

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

Jinbang Huang¹, Yuanzhao Hu^{2,*}, Zhiyuan Li^{3,*}, Ran Qi^{3,*}, Yixin Xiao¹, Zhanguang Zhang¹, Mark Coates⁴, Tongtong Cao⁵, Yingxue Zhang¹

¹Huawei Noah’s Ark Lab, ²University of British Columbia, ³University of Toronto,

⁴McGill University, ⁵Department of Foundation Model, 2012 Labs,

* Work done during the internship at Huawei Noah’s Ark Lab

ABSTRACT

Long-horizon robotic tasks require diverse capabilities that no single policy can reliably provide. Heterogeneous policies offer complementary strengths, but orchestrating them requires reasoning over uncertain capability boundaries and cross-policy distribution mismatch, which are largely overlooked by existing planning methods built on homogeneous, predefined skills with fixed applicability. We propose RoboHarness, a unified framework that encapsulates independently developed robot control systems as reusable agentic skills. Although instantiated in this work with VLAs, RL policies, and task-and-motion planning (TAMP) systems, RoboHarness is designed as a general framework compatible with a broader range of robot policies, such as navigation policies, model predictive controllers, and world-action models. RoboHarness uses multi-modal execution memory and online evidence to characterize policy capability boundaries for capability-aware decomposition and routing. To stabilize policy handoffs, its Memory Bridge retrieves execution trajectories associated with the next policy, estimates its in-distribution state region, and guides the robot toward that region without joint policy retraining. Extensive experiments on three public benchmarks, 500 customized tasks, and 135 real-robot experiments demonstrate effective capability-aware routing and stable policy orchestration, yielding substantial improvements in zero-shot long-horizon planning and out-of-distribution robustness.

1 INTRODUCTION

Long-horizon robotic tasks require policies to maintain instruction alignment, generalize across environment variations, preserve logical consistency and remain robust to noise (Garrett et al., 2020; Yang et al., 2025). Recent studies have made substantial progress in robotic systems (Huang et al., 2026b; Black et al., 2024; Silver et al., 2024; Ye et al., 2025), yet no existing policy excels across all requirements, exhibiting distinct strengths and limitations. For example, vision-language-action models (VLAs) provide semantic grounding and open-vocabulary instruction following (Black et al., 2024; Kim et al., 2025; Zheng et al., 2025), but their long-horizon consistency and geometric precision remain limited. Reinforcement learning (RL) post-training can produce closed-loop behaviors within specific training distributions (Zang et al., 2025; Intelligence et al., 2025), but its applicability often degrades under distribution shift and when reward functions are poorly specified. Task and motion planning (TAMP) systems provide structured logical and geometric reasoning (Silver et al., 2023; Kumar et al., 2023; Liu et al., 2024; Huang et al., 2025), but their generalization remains constrained by predefined abstractions, skills, and state representations. More broadly, existing policies are capability-bounded, and no single policy provides a universal solution for long-horizon tasks that span subtasks requiring different strengths. This motivates formulating long-horizon execution as the coordination of heterogeneous policies with distinct capability boundaries.

This perspective introduces a planning problem that remains largely unaddressed by existing planning methods: how to coordinate independently designed or trained heterogeneous control policies to solve long-horizon tasks that exceed the intrinsic capabilities of any individual policy. Conventional

long-horizon planning approaches typically assume a homogeneous set of predefined skills with fixed applicability (Liang et al., 2024; Han et al., 2024; Kaelbling & Lozano-Pérez, 2011), so task decomposition can be performed over a static skill space with clear skill boundaries. In contrast, heterogeneous robot policies differ in architecture, input-output requirements, policy assumptions, and execution history (Zhang et al., 2026; Garrett et al., 2021). Their capability boundaries are often unclear and context-dependent: different policies may have overlapping applicability, while their relative performance can vary with object configurations, visual observations, language instructions, and execution history. Consequently, a planner must decompose tasks according to policy capabilities and determine which policy can reliably execute each subtask under the current conditions. Moreover, independently trained or hand-designed policies are not necessarily directly composable: the terminal state produced by one policy may not lie within the feasible execution region or input distribution of the next policy (Lee et al., 2021; 2019). Transferring control without accounting for this mismatch can therefore introduce distribution shift and cascading failures across policy handoffs.

We propose RoboHarness, a unified framework for policy capability-aware planning with heterogeneous robot policies. RoboHarness encapsulates independently trained or hand-designed policies as reusable agentic skill modules and augments them with auxiliary modules that characterize capability boundaries and mediate policy-specific input and output requirements. To support reliable inter-policy handoff, RoboHarness introduces a chaining technique, the memory bridge, which combines memory retrieval and spatial distribution learning to preserve spatial consistency. This design enables RoboHarness to decompose tasks according to policy applicability and adaptively coordinate selected policies to accomplish complex long-horizon tasks in a zero-shot manner. The main contributions of this paper are: (1) We introduce RoboHarness, a unified framework that orchestrates heterogeneous robot policies to solve zero-shot long-horizon robotic tasks. (2) We organize three groups of auxiliary skills, understanding, memory, and self-evolution, to characterize policy capabilities and support capability-aware task decomposition, policy routing, and orchestration. (3) We design the memory bridge, a plug-and-play inter-policy chaining technique that combines multimodal memory retrieval with spatial distribution learning to enable stable policy handoffs without additional joint training. (4) We extensively evaluate RoboHarness across multiple benchmarks, demonstrating improved long-horizon consistency, out-of-distribution resilience, and zero-shot robustness.

2 RELATED WORK

Harness Systems for Agentic and Robotic Execution. Harness systems originated from software and coding agents, where language models are embedded in execution loops with access to tools, code interpreters, file systems, memory, tests, and feedback signals (Yao et al., 2022; Schick et al., 2023; Shinn et al., 2023). Such systems provide an external scaffold for tool invocation, state tracking, iterative refinement, and correction from intermediate feedback, and were later extended to executable skill libraries, multi-agent collaboration, and software-engineering agents (Wang et al., 2023; Wu et al., 2023; Hong et al., 2024; Wang et al., 2024; Yang et al., 2024; Wang et al., 2025; Xia et al., 2024). This idea has recently been extended to robotics (Li et al., 2026b), where VLAs are coupled with management and scheduling systems for data collection, execution, and self-improvement. Compared with coding agents, robotic harnesses must operate under physical-world constraints (Xiao et al., 2026), which inherently impose substantially greater requirements on the underlying models. Robotic agent systems are further challenged by policy heterogeneity: different policy families exhibit distinct capabilities, assumptions, representations, and execution regimes (Zhang et al., 2026; Dalal et al., 2023; Huang et al., 2026a; 2022), making policy selection and coordination nontrivial. Existing systems therefore mainly organize skills from the same policy family under fixed execution interfaces, implicitly assuming compatible representations and in-distribution handoffs (Li et al., 2026b; Xiao et al., 2026), without explicitly addressing heterogeneous policy orchestration or uncertain capability boundaries. RoboHarness addresses this gap with a harness tailored to capability-aware coordination among heterogeneous robot policies.

Long-Horizon Robot Planning and Task Decomposition. Long-horizon robot planning has been studied through hierarchical planning, task and motion planning, skill-based planning, and language-model-based planning (Silver et al., 2024; Liang et al., 2024; Athalye et al., 2024; Garrett et al., 2020; Liu et al., 2024; Oswald et al., 2024; Huang et al., 2025). These methods decompose high-level instructions into sequences of symbolic actions, primitive skills, or subtasks, reducing the complexity of solving long-horizon manipulation tasks. Task and motion planning methods

combine symbolic task reasoning with geometric feasibility checks with pre-designed abstract reusable skills (Garrett et al., 2021; Kaelbling & Lozano-Pérez, 2011). Recent language-model-based planners leverage semantic knowledge and instruction-following capabilities to generate task plans from natural language goals and adapt them to different task contexts (Zhao et al., 2023; Yang et al., 2025; Guan et al., 2023; Chen et al., 2024; Hu et al., 2023). However, these methods largely share the assumption of homogeneous skills or policies with predefined, clear, and non-overlapping capabilities, implicitly assuming that control can be transferred without additional mediation and that input and output conditions across policies are aligned. They therefore do not address long-horizon planning over heterogeneous policies, where task decomposition, policy assignment, and inter-policy handoff must be reasoned about jointly.

