HARNESSWAM: BRIDGING PREDICTION AND DELIBERATION IN WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) jointly learn environmental dynamics and robot actions, introducing priors over physical evolution into embodied control. However, finite-horizon prediction and action generation are insufficient for complex embodied tasks that require global planning, cross-stage state maintenance, execution verification, and failure recovery. We refer to this mismatch as the *prediction-deliberation gap*. To address this gap, we propose **HarnessWAM**, an agentic framework for WAMs. HarnessWAM employs a vision-language-model-based Task Manager to maintain an evidence-grounded scene belief and organize the global instruction into a structured task graph. Capability-conditioned executable-space projection then compiles the graph into WAM-supported skill sequences that satisfy task dependencies and embodiment-state constraints. During execution, HarnessWAM couples a lightweight progress estimator with event-triggered Task Manager deliberation. The estimator supplies frequent execution cues, while the Task Manager uses observations, task state, and interaction history to verify outcomes and decide whether to advance, acquire evidence, revise the plan, or recover. Local recovery preserves acquired scene knowledge so that execution can resume after failure. HarnessWAM achieves state-of-the-art average performance on RoboMemArena, with full-task and subtask success rates of 64.4% and 76.8%, respectively, and reaches 23.7% success on RoboCerebra Ideal. On a physical UR3 arm, full-task success reaches 95% for shape-conditioned tabletop tidying and 90% for memory-dependent box exploration. These results demonstrate the effectiveness of model-external agentic orchestration in bridging the prediction-deliberation gap for complex embodied tasks.

1 INTRODUCTION

World Action Models (WAMs) jointly model future observations and robot actions, connecting predictions of physical dynamics to embodied control (Li et al., 2026; Ye et al., 2026). These learned dynamics support manipulation and generalization to unfamiliar tasks and environments (Ye et al., 2026). However, complex tasks require decisions that extend beyond the model's prediction horizon. Placing a cookie in a non-empty drawer requires inspecting drawers, remembering their contents after closure, selecting the target, and verifying placement. A failed grasp further requires recovering the robot and resuming execution while retaining the evidence identifying the drawer. These requirements connect local action generation to persistent state, planning, verification, and recovery. We refer to this mismatch between predictive control and task-level deliberation as the *prediction-deliberation gap*, illustrated in Figure 1.

Recent efforts address aspects of this gap through persistent visual memory (Yang et al., 2026; Wang et al., 2026) and hierarchical skill prediction (Sun et al., 2026). These mechanisms provide policies with historical context and temporal structure for skill execution. However, access to history and skill predictions does not ensure that target selection is grounded in observations, that operations satisfy their prerequisites, or that attempted actions achieve their intended effects. Language-model agents suggest a harness that maintains state and incorporates feedback across model calls (Yao et al., 2022; Shinn et al., 2023; Yang et al., 2024). However, extending these harnesses to robots requires handling incomplete observations, physical constraints, and uncertain action outcomes. An embodied harness must therefore preserve task knowledge, ground semantic plans in supported skills, and verify physical effects before advancing execution.

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.

