLLM (Huang et al., 2026c) as adaptation methods for the TAMP system. SIMPACT adjusts motion-level parameters, such as object grasp poses, using execution feedback. PDDLLM extends the symbolic task model by learning new logical representations from observed interactions. These methods support adaptation to previously unsupported objects, relations, and task conditions. Policy adaptation also includes tuning exposed policy hyperparameters using the parameter-update procedure described below.

- **Harness Refinement.** RoboHarness uses Codex with GPT-5.5 to refine orchestration code online. Based on execution traces, runtime errors, and discrepancies between expected and observed outcomes, the coding agent identifies implementation bugs, repairs policy interfaces, and revises policy-routing and coordination logic. Modified implementations are evaluated through subsequent execution before being retained. Harness refinement also adjusts orchestration parameters, including retrieval counts, Memory Bridge support and score thresholds, and bridge-trajectory acceptance criteria.
- **Metadata Update.** Policy capability metadata is revised using execution evidence in $\mathcal{H}$. It records contexts in which a policy performs reliably or fails, including relevant objects, scenes, task types, and environmental conditions. Updated metadata is reflected in the policy cards and recorded in $\mathcal{H}$ for subsequent planning, routing, and evolution.

Parameter updates within Policy Adaptation and Harness Refinement share a directional tuning procedure. Each tunable parameter is assigned a discrete grid of candidate values. The coding agent selects whether to increase, decrease, or retain the current setting, and a separate tuning module maps this decision to the corresponding grid value and applies the update. This implements directional search over discretized parameters rather than exhaustive grid search.

All three components are enabled in the simulated experiments. Persistent failures trigger the retrieval of relevant execution evidence from $\mathcal{H}$, after which the coding agent selects the appropriate updates. The selected adaptations, their validation outcomes, and retained changes are appended to the log. Validated changes are retained across evaluation episodes, allowing continual adaptation. Thus, the setting does not require task-specific pretraining of the orchestration framework but permits the use of experience accumulated during evaluation.

C.3.1 Underlying Heterogeneous Policies

We incorporate four independently developed policies with complementary capabilities. Each is exposed through a callable skill interface and a policy card, as described in Section 4.3, while retaining its native inputs, outputs, and execution procedure.

- $\boldsymbol{\pi_{0.5}}$**.** For LIBERO, LIBERO-Plus, and LIBERO-LoHo, we use the official `pi05_libero` checkpoint released by Physical Intelligence through the `openpi` repository. This checkpoint was fine-tuned on LIBERO-Spatial, LIBERO-Goal, LIBERO-Long, and LIBERO-Object, providing broad language-conditioned manipulation capabilities (Black et al., 2025a). For RMBench and BEHAVIOR-1K, we use checkpoints trained by us for the respective benchmarks. For our $\pi_{0.5}$ fine-tuning, we adapt the official `openpi` training configuration[1] to the task-specific data. We use an action horizon of 50, a global batch size of 256, and the learning rate warms up for 10,000 steps to 5×10^{-5} and then decays over the next 20,000 steps. We use the checkpoint at step 30,000.
- **Faster-WAM.** We incorporate Faster-WAM (Ma et al., 2026) as a world-action-model policy and train it on LIBERO using the authors' released training pipeline. Its Dock of Transformer architecture connects a lightweight action head to a video Transformer backbone, allowing action prediction to use representations from the video model. RoboHarness invokes Faster-WAM through its native inference interface, adding video-model-based control to the heterogeneous policy library without requiring a shared action representation or joint retraining with the other policies.
- **RL-Post-Trained OpenVLA-OFT.** We use the RLinf/RLinf-OpenVLAOFT-GRPO-LIBERO-90 checkpoint released on Hugging Face. This policy was post-trained with GRPO on the 90 short-horizon tasks in LIBERO-90 using the RLinf framework Zang et al. (2026). It represents the RL-post-trained policy family within RoboHarness.
- **TAMP Planner.** We implement a TAMP system for geometrically constrained manipulation over a predefined object set. PDDL represents object types, predicates, and manipulation actions with their preconditions and effects, while each problem instance specifies the available objects, initial state, and task goal. We use the Fast-Forward (FF) planner (Hoffmann & Nebel, 2001) through PDDLStream (Garrett et al., 2020), which connects symbolic planning

[1] `https://github.com/Physical-Intelligence/openpi/blob/main/src/openpi/training/config.py`

with Python sampling procedures for continuous quantities such as grasp poses, placement poses, and robot configurations.