Policy Chaining and Spatially Consistent Handoff. Skill chaining studies the distribution shift when connecting short-horizon policies to solve long-horizon tasks, where handoffs between adjacent policies often become a bottleneck. Prior work has studied this distribution-mismatch problem by learning initiation sets (Konidaris & Barto, 2009; Huang et al., 2023), transition mechanisms (Lee et al., 2019; Sahni et al., 2017; Mishra et al., 2023), or terminal-state regularizers (Lee et al., 2021) to improve the compatibility between one policy’s terminal states and the next policy’s executable region. These methods demonstrate that long-horizon performance is governed not only by the reliability of individual policies, but also by the compatibility between one policy’s terminal states and the next policy’s executable region. However, these methods are primarily training-side solutions, requiring additional learning or policy regularization rather than online plug-and-play coordination. They also assume a given skill set within a shared policy family with relatively clear capability boundaries, leaving heterogeneous policy decomposition and routing largely unaddressed.

3 PROBLEM DEFINITION

We consider zero-shot long-horizon planning with a library of heterogeneous robot policies $\Pi = \{\pi_1, \pi_2, \dots, \pi_N\}$. Here, zero-shot refers to solving previously unseen long-horizon task compositions without task-specific joint training of the constituent policies or their orchestration. Let $\mathcal{X}$ denote the system state space, $\mathcal{O}$ the global observation space, and $\mathcal{G}$ the space of subtask instructions. Each policy π_i has a policy-specific input space $\mathcal{O}_i \subseteq \mathcal{O}$, containing the in-distribution observations under which π_i can be reliably executed, and an achievable subtask space $\mathcal{G}_i \subseteq \mathcal{G}$, containing the subtasks that π_i is capable of accomplishing. RoboHarness wraps each policy as an executable skill and maintains a policy-specific memory bank $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_N\}$, where $\mathcal{M}_i$ stores successful execution trajectories of π_i .

Given a high-level task instruction I , an initial state $\mathbf{x}_0 \in \mathcal{X}$, the policy library Π , and the memory bank $\mathcal{M}$, RoboHarness seeks a sequence of subtasks $\tau = (g_1, \dots, g_T)$, where $g_t \in \mathcal{G}$, a corresponding policy assignment $\rho = (\pi_{k_1}, \dots, \pi_{k_T})$, where $k_t \in \{1, \dots, N\}$, and a sequence of bridge trajectories $B = (b_1, \dots, b_{T-1})$ whose joint execution completes I . Let $o_t \in \mathcal{O}$ denote the observation provided to π_{k_t} at stage t . The assignment of π_{k_t} to g_t is valid only if $o_t \in \mathcal{O}_{k_t}$ and $g_t \in \mathcal{G}_{k_t}$. Let $\bar{o}_t$ denote the terminal observation produced by π_{k_t} . For each pair of consecutive policies, the bridge trajectory b_t , initialized from $\bar{o}_t$, produces the next-stage observation $o_{t+1} = \text{Terminal}(b_t) \in \mathcal{O}_{k_{t+1}}$. The problem is therefore to jointly perform task decomposition, capability-aware policy assignment, and inter-policy bridging such that the resulting execution completes I while every selected policy is assigned an achievable subtask and receives an in-distribution input.

4 METHODOLOGY

RoboHarness is an agentic orchestration framework in which a coding agent serves as the high-level planner and router for heterogeneous robot policies. The coding agent dynamically invokes three types of auxiliary skills: information understanding, memory, and self-evolution. It interprets their structured outputs to decompose long-horizon instructions, assign appropriate policies to subtasks, and coordinate the required inter-policy handoffs. As illustrated in Figure 1, these auxiliary skills do not replace the underlying control policies. Instead, they provide the information, memory, and adaptation mechanisms needed for capability-aware planning and stable policy orchestration across different execution distributions. RoboHarness preserves the native interfaces of existing policies,

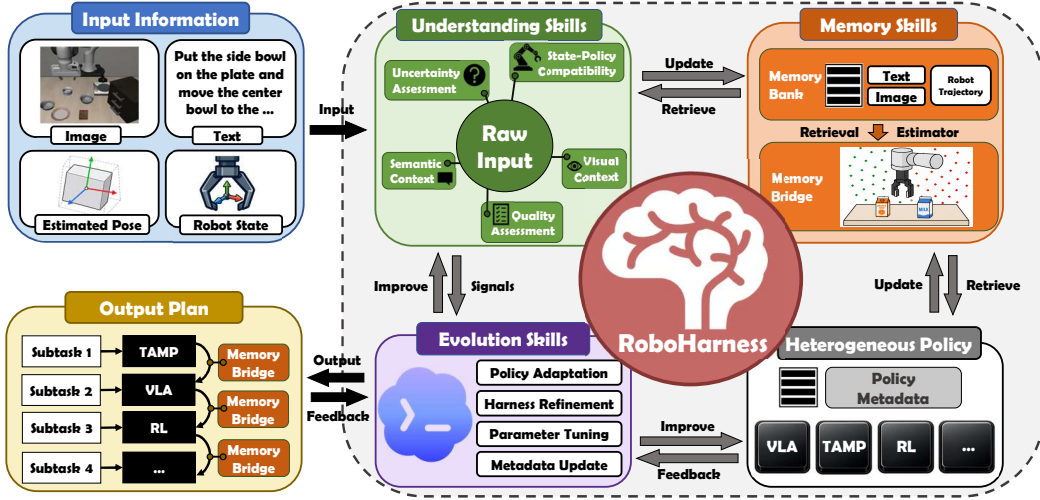


Figure 1: Overall Framework. RoboHarness integrates heterogeneous robot policies with three auxiliary skill libraries: Understanding Skills, Memory Skills, and Evolution Skills. Understanding Skills transform multimodal observations into rich, task-relevant information for planning and policy selection. Memory Skills manage, retrieve, and update execution histories to support policy coordination and handoffs. Evolution Skills use online execution evidence to update both individual policies and the harness, forming a closed information loop that enables capability-aware planning and stable orchestration of heterogeneous policies.

allowing independently developed policies to be combined without joint retraining or a shared action representation. The full implementation detail of the harness system is presented in Section 11.2

4.1 UNDERSTANDING SKILLS

Understanding skills further interpret the raw inputs available to RoboHarness, including the task instruction, visual observations, estimated object poses, robot states, and intermediate execution results. Their purpose is not limited to scene recognition. Instead, they extract decision-relevant information that reveals the relationship between the current task state and the capability boundaries of the available policies. RoboHarness currently includes five types of understanding skills and can be expanded when additional information is required. Uncertainty assessment computes the mean and variance of estimated object poses within a sliding time window to measure temporal stability. Visual-context assessment projects the current observation into a latent space and compares it with visual embeddings of policy-specific training trajectories. Semantic-context assessment similarly compares encoded task instructions and candidate subtasks with the language descriptions associated with each policy. State-policy compatibility assessment uses the spatial distribution reconstructed by the **Memory Bridge** to score the current end-effector pose and joint configuration, indicating whether the robot state lies within the next policy’s in-distribution region; the detailed construction and scoring procedure is described in Section 4.2. Input-quality assessment evaluates image sharpness, exposure, noise, and task-relevant object visibility to determine whether the current observation is sufficiently reliable. Together, these understanding skills provide structured, task-relevant evidence that is difficult to infer reliably from raw inputs alone. The coding agent invokes the relevant skills on demand and jointly reasons over their outputs to decompose the task and assigns each subtask to the most-suitable policy.