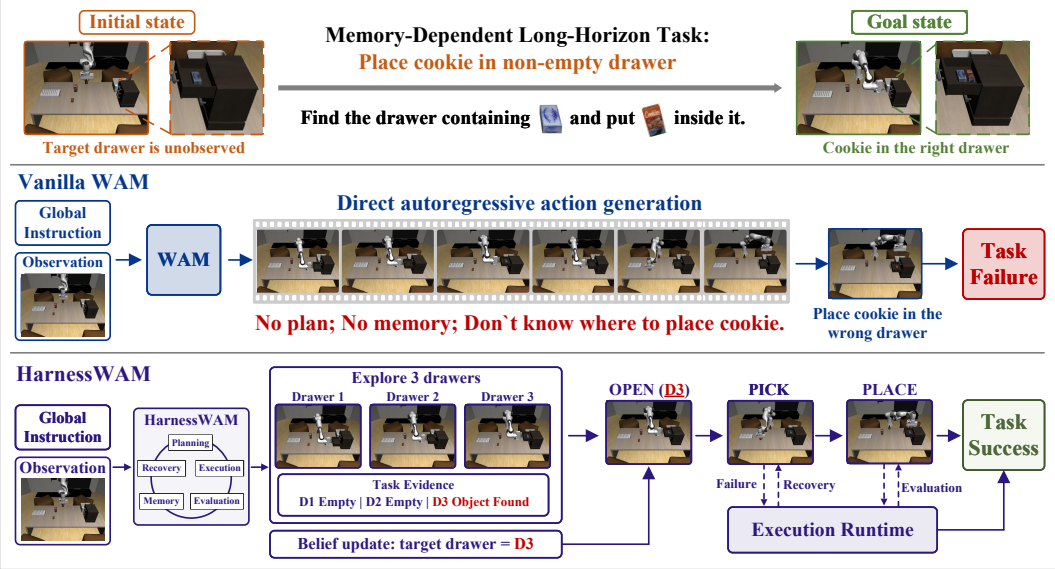

Figure 1: The prediction-deliberation gap. Vanilla WAM denotes direct execution of the global instruction without external task state. HarnessWAM retains drawer observations, binds the target after occlusion, and coordinates skill execution, verification, and recovery.

To address this gap, we introduce **HarnessWAM**, an agentic framework for WAMs. HarnessWAM employs a vision-language-model (VLM) *Task Manager* to maintain evidence-grounded scene belief and construct a task graph containing physical and cognitive operations. The graph includes evidence acquisition before unresolved targets are bound, allowing later skills to use targets identified through exploration. *Capability-conditioned executable-space projection* then compiles the graph into validated WAM skills, checking dependencies against represented scene and robot constraints. Since admissible plans can still fail during execution, HarnessWAM couples a lightweight progress estimator with event-triggered semantic verification. Verification updates the task state and determines whether to advance, observe, revise, or recover. Revised suffixes undergo projection while executed history remains fixed. Recovery returns the robot toward its saved configuration while retaining evidence for subsequent decisions. The same task state therefore connects initial planning to verification, revision, and recovery throughout execution.

HarnessWAM achieves state-of-the-art performance on RoboMemArena, with full-task and subtask success rates of 64.4% and 76.8%, and reaches 23.7% SR on RoboCerebra Ideal. Gains with FastWAM further demonstrate applicability across WAM backbones. On a UR3 arm, full-task success reaches 95% for tabletop tidying and 90% for box exploration.

Our main contributions are:

- We identify the prediction-deliberation gap and introduce a WAM harness that connects local predictive control to persistent task-level deliberation.
- Evidence-grounded task graphs link target binding to information acquisition; capability-conditioned projection checks initial and revised plans against execution constraints.
- An event-driven runtime uses semantic verification to govern graph transitions, preserving executed history and acquired evidence across plan revision and recovery.
- State-of-the-art results across RoboMemArena, RoboCerebra Ideal, and UR3 experiments demonstrate the effectiveness of HarnessWAM.

2 Related Work

World Models and World Action Models. World models learn environmental dynamics to predict how candidate actions change future observations. Visual foresight uses image predictions for model

predictive control (Ebert et al., 2017; 2018b), while Dreamer learns latent dynamics and trains policies on imagined trajectories (Hafner et al., 2025). Video-based policies transfer learned visual dynamics to action generation (Du et al., 2023; Wen et al., 2024; Liao et al., 2026), while WAMs jointly learn these dynamics and robot actions (Li et al., 2026; Ye et al., 2026; Yuan et al., 2026).

Recent WAMs extend this predictive foundation to longer interactions. MemoryWAM combines recent observations, anchor frames, and compressed history (Yang et al., 2026), while DiM-WAM uses event memory with progress supervision (Wang et al., 2026). HiMem-WAM couples hierarchical skill prediction with boundary-gated memory (Sun et al., 2026). AHA-WAM separates slower context updates from fast action generation (Cai et al., 2026). Together, these mechanisms extend the temporal context available for action generation and skill selection. However, this context alone does not ensure that an operation is executable or that its prerequisites have been achieved. HarnessWAM addresses this gap through evidence-grounded task graphs, capability-conditioned projection, and outcome verification that governs subsequent execution.

Harnesses for Embodied Agents. Robot agents ground language plans through skill affordances (Ahn et al., 2022) and geometric representations (Rana et al., 2023; Huang et al., 2023), with task-and-motion planning formalizing physical feasibility (Garrett et al., 2021). During execution, observed outcomes guide replanning (Huang et al., 2022). In concurrent work, Harness VLA combines such feedback with task-specific traces and global rules to coordinate frozen VLA skills and analytic primitives (Zhang et al., 2026). Complementary work addresses stage-aware progress estimation for policy training (Chen et al., 2026) and runtime failure detection (Agia et al., 2024). Recovery mechanisms then support continued execution after failure through local correction and additional attempts (Ebert et al., 2018a; Zhao et al., 2026).