High-level actions, including picking, placing, stacking, and aligning, are executed using three motion generation and control mechanisms: (1) pose-based motion planning, which estimates object poses and generates robot trajectories using RRT through OMPL Guo et al. (2026); (2) VLM-based visual control, which guides robot motion from visual observations by outputting grid-based adjustments in eight translational and six rotational directions; and (3) memory-based trajectory augmentation, which incorporates retrieved demonstration segments into execution. In the memory-based configuration, memory skills retrieve successful episodic trajectories from $\mathcal{M}$ to disambiguate visually identical states, and each policy invocation is preceded by replaying the leading fraction ($\beta = 0.4$) of a retrieved demonstration segment. These mechanisms provide alternative ways to ground symbolic actions, while the TAMP system remains constrained by its predefined task domain and available execution capabilities.

Together, these policies provide complementary language-conditioned, video-model-based, reinforcement-learned, and symbolic-geometric capabilities for evaluating heterogeneous policy orchestration. Policy cards record their interface requirements, supported tasks, assumptions, constraints, and execution statistics, and are updated as new evidence accumulates.

D Evaluation Details on Benchmarks

D.1 Heterogeneous Policy Configurations across Benchmarks

Table 3 summarizes the heterogeneous policies available to RoboHarness across benchmarks and ablation studies. Policy configurations are determined by the availability of compatible checkpoints and the requirements of each benchmark. For LIBERO, LIBERO-Plus, RMBench, and BEHAVIOR-1K, RoboHarness orchestrates $\pi_{0.5}$ and TAMP, combining learned visuomotor control with explicit task and motion planning. For LIBERO-LoHo and the ablation study, the policy library additionally includes Faster-WAM and RLinf-OpenVLA, enabling evaluation with all four policy families. A checkmark denotes that a policy is available to the orchestrator, rather than necessarily invoked in every task; the policies used during execution depend on capability-aware task decomposition and routing.

Table 3: Heterogeneous policy configurations used by RoboHarness across benchmarks and ablation studies. A checkmark indicates that the policy is available for orchestration.

Benchmark / Study	$\pi_{0.5}$	Faster-WAM	RLinf-OpenVLA	TAMP
LIBERO	✓	–	–	✓
LIBERO-Plus	✓	–	–	✓
LIBERO-LoHo	✓	✓	✓	✓
RMBench	✓	–	–	✓
BEHAVIOR-1K	✓	–	–	✓
Ablation Study	✓	✓	✓	✓

D.2 RMBench

RMBench (Chen et al., 2026) is a memory-dependent robotic manipulation benchmark built on RoboTwin 2.0, comprising nine dual-arm manipulation tasks across two levels of task memory complexity: five $M(1)$ tasks and four $M(n)$ tasks. The $M(1)$ tasks require retaining information from a single key observation or a fixed, limited set of historical frames, such as an object's original position. The $M(n)$ tasks require accumulating and reasoning over multiple past observations, such as tracking previous attempts and their outcomes during trial-and-error interactions. These tasks evaluate memory retention and utilization when the current observation alone is insufficient for

decision-making. We evaluate RoboHarness on 20 episodes per task, with each episode recorded as a binary success or failure.

Table 4: Per-task success rates (%) on RMBench. Diffusion Policy, $\pi_{0.5}$, X-VLA, and Mem-0 results are from the RMBench paper (Chen et al., 2026); RoboHarness results use our full nine-task evaluation.

Task	TMC	Diffusion Policy	$\pi_{0.5}$	X-VLA	Mem-0	**RoboHarness**
Observe and Pick Up	$M(1)$	1	9	9	4	90
Rearrange Blocks	$M(1)$	0	13	13	89	85
Put Back Block	$M(1)$	0	11	18	90	100
Swap Blocks	$M(1)$	11	24	16	67	95
Swap T	$M(1)$	20	15	3	14	15
Battery Try	$M(n)$	10	16	26	28	50
Blocks Ranking Try	$M(n)$	10	6	1	18	80
Cover Blocks	$M(n)$	0	0	2	68	80
Press Button	$M(n)$	0	0	0	0	100
Overall average	–	5.8	10.4	9.8	42.0	**77.2**