4.2 MEMORY SKILLS AND MEMORY BRIDGE

Memory skills consist of two components: multimodal memory management and memory-based inter-policy bridging.

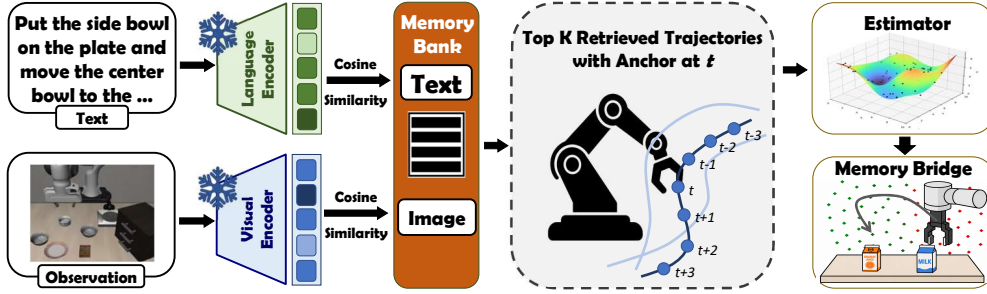


Figure 2: Memory Bridge. (1) The current observation and task instruction are embedded to retrieve the top-K most relevant in-distribution trajectories from memory. (2) Robot states from the retrieved trajectories are used to estimate a spatial state distribution, which scores how closely a state matches the next policy’s in-distribution region. (3) Candidate states are sampled and ranked by this score to generate a bridge trajectory toward a high-confidence state for the next policy.

4.2.1 MEMORY MANAGEMENT

The memory bank $\mathcal{M}$ is organized as a set of linked-node trajectories. A trajectory $\xi = (n_1, \dots, n_L) \in \mathcal{M}$ consists of L temporally ordered nodes, where each node n stores a subtask instruction $g(n) \in \mathcal{G}$, an observation $o(n) \in \mathcal{O}$, and a robot-state vector $\mathbf{s}(n) \in \mathbb{R}^{d_s}$, together with the corresponding text and visual embeddings. Here, d_s denotes the dimension of the robot-state representation, while $e_{\text{text}} : \mathcal{G} \rightarrow \mathbb{R}^{d_{\text{text}}}$ and $e_{\text{vis}} : \mathcal{O} \rightarrow \mathbb{R}^{d_{\text{vis}}}$ denote the text and visual encoders, respectively. Let $\mathcal{V}(\mathcal{M})$ denote the set of all nodes stored in $\mathcal{M}$. Given a query subtask $g \in \mathcal{G}$ and observation $o \in \mathcal{O}$, RoboHarness performs hierarchical retrieval over the nodes in $\mathcal{M}$. It first retrieves the nodes whose stored instructions are most semantically relevant to g :

$$\tilde{\mathcal{N}} = \text{TopK}_{K_{\text{text}}} \cos(e_{\text{text}}(g), e_{\text{text}}(g(n))),$$

$$n \in \mathcal{V}(\mathcal{M})$$

where K_{text} is the number of retrieved nodes. RoboHarness then compares the query observation with the visual embeddings of nodes in $\tilde{\mathcal{N}}$ and retrieves

$$\mathcal{N} = \text{TopK}_{K_{\text{vis}}} \cos(e_{\text{vis}}(o), e_{\text{vis}}(o(n))),$$

$$n \in \tilde{\mathcal{N}}$$

where K_{vis} is the number of retrieved nodes, TopK_K returns the K elements with the highest similarity values, and $\cos(\mathbf{a}, \mathbf{b})$ denotes cosine similarity. We denote the overall memory retrieval function by

$$\text{Retrieve}_{\mathcal{M}}(g, o) = \mathcal{N},$$

where $\mathcal{N}$ is the set of nodes returned by the hierarchical text–visual retrieval process described above. Information from these retrieved nodes is provided to downstream auxiliary skills and used for Memory Bridge construction.

Separately, RoboHarness maintains global historical statistics that summarize policy execution outcomes across different tasks, providing empirical evidence for capability estimation and policy assignment. Both the linked-node trajectories and the execution statistics are continuously updated after each rollout.

4.2.2 MEMORY BRIDGE

The Memory Bridge handles inter-policy transitions for which the terminal observation o_t produced by π_{k_t} does not lie within the input space $\mathcal{O}_{k_{t+1}}$ supported by $\pi_{k_{t+1}}$. Given the upcoming subtask g_{t+1} and o_t , RoboHarness retrieves a set of relevant anchor nodes as $\mathcal{N}_t = \text{Retrieve}_{\mathcal{M}+\infty}(g_{t+1}, o_t)$. Based on these nodes, the Memory Bridge constructs a local progress function and generates a bridge trajectory b_t that guides the robot toward a handoff state whose observation lies within $\mathcal{O}_{k_{t+1}}$ before invoking $\pi_{k_{t+1}}$. The procedure consists of three stages.

Spatial distribution construction. Let $\mathcal{N}_t = \{n_{1,0}, \dots, n_{K,0}\}$, where $K = |\mathcal{N}_t|$ and $n_{i,0}$ denotes the i -th retrieved node, which is regarded as an anchor node serving as the temporal reference

for trajectory expansion and progress scoring. RoboHarness expands each anchor along its linked trajectory over a temporal horizon $l \in \mathbb{N}$ in both the forward and backward directions. Let $n_{i,j}$ denote the node linked to $n_{i,0}$ at temporal offset $j \in \{-l, \dots, l\}$. Each anchor is assigned a score of zero, while its linked nodes are assigned scores according to their temporal offsets. Specifically, RoboHarness constructs the robot-state–score pairs

$$(\mathbf{s}_{i,j}, y_{i,j}) = (\mathbf{s}(n_{i,j}), j\Delta t), i \in \{1, \dots, K\}, j \in \{-l, \dots, l\},$$

where $\Delta t > 0$ denotes the time interval between adjacent nodes, $\mathbf{s}_{i,j} \in \mathbb{R}^{d_s}$ is the robot state stored in $n_{i,j}$, and $y_{i,j} \in \mathbb{R}$ is its assigned progress score. These pairs are used to train a lightweight progress estimator $f_{\text{score},t}$, where $f_{\text{score},t}(\mathbf{s})$ predicts the local progress of state $\mathbf{s}$. Let $\mathcal{S}_{\text{ret},t} = \{\mathbf{s}_{i,j}\}$ denote the expanded robot-state samples. RoboHarness then constructs the local support region $\mathcal{R}_{\text{conf},t} = \{\mathbf{s} \in \mathbb{R}^{d_s} \mid d(\mathbf{s}, \mathcal{S}_{\text{ret},t}) \leq \epsilon\}$, where $\epsilon > 0$ is the support threshold and $d(\mathbf{s}, \mathcal{S}_{\text{ret},t}) = \min_{\bar{\mathbf{s}} \in \mathcal{S}_{\text{ret},t}} \|\mathbf{s} - \bar{\mathbf{s}}\|_2$ denotes the distance from $\mathbf{s}$ to its nearest expanded memory state. The region $\mathcal{R}_{\text{conf},t}$ restricts the learned progress function to states locally supported by the retrieved memory.

Spatial state scoring. RoboHarness uses $f_{\text{score},t}$ to evaluate candidate robot states within $\mathcal{R}_{\text{conf},t}$. States outside this region may be traversed during bridge execution but are not considered valid handoff targets, since their progress estimates are not sufficiently supported by the retrieved memory. Within $\mathcal{R}_{\text{conf},t}$, the sign of $f_{\text{score},t}(\mathbf{s})$ indicates the estimated local progress of $\mathbf{s}$ relative to the retrieved anchors: a positive value indicates greater task progress, whereas a negative value indicates less task progress. Thus, $f_{\text{score},t}$ measures relative progress, while $\mathcal{R}_{\text{conf},t}$ defines the set of admissible handoff states supported by the next policy’s retrieved execution experience.