However, combining these mechanisms for WAM execution leaves a consistency problem when new evidence or failure changes the plan. Revised operations must respect acquired knowledge, skill preconditions, and completed execution history. HarnessWAM makes these dependencies explicit in an evidence-grounded task graph. Evidence acquisition resolves symbolic targets, capability-conditioned projection checks motor nodes against the validated WAM interface, and outcome verification governs dependent execution. When observations require replanning, suffix revision preserves executed node definitions and rechecks future operations. This coupling maintains consistency between task knowledge and execution constraints as the plan evolves.

3 METHOD

3.1 PROBLEM FORMULATION

Given a global instruction x, the robot receives dual-view RGB observations o_t and proprioception q_t at environment step t. A WAM W_θ produces an H-step action chunk $A_t = (a_t, \ldots, a_{t+H-1})$ conditioned on its available observation history and a local instruction g_k. The task-event index k advances when execution calls for a task-level decision. To connect successive local skills, HarnessWAM maintains a runtime state $z_k = (B_k, G_k, M_k, r_k)$ and updates it at event time τ_k:

$$A_t = W_\theta(o_{\le t}, q_t, g_k), \qquad (z_{k+1}, g_{k+1}) = \mathcal{H}(x, z_k, o_{\tau_k}, e_k), \tag{1}$$

where B_k, G_k, and M_k denote scene belief, task graph, and memory. The execution state r_k tracks the active node, its attempt, and the remaining budget; e_k is the triggering event. The harness selects a local instruction and updates task state from its outcome. Between events, the WAM generates action chunks for that instruction, as shown in Figure 2.

3.2 EVIDENCE-GROUNDED TASK STATE

Task-level decisions require both scene knowledge and the history of attempted operations. Scene belief B_k records entities, attributes, and relations with confidence, supporting RGB evidence, and an epistemic status of observed, inferred, or unknown. Established facts, such as the contents of a closed drawer, remain available after occlusion. Task memory $M_k = (M_k^{\text{task}}, M_k^{\text{evidence}})$ stores completed and failed nodes, retry counts, bindings, and revision history alongside salient RGB evidence. The Task Manager updates these records at event times, making earlier observations and outcomes available for subsequent decisions.

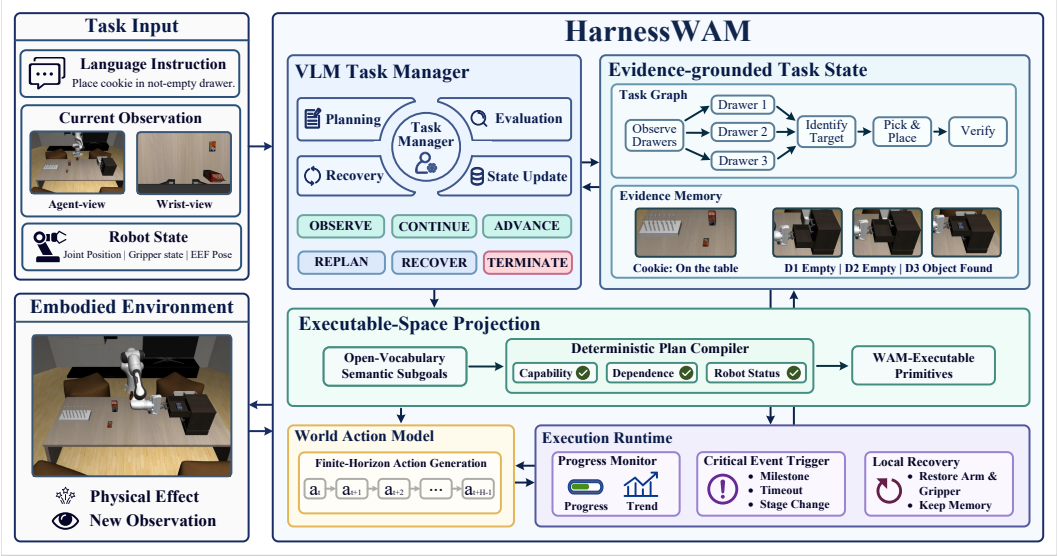

Figure 2: HarnessWAM. The Task Manager maintains evidence-grounded state and a task graph. Capability projection compiles semantic goals into supported skills. Progress events connect the fast WAM loop to verification, future-plan revision, and local recovery.