D.3 BEHAVIOR-1K

BEHAVIOR-1K (Li et al., 2024) is a large-scale benchmark for evaluating embodied agents on diverse, realistic household activities in interactive environments. We select six mobile-manipulation tasks with varying horizons and interaction requirements: Turning On Radio (Task 0), Picking Up Trash (Task 1), Setting Mousetraps (Task 5), Moving Boxes to Storage (Task 16), Tidying Bedroom (Task 18), and Putting Shoes on Rack (Task 22). Our evaluations of RoboHarness, $\pi_{0.5}$ (Black et al., 2025a), Robot Learning Collective (RLC) (Larchenko et al., 2025), and X-VLA (Zheng et al., 2025) use 10 episodes per task. For G0.5 (Liu et al., 2026b), we use the published per-task progress scores of its four-epoch model; per-task binary success rates are unavailable. These tasks require agents to navigate across environments, manipulate multiple objects, and execute temporally ordered subtasks under substantial viewpoint and background changes. We report the arithmetic mean across the six tasks and evaluate SR independently from PS using the benchmark's binary success criterion. The task-level results are provided in Table 5.

Table 5: Per-task results on the six selected BEHAVIOR-1K tasks. SR and PS denote success rate and progress score, respectively. Dashes mark unavailable results. The average is the arithmetic mean across tasks, and all results are rounded to two decimal places.

	RoboHarness		$\pi_{0.5}$		RLC		X-VLA		G0.5	
Task	SR	PS	SR	PS	SR	PS	SR	PS	SR	PS
Task 0: Turning On Radio	0.90	0.90	0.20	0.20	0.30	0.30	0.20	0.20	–	0.05
Task 1: Picking Up Trash	0.50	0.73	0.00	0.15	0.30	0.62	0.10	0.40	–	0.85
Task 5: Setting Mousetraps	0.40	0.67	0.00	0.23	0.10	0.36	0.00	0.12	–	0.57
Task 16: Moving Boxes to Storage	0.60	0.75	0.10	0.33	0.70	0.73	0.00	0.03	–	0.55
Task 18: Tidying Bedroom	0.00	0.43	0.00	0.33	0.00	0.20	0.00	0.08	–	0.62
Task 22: Putting Shoes on Rack	0.20	0.46	0.00	0.34	0.00	0.33	0.00	0.05	–	0.57
Average	**0.43**	**0.66**	0.05	0.26	<u>0.23</u>	0.42	0.05	0.15	–	<u>0.54</u>

D.4 LIBERO-LoHo

LIBERO-LoHo evaluates long-horizon task execution across five tasks using progress score and success rate. RoboHarness achieves an average progress score of 98.2% and success rate of 96.0%, outperforming all compared methods on both metrics across all five tasks (Table 6).

Table 6: LIBERO-LoHo long-horizon evaluation. Each entry reports the progress score / success rate (%). The best and second-best results for each metric are shown in **bold** and underlined, respectively.

Method	Task 1	Task 2	Task 3	Task 4	Task 5	Overall
H-WM-$\pi_{0.5}$	<u>98.0 / 94.0</u>	<u>86.7 / 60.0</u>	<u>74.0 / 46.0</u>	<u>70.7 / 42.0</u>	<u>95.0 / 82.0</u>	<u>84.9 / 64.8</u>
Logic-guided $\pi_{0.5}$	95.3 / 86.0	84.7 / 58.0	54.7 / 16.0	39.3 / 4.0	92.0 / 78.0	73.2 / 48.4
LLM-guided $\pi_{0.5}$	84.7 / 54.0	80.7 / 42.0	68.0 / 24.0	41.3 / 4.0	59.5 / 10.0	66.8 / 26.8
$\pi_{0.5}$	66.0 / 4.0	73.3 / 24.0	54.7 / 4.0	44.7 / 0.0	38.0 / 0.0	55.3 / 6.4
π_0	54.0 / 0.0	62.0 / 28.0	44.0 / 0.0	31.3 / 0.0	34.0 / 0.0	45.1 / 5.6
X-VLA	48.8 / 0.0	16.7 / 0.0	23.3 / 0.0	33.3 / 0.0	40.0 / 0.0	32.4 / 0.0
Faster-WAM	0.0 / 0.0	64.0 / 2.0	28.7 / 0.0	32.0 / 0.0	0.0 / 0.0	24.9 / 0.4
OpenVLA-OFT	0.0 / 0.0	60.0 / 0.0	17.3 / 0.0	22.7 / 0.0	0.0 / 0.0	20.0 / 0.0
GR00T-N1.5	0.0 / 0.0	30.7 / 0.0	34.7 / 0.0	2.7 / 0.0	0.0 / 0.0	13.6 / 0.0
RoboHarness (ours)	**99.3 / 98.0**	**97.3 / 96.0**	**99.3 / 98.0**	**98.0 / 96.0**	**97.0 / 92.0**	**98.2 / 96.0**