Bridge trajectory generation. Let $\mathbf{s}_t \in \mathbb{R}^{d_s}$ denote the post-execution robot state of π_{k_t} , and let $\mathcal{S}_{\text{plan},t} \subseteq \mathbb{R}^{d_s}$ denote the set of states that can be feasibly connected from $\mathbf{s}_t$ by the adopted motion planner. RoboHarness selects the handoff target robot state

$$\begin{aligned} \mathbf{s}_t^* &= \arg \max_{\mathbf{s}} [f_{\text{score},t}(\mathbf{s}) - \lambda_{\text{motion}} C_{\text{motion}}(\mathbf{s}_t, \mathbf{s})], \\ \text{s.t. } \mathbf{s} &\in \mathcal{R}_{\text{conf},t} \cap \mathcal{S}_{\text{plan},t} \quad \text{and} \quad f_{\text{score},t}(\mathbf{s}) \geq 0 \end{aligned}$$

where $C_{\text{motion}} : \mathbb{R}^{d_s} \times \mathbb{R}^{d_s} \rightarrow \mathbb{R}_{\geq 0}$ measures the motion cost between two robot states and $\lambda_{\text{motion}} \geq 0$ controls its contribution. The support constraint ensures that the selected target lies within the execution distribution represented by the retrieved memory, while the nonnegative-score constraint restricts the target to states exhibiting forward progress relative to the retrieved anchors. RoboHarness then invokes an off-the-shelf motion planning policy to generate a feasible bridge trajectory b_t connecting $\mathbf{s}_t$ to $\mathbf{s}_t^*$ before executing $\pi_{k_{t+1}}$. The resulting handoff is stored in $\mathcal{M}$ together with its associated instruction, visual observation, and robot-state trajectory, enabling its nodes to be retrieved through $\text{Retrieve}_{\mathcal{M}}$ and reused for similar inter-policy transitions.

4.3 SELF-EVOLUTION SKILLS

Self-evolution skills adapt RoboHarness using online execution evidence through four types of updates. **Policy adaptation** incorporates off-the-shelf adaptation methods to extend individual policy capabilities, such as SIMPACT (Liu et al., 2026) for grasp-pose adjustment and PDDL (Huang et al., 2025) for learning new logical representations online. **Harness refinement** employs the coding agent to modify the orchestration structure, including task-routing strategies, skill-invocation rules, and inter-skill coordination. **Parameter tuning** adjusts hyperparameters through grid search, including memory-retrieval settings, state-scoring thresholds, and bridge-acceptance criteria. **Metadata update** revises policy capability metadata based on newly observed successes and failures. All four self-evolution mechanisms are enabled in the simulated experiments, but updates are triggered only when the system encounters persistent execution failures rather than after every rollout. In such cases, the memory skills retrieve relevant failure histories and execution evidence and provide them to the coding agent, which determines whether policy adaptation, harness refinement, parameter tuning, metadata update, or a combination of these mechanisms should be applied. Validated updates are retained across evaluation episodes, allowing RoboHarness to accumulate experience and progressively adapt to new tasks, objects, scenes, and environments.

4.4 HETEROGENEOUS POLICIES AS AGENTIC SKILLS

RoboHarness wraps each heterogeneous robot policy as a callable agentic skill consisting of the policy itself and an associated policy card. Each policy retains its native implementation, inputs,

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 best and second-best results are shown in **bold** and underlined, respectively.

Model	Original	Robot State	Language	Layout	Background	Sensor	Camera	Light	Average
RoboHarness (ours)	98.7	90.4	97.0	86.8	97.1	<u>90.3</u>	87.6	97.4	93.2
$\pi_{0.5}$	96.9	75.4	85.6	<u>85.7</u>	94.6	<u>89.7</u>	75.4	<u>96.9</u>	<u>85.7</u>
π_0	94.2	61.0	63.5	76.4	79.0	80.1	61.0	85.0	53.6
UniVLA	95.2	46.2	77.6	31.9	81.0	21.2	1.8	69.0	42.9
OpenVLA-OFT	97.6	21.7	81.0	68.7	91.0	78.6	55.6	92.7	67.9
X-VLA	98.1	<u>89.7</u>	75.7	71.8	<u>96.0</u>	62.7	23.4	88.2	71.4
Cosmos-Policy	<u>98.5</u>	<u>63.3</u>	81.7	82.2	<u>88.9</u>	92.7	<u>75.8</u>	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	<u>98.5</u>	83.0	<u>86.4</u>	76.2	53.1	64.4	40.9	82.3	69.5

and outputs without requiring a shared action representation, while the policy card records its type, capabilities, interface requirements, constraints, assumptions, training tasks, and historical statistics that summarize policy execution outcomes. The coding agent can also directly inspect the policy’s implementation code to reason about capabilities and compatibility beyond the summarized metadata. This representation enables fundamentally different policies, such as VLAs, task-and-motion planners, and specialized RL policies, to be compared and composed at the planning level. As new execution evidence accumulates in memory, RoboHarness continuously updates each policy card to refine its capability boundaries and support subsequent planning and policy assignment.

5 EXPERIMENTAL SETUP AND BASELINES

We evaluate RoboHarness against diverse baselines on three public benchmarks and in 135 real-robot trials. The coding agent is implemented using Codex powered by GPT-5.5. Further implementation details, including the heterogeneous policy checkpoints and skill implementations, are provided in Section 11.2.

5.1 EXPERIMENTAL SETUP

Heterogeneous Policies in Simulated Benchmarks RoboHarness is equipped with three independently developed policies with complementary capabilities unless explicitly specified: (1) an open-source $\pi_{0.5}$ checkpoint trained on LIBERO-Spatial, LIBERO-Goal, LIBERO-Long, and LIBERO-Object (Black et al., 2025); (2) an open-source OpenVLA-OFT checkpoint (Kim et al., 2025), post-trained with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) on the LIBERO-90 tasks using RLinf (Zang et al., 2025); and (3) a TAMP planner designed for geometrically constrained pick-and-place tasks over a predefined object set.

Simulated Evaluation on Benchmarks We evaluate RoboHarness on three public benchmarks that examine complementary aspects of RoboHarness. LIBERO evaluates general task-following performance (Liu et al., 2023), while LIBERO-Plus evaluates out-of-distribution robustness and how accurately RoboHarness characterizes the capability boundaries of its constituent policies (Fei et al., 2025). LIBERO-LoHo targets zero-shot long-horizon planning, comprising tasks with an average horizon approximately four times longer than those in the original LIBERO benchmark (Huang et al., 2026a). In addition to the benchmarks described above, we designed In addition, we designed 500 more complex simulated tasks across 10 task types, each requiring the integration of multiple heterogeneous policies. Since many tasks in existing public benchmarks can be solved by a single policy, they do not fully evaluate the robustness of multi-policy integration or the stability of inter-policy handoffs. Full details of the tasks are available in Section 11.1.

Real Robot Experiments We further conducted 135 real-robot experiments across five challenging task classes and four type of disturbances, each requiring the coordinated use of multiple heterogeneous policies. Given the available blocks and a language instruction, RoboHarness must reason about the requested structure and generate an assembly plan for constructing structures such as a short bridge, a tall bridge, a Chinese character, or a boat. Some required blocks are initially hidden inside a cabinet, requiring the robot to open the cabinet, explore its contents, and retrieve the missing blocks. Our real-robot system integrates a TAMP planner with a VLA policy($\pi_{0.5}$): the TAMP planner inter-

Table 2: LIBERO-LoHo zero-shot 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	Average
RoboHarness (ours)	100.0 / 100.0	97.3 / 96.0	98.7 / 96.0	94.7 / 92.0	97.0 / 92.0	97.5 / 95.2
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
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
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
OpenVLA	0.0 / 0.0	2.7 / 0.0	24.0 / 0.0	2.7 / 0.0	0.0 / 0.0	5.9 / 0.0

prets the request and generates the corresponding assembly plan, while the VLA handles contact-rich cabinet-opening and cabinet-closing operations that are difficult to model within TAMP. To further evaluate robustness, we introduce four types of execution-time disturbances: hiding required blocks again during execution, breaking the task progress by dismantling a partially completed structure, injecting perception noise into pose estimates, and introducing distracting objects into the scene.

5.2 BASELINES

We compare RoboHarness with a comprehensive set of baselines spanning three representative methodological categories. The VLA baselines comprise $\pi_{0.5}$ (Black et al., 2025), π_0 (Black et al., 2024), UniVLA (Bu et al., 2025), X-VLA (Zheng et al., 2025), OpenVLA-OFT (Kim et al., 2025), and GR00T-N1.5 (Bjorck et al., 2025). We further consider three world-action models, including LingBot-VA (Li et al., 2026a), Fast-WAM (Yuan et al., 2026), and Cosmos Policy (Kim et al., 2026). We also include three hierarchical planning approaches, H-WM (Huang et al., 2026a), LLM-guided VLA (Shi et al., 2025), and logic-guided VLA (Huang et al., 2026a). Together, these comparisons position RoboHarness against both end-to-end policies and methods explicitly designed for long-horizon decision-making. The evaluation results were obtained from the benchmark evaluation of Huang et al. (2026a) and Zhang et al. (2026)

6 RESULTS

In this section, we present the evaluation results to answer the following questions. RQ1: Can RoboHarness effectively coordinate heterogeneous policies to accomplish diverse robotic tasks? RQ2: Can RoboHarness enable zero-shot generalization to long-horizon robotic tasks requiring the composition of heterogeneous policies? RQ3: Can RoboHarness improve robustness under diverse out-of-distribution perturbations? RQ4: Can RoboHarness accurately characterize the capability boundaries of heterogeneous policies and dynamically route subtasks accordingly?

RQ1: Coordination of heterogeneous policies. We first evaluate RoboHarness on the original LIBERO benchmark, which contains diverse language-conditioned manipulation tasks. As shown in the Original column of Table 1, RoboHarness achieves a success rate of 98.7%, outperforming all baselines. Notably, RoboHarness coordinates $\pi_{0.5}$ and OpenVLA-OFT, surpassing these two constituent policies by 1.8 and 1.1 percentage points, respectively. This improvement shows that RoboHarness effectively exploits their complementary capabilities through capability-aware policy selection and composition.

RQ2: Zero-shot long-horizon generalization. Table 2 reports the results on LIBERO-LoHo, where each task contains a substantially longer sequence of dependent subtasks and requires complementary capabilities beyond those of any single underlying policy. RoboHarness achieves the best result on every task, with an average progress score of 97.5% and an average success rate of 95.2%, outperforming all baselines. RoboHarness also substantially outperforms all of its underlying policies. In particular, the nonzero progress scores of $\pi_{0.5}$ and OpenVLA-OFT demonstrate their ability

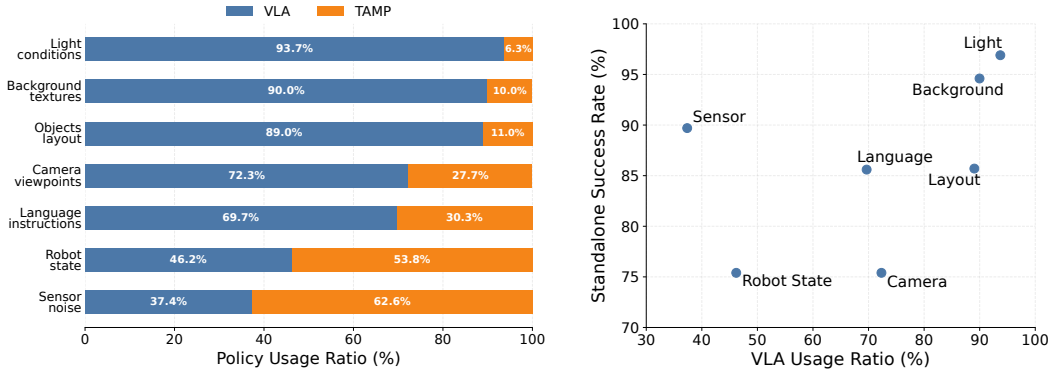


Figure 3: (a) Usage ratios of the underlying policies across LIBERO-Plus perturbation categories. (b) Relationship between the VLA invocation ratio and its standalone performance across perturbation categories.

to complete individual subtasks, while their low success rates show that completing long-horizon tasks requires these complementary capabilities to be properly composed. High-level guidance improves $\pi_{0.5}$ but remains substantially below RoboHarness, indicating that task decomposition alone cannot overcome missing policy capabilities or incompatible intermediate states. RoboHarness addresses both limitations through capability-aware routing and reliable inter-policy handoffs, enabling heterogeneous policies to solve tasks that none can complete independently.

RQ3: Robustness to perturbations. Table 1 reports performance under seven perturbation categories in LIBERO-Plus. RoboHarness achieves the highest average success rate of 93.2% and ranks first in six of the seven categories. It also consistently outperforms its underlying policies, $\pi_{0.5}$ and OpenVLA-OFT, demonstrating that RoboHarness improves their out-of-distribution task-execution robustness rather than merely inheriting their individual capabilities. The particularly large improvement under robot-state perturbations demonstrates that the Memory Bridge can guide out-of-distribution robot states back toward the VLA’s training distribution. The improvement under language perturbations shows that the understanding skills can interpret ambiguous or imprecise instructions and translate them into subgoals compatible with the VLA’s capabilities. Similarly, the gain under camera-viewpoint changes indicates that RoboHarness can interpret viewpoint shifts and dynamically adjust policy routing according to each policy’s robustness to such changes. The improvements are smaller under sensor noise because this perturbation directly degrades the input images, limiting the ability of the understanding skills to extract reliable task information. Gains under layout perturbations are also constrained because the benchmark includes cases in which objects fall outside the robot’s reachable workspace, making task completion impossible regardless of the system’s decisions.

RQ4: Capability-aware policy routing. We conduct an additional experiment on LIBERO-Plus to examine whether RoboHarness can characterize context-dependent policy capability boundaries and adapt its routing decisions accordingly. All tasks in this benchmark are within the nominal capability of $\pi_{0.5}$ and can be completed by the policy under suitable execution conditions. To isolate the relationship between the VLA’s standalone capability and its invocation frequency, we use only $\pi_{0.5}$ and the TAMP system as the underlying heterogeneous policies. This setup allows variations in the VLA usage ratio to primarily reflect changes in its reliability under different perturbations rather than differences in task coverage. As shown in Figure 3(a), RoboHarness substantially adjusts policy usage across perturbation categories. It invokes $\pi_{0.5}$ most frequently under lighting, background, and layout changes, where the policy exhibits strong standalone robustness, while relying more heavily on TAMP under robot-state, camera-viewpoint, and language-instruction perturbations, where $\pi_{0.5}$ performs less reliably. This routing pattern reflects the context-dependent strengths and limitations of the underlying policies rather than a fixed invocation rule. Figure 3(b) further shows a clear positive relationship between the VLA invocation ratio and its standalone success rate across perturbation categories. Sensor noise is the primary outlier because degraded visual observations limit the reliability of the information available to the coding agent for capability assessment and

routing. Overall, these results demonstrate that RoboHarness can infer how policy reliability changes with the execution context and dynamically route subtasks to the policy better suited to the current perturbation.

7 ABLATION STUDY

RQ5: How does each component contribute to the performance of RoboHarness?

RQ5: Contribution of System Components.

We conducted the ablation study on 500 custom-designed, long-horizon manipulation tasks that require the integration of heterogeneous underlying policies for successful completion. Results in Figure 4 reveal that the three auxiliary modules address distinct failure modes in heterogeneous policy orchestration. Removing the Understanding Skills causes the most fundamental degradation because incomplete task and scene interpretation propagates into incorrect decomposition and policy assignment. Without the Evolution Skills, RoboHarness can still perform initial planning but cannot refine its estimates of policy capability boundaries from execution experience, making it prone to repeated routing errors under unfamiliar conditions. In contrast, removing the Memory Bridge preserves relatively strong task progress but frequently prevents full task completion, indicating that appropriate policy selection does not guarantee that the output state of one policy is suitable for the next. The Memory Bridge therefore primarily converts correct high-level orchestration into reliable execution by stabilizing inter-policy handoffs. Finally, the single-policy variants can solve only the task segments aligned with their individual capabilities, confirming that long-horizon performance arises from both the complementary capabilities of heterogeneous policies and the auxiliary modules that understand, coordinate, and adapt their use.

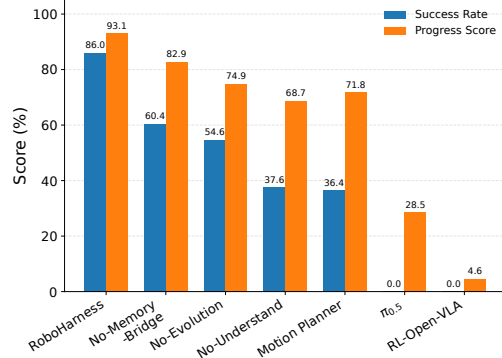


Figure 4: Success rate and progress score of the ablated systems.

8 REAL ROBOT EXPERIMENTS

We conducted 135 real-robot experiments: 15 for each of five target structures and 15 under each of four disturbance settings. In each construction trial, one to three required blocks were randomly hidden inside a cabinet, requiring RoboHarness to identify and retrieve them using the VLA before transferring control to TAMP for assembly. As is shown in Figure 5, RoboHarness performed consistently across all structures, with relatively lower success on the Taller Bridge due to its structural complexity and physical instability. We further evaluated the four disturbances on the Bridge task. Re-hiding blocks caused the largest degradation, reducing success from 86.7% to 66.7%, as repeated exploration and replanning frequently exceeded the time limit. RoboHarness retained 80.0% success after a partially completed structure was dismantled, demonstrating online reactivity and replanning. Under 5%–10% random errors in object-pose estimates, it achieved 73.3% success, showing tolerance to moderate perception errors. Distracting blocks had minimal effect, with success remaining at 86.7%, indicating robust identification of task-relevant objects.

9 LIMITATIONS AND FUTURE WORK

RoboHarness is constrained by the collective capabilities of its underlying policy library: orchestration can combine complementary strengths but cannot solve subtasks that fall outside the capabilities of all available policies. Moreover, capability characterization and Memory Bridge construction rely on accumulated execution evidence and may be unreliable when a policy is newly introduced or relevant experience is sparse. Although the current implementation uses a fixed set of heterogeneous policies and auxiliary skills, RoboHarness is policy-agnostic and extensible to other policy families. Future work will expand the policy library to include more diverse systems, such as navigation

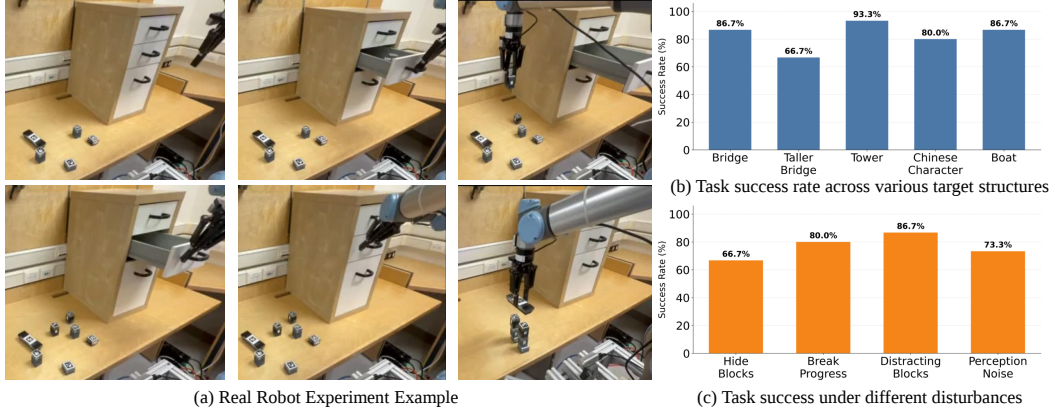


Figure 5: (a) Real robot experiment examples. (b) Overall success Rate of various target structure. (c) Overall success rate under different disturbances

policies, model predictive controllers, and world-action models. Another direction is to automate the integration of new policies, enable the discovery of new auxiliary skills, and support online policy training for previously unsupported scenarios, allowing RoboHarness to continually expand its capabilities. Incorporating additional sensory modalities, such as tactile feedback, could further improve capability characterization and support scalable deployment in open-world environments.

10 CONCLUSION

We presented RoboHarness, a unified framework for orchestrating heterogeneous robot policies in zero-shot long-horizon manipulation. RoboHarness represents independently developed policies as callable agentic skills and employs Understanding, Memory, and Evolution Skills to interpret the current context, characterize policy capability boundaries, select complementary policies, and refine orchestration using execution experience. The Memory Bridge further mediates distribution mismatch between consecutive policies, enabling their native implementations to be chained without joint retraining or a shared action representation. Evaluations on LIBERO, LIBERO-Plus, LIBERO-LoHo, 500 custom-designed long-horizon tasks, and 135 real-robot trials demonstrate that heterogeneous policies become substantially more effective when supported by capability-aware planning and stable inter-policy handoffs. These results position RoboHarness as a promising path toward integrating specialized robot intelligence into adaptive and general-purpose long-horizon systems.

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 vision-language models. arXiv preprint arXiv:2501.00296, 2024.
- Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. arXiv preprint arXiv:2503.14734, 2025.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0 : A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164, 2024.
- 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, 2025.
- Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. arXiv preprint arXiv:2505.06111, 2025.
- 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.
- 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. Proc. Int. Conf. Autom. Plan. Sched., 2020.
- 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, 2021.
- 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.
- 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.
- J Hoffmann and B Nebel. The FF planning system: Fast plan generation through heuristic search. arXiv [cs.AI], June 2011.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In International Conference on Learning Representations, volume 2024, pp. 23247–23275, 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. arXiv preprint arXiv:2310.08582, 2023.

- 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. arXiv preprint at arXiv:2505.18382, May 2025.
- 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, 2026b. URL <https://arxiv.org/abs/2509.21543>.
- 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, 2023.
- 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 Proc. Conf. on Robot Learning, 2022.
- 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.
- 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. arXiv preprint arXiv:2502.19645, 2025.
- 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.
- 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. arXiv preprint arXiv:2111.07999, 2021.
- Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. arXiv preprint arXiv:2601.21998, 2026a.
- 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. arXiv preprint arXiv:2410.23156, 2024.

- 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, 2026.
- Weiyu Liu, Neil Nie, Ruohan Zhang, Jiayuan Mao, and Jiajun Wu. Learning compositional behaviors from demonstration and language. In 8th Annual Conference on Robot Learning, 2024.
- 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.
- 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.
- 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 Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. arXiv preprint arXiv:2502.19417, 2025.
- 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. arXiv preprint arXiv:2305.16291, 2023.
- 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, 2024.
- 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, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. arXiv preprint arXiv:2308.08155, 2023.

- Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. [arXiv preprint arXiv:2407.01489](#), 2024.
- Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, et al. Empire: 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.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. [arXiv preprint arXiv:2210.03629](#), 2022.
- 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, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, et al. Rlinf-vla: A unified and efficient framework for vla+ rl training. [arXiv preprint arXiv:2510.06710](#), 2025.
- 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](#), 2026.
- 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.

11 APPENDIX

11.1 CUSTOMIZED TASKS

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

11.1.1 SIMULATED TASKS

We design 500 simulated task instances with longer execution horizons, ambiguous instructions, relational or state-dependent instructions, and frequent switching among heterogeneous policies. These tasks combine object rearrangement, articulated-object manipulation, spatial reasoning, multi-object handling, and state-changing interactions. These tasks are designed to evaluate the contribution of each system component when no single underlying policy can accomplish the complete task. The results are reported in the ablation study in Section 7. There are 10 total classes for the tasks, each class includes 50 randomly initialized tabletop configurations. The details of the ten classes 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.

11.1.2 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. 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 target structure for

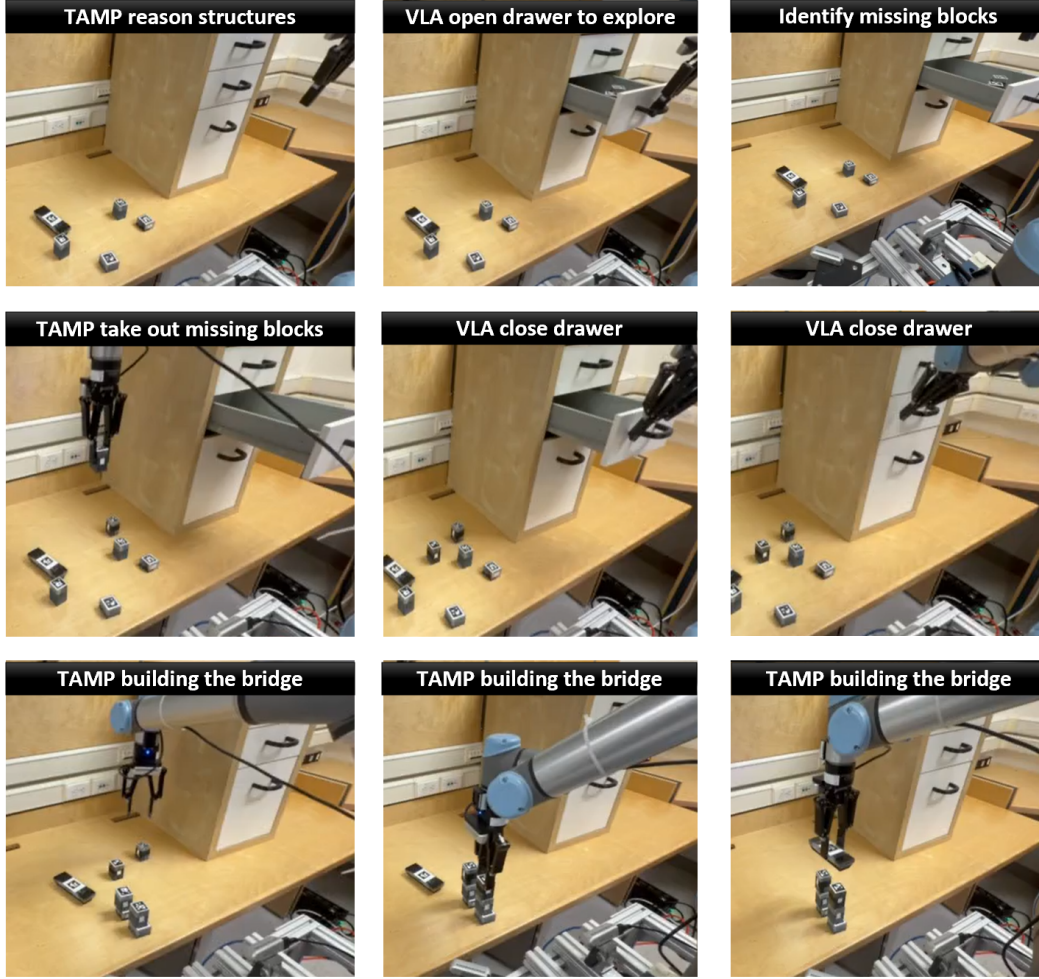


Figure 6: Full process of the real-robot experiment

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; (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. The TAMP policy uses the UR5e low-level control API for robot execution, 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 is responsible for reasoning about the required blocks and executing 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.

11.2 DETAILS OF AUXILIARY SKILL MODULES

11.2.1 UNDERSTANDING SKILLS

- **Uncertainty Assessment.** Aggregates perception outputs over a temporal window to estimate the mean and variance of object poses. These uncertainty statistics are provided to the coding agent to support uncertainty-aware planning and decision-making.
- **Visual Context.** Uses two frozen visual encoders, (1) `google/siglip2-base-patch16-224`, (2) `facebook/dinov2-base`, to extract visual embeddings for comparison with those stored in memory. The similarities computed by the two encoders are combined through a weighted sum, producing the final similarity score used as the retrieval key. The above models are pulled from Hugging Face.
- **Semantic Context.** Uses a frozen language encoder, `BAAI/bge-large-en-v1.5`, to extract semantic embeddings from textual context and compare them with semantic embeddings stored in memory. The above models are pulled from Hugging Face.
- **Quality Assessment.** Evaluates observation quality using conventional image-processing algorithms. Exposure is measured from the mean grayscale intensity and the proportions of underexposed and overexposed pixels, whose intensities fall below or above predefined thresholds. Image sharpness is quantified by the variance of the Laplacian response, while contrast is measured by the standard deviation of grayscale intensities. Image entropy is additionally computed from the intensity histogram to estimate the amount of visual information. 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.** Constructs a Memory Bridge for each candidate policy and evaluates the compatibility of the current robot state with the policy’s execution distribution.

11.2.2 MEMORY SKILL IMPLEMENTATION DETAILS

The memory skill library mainly consists of three components: memory retrieval, memory update, and the Memory Bridge. The key mechanisms underlying memory retrieval and the Memory Bridge are described in the main paper. In this section, we provide further implementation details on the data structure used for memory update and present a representative run-time example of the Memory Bridge.

Memory Data Structure. As illustrated in Figure 7, the memory is organized as a collection of linked-node trajectories, with each trajectory recording a complete policy rollout from its initial observation to its terminal state. Nodes are connected in temporal order, allowing the system to recover both the preceding and subsequent execution context from any retrieved node. Each node stores the current visual observation, active task prompt, robot state, and their corresponding embeddings. The

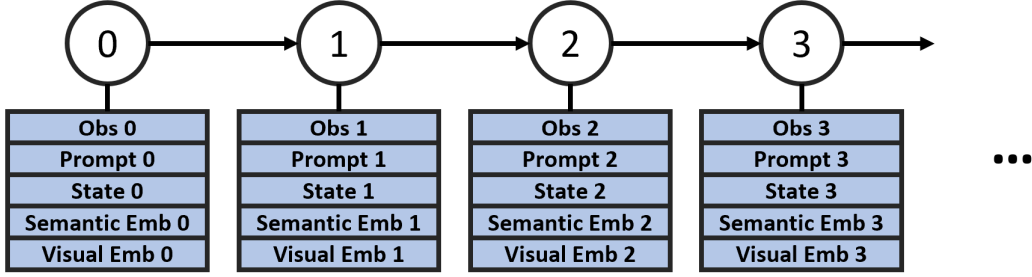


Figure 7: Memory Data Structure

robot state includes the joint configuration and end-effector pose. Visual embeddings extracted by the frozen SigLIP2 and DINOv2 encoders and semantic embeddings extracted by the frozen BGE encoder are stored alongside the raw data to support efficient multimodal retrieval. After each rollout, the recorded nodes are linked chronologically and added to the memory as a new trajectory.

Memory Bridge Example. Figure 8 presents a representative run-time example of the Memory Bridge. The practical progress estimator algorithm adopted in our experiment is Pairwise Ranking (SVM). When control is transferred between two policies, the terminal robot state produced by the preceding policy may be incompatible with the execution distribution of the subsequent policy. The Memory Bridge retrieves relevant trajectories associated with the next subtask and uses their robot states to characterize the state distribution preferred by the incoming policy. It then scores the current robot state against this distribution and generates a bridge trajectory toward a state with higher compatibility. The resulting state provides the incoming policy with a familiar initialization, enabling a stable policy handoff without jointly retraining the heterogeneous policies.

11.2.3 EVOLUTION SKILL IMPLEMENTATION DETAILS

The evolution skill library contains four components: policy adaptation, harness refinement, parameter tuning, and metadata update. These components use execution outcomes and retrieved experience to improve both the underlying policies and their orchestration within RoboHarness.

- **Policy Adaptation.** We implement SIMPACT (Liu et al., 2026) and PDDLML (Huang et al., 2025) as policy-adaptation methods for the TAMP system. SIMPACT adapts motion-level parameters, such as object grasp poses, based on execution feedback, while PDDLML extends the symbolic task model by learning new logical representations from observed interactions. These methods enable the TAMP system to handle previously unsupported objects, relations, and task conditions.
- **Harness Refinement.** RoboHarness uses Codex with GPT-5.5 to refine the 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 optimizes policy-routing and coordination logic. The modified implementation is evaluated through subsequent execution before being retained.
- **Parameter Tuning.** Each tunable continuous parameter is discretized into a set of bins forming a search grid. Instead of directly modifying its value, the coding agent only predicts a movement direction on the grid, such as increasing, decreasing, or retaining the current setting. A separate tuning module converts this decision into the corresponding grid value and applies the update. Tunable parameters include the hyperparameters of all underlying policies and the Memory Bridge, such as retrieval settings, state-scoring thresholds, and bridge-trajectory acceptance criteria.
- **Metadata Update.** This component continually refines each policy’s capability metadata using newly observed successes and failures. It records the task and execution contexts in which each policy performs reliably or fails, including relevant objects, scenes, task types, and environmental conditions. The updated metadata is stored in memory and used in

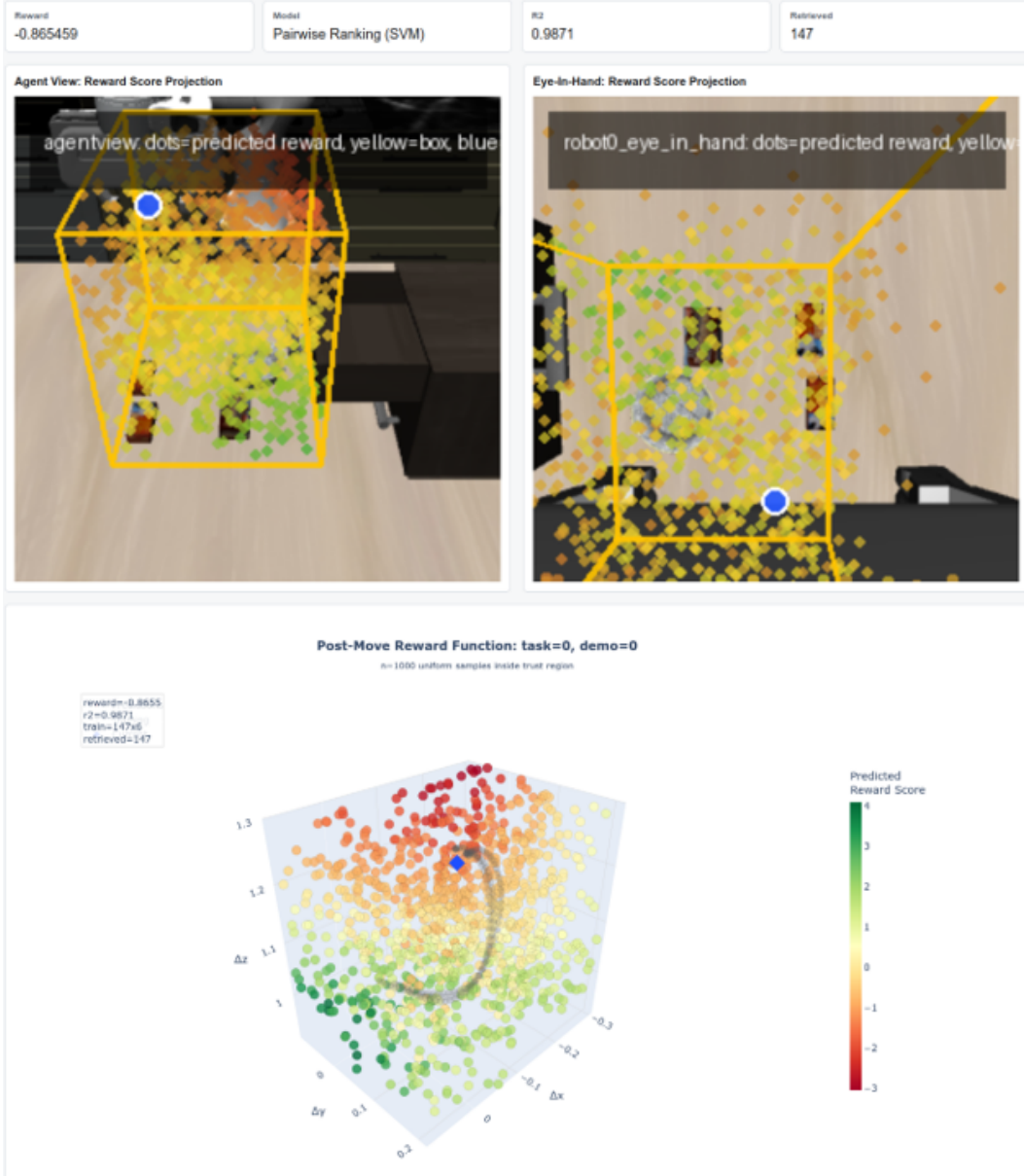


Figure 8: Memory Bridge Example

subsequent planning and policy-routing decisions to progressively characterize each policy’s capability boundaries.

11.2.4 UNDERLYING HETEROGENEOUS POLICIES

We incorporate three policies that were developed and trained independently and exhibit complementary capabilities:

- $\pi_{0.5}$. We use the official `pi05_libero` checkpoint released by Physical Intelligence (<https://github.com/Physical-Intelligence/openpi>). This checkpoint was fine-tuned on LIBERO-Spatial, LIBERO-Goal, LIBERO-Long, and LIBERO-Object, providing broad language-conditioned manipulation capabilities (Black et al., 2025).

- **OpenVLA-OFT.** We use the `RLinf/RLinf-OpenVLAOFT-GRPO-LIBERO-90` checkpoint released on Hugging Face. This checkpoint was post-trained with GRPO on the 90 short-horizon tasks in LIBERO-90 using the RLinf reinforcement-learning framework.
- **TAMP Planner.** We implement a TAMP planner for geometrically constrained pick-and-place tasks over a predefined object set. At the task level, we use the Planning Domain Definition Language (PDDL) to represent object types, predicates, and manipulation actions with their preconditions and effects, while each problem instance specifies the available objects, initial state, and task goal. Our implementation uses PDDL for symbolic task representation and the Fast-Forward (FF) planner (Hoffmann & Nebel, 2011) with Python interface provided by PDDLStream (Garrett et al., 2020), which connects symbolic PDDL planning with Python-based sampling procedures for continuous quantities such as grasp poses, placement poses, and robot configurations. The planner generates high-level actions, such as picking, placing, stacking, and aligning objects, and grounds each action through motion planning using the estimated object poses. This combination supports precise and interpretable manipulation but is less flexible for contact-rich interactions and objects not covered by the predefined domain.

Together, these policies provide complementary semantic, reactive, and model-based planning capabilities for evaluating heterogeneous policy orchestration.