Using this state, the Task Manager encodes the instruction in a graph $G_k = (V_k, E_k, \mathcal{X}_k, \beta_k)$ of operations, dependencies, unresolved variables, and a partial binding. Each node is represented as

$$v_i = (\mathrm{op}_i, \mathrm{arg}_i, \mathrm{pre}_i, \mathrm{eff}_i, \mathrm{term}_i, \mathrm{rec}_i), \tag{2}$$

where the entries specify the operation, typed arguments, preconditions, expected effects, termination conditions, and recovery strategy, respectively. Motor nodes invoke physical skills; cognitive nodes acquire evidence, verify state, bind variables, or update memory. An unresolved target remains symbolic until evidence supports its assignment in β_k. Dependencies encode both physical and informational prerequisites, linking the order of motor operations to the evidence required for target selection and subsequent manipulation.

For example, a container-exploration task initially specifies a target through its unobserved contents. Inspection nodes make those contents observable, and a binding operation assigns the target variable from the accumulated evidence. Later motor nodes refer to that variable even after the containers close. Recording epistemic status distinguishes missing evidence from an observed negative result. Motor nodes await resolved arguments, completed dependencies, and satisfied preconditions, allowing evidence acquisition to precede target-dependent actions.

3.3 Capability-Conditioned Executable-Space Projection

A task graph specifies semantic operations, but these operations may exceed the WAM's capabilities or violate physical prerequisites. We therefore restrict execution to a set $\mathcal{P}_W$ of validated WAM primitives, each specifying typed parameters, preconditions, effects, termination criteria, and recovery rules. Let $\mathcal{L}(\mathcal{P}_W)$ contain graphs whose motor nodes use these primitives, with cognitive nodes handled by the Task Manager. Let $\mathcal{F}(z_k)$ contain graphs consistent with scene state, bindings, dependencies, and robot configuration. For the proposed graph G_k^{vlm}, we obtain

$$G_k^{\mathrm{exec}} = \Pi_{\mathcal{L}(\mathcal{P}_W) \cap \mathcal{F}(z_k)}(G_k^{\mathrm{vlm}}), \tag{3}$$

where Π denotes deterministic constrained compilation. The compiler checks argument types, dependencies, preconditions and effects, single-arm holding state, and graph acyclicity. It normalizes operator names and safe entity aliases into the WAM instruction vocabulary and expands supported compositions. For example, `PLACE(object, target)` requires the robot to have acquired the same object. If a graph is infeasible, the compiler reports the violated constraints to the Task Manager for replanning. Projection thus constrains both instruction wording and physical consistency; the WAM generates continuous trajectories for the compiled skills.

New evidence can resolve variables or invalidate operations. Suffix revision modifies the unexecuted portion of the graph through node insertion, deletion, replacement, reordering, or dependency updates, subject to the constraint

$$G_{k+1}[V_k^{\text{executed}}] = G_k[V_k^{\text{executed}}], \tag{4}$$

where $G[V]$ denotes the operation definitions and dependencies of nodes in V, and V_k^{executed} contains executed nodes. The recorded prefix preserves attempted operations and their dependencies, while updated belief and runtime status describe the current scene and execution outcome. If a placement attempt drops an object, the revised suffix must include reacquisition to satisfy the holding-state precondition of the next placement attempt. Every revised suffix undergoes projection before execution. Appendix A.3 shows suffix revision after a placement failure.

The primitive vocabulary expands as additional WAM skills are validated; Appendix A describes its ontology and scope. Projection establishes consistency under the represented constraints. However, satisfying these constraints does not ensure physical success, so execution requires outcome verification before subsequent nodes can proceed.

3.4 Progress-Conditioned Event Control

To verify outcomes while maintaining continuous control, HarnessWAM uses a fast loop for local execution and progress estimation, with slower VLM deliberation triggered by execution events. In the fast loop, a prompt-conditioned estimator predicts continuous progress p_t, completion likelihood c_t, and a distribution π_t^{bin} over discrete progress levels:

$$(p_t, c_t, \pi_t^{\text{bin}}) = F_\phi(o_{t-L+1:t}, g_k), \tag{5}$$

where $L = 5$ is the dual-view RGB window length. A frozen SigLIP2 encoder (Tschannen et al., 2025) and a four-layer causal Transformer provide spatial and temporal features. The estimator receives neither robot state nor action commands; instruction conditioning relates observed visual changes to the currently active skill. Training combines progress regression, progress-bin classification, temporal ranking, endpoint anchoring, completion prediction, and local monotonicity. Completion labels derive from temporal position in demonstrations, so the estimator supplies cues for verification without certifying physical success.

Progress milestones and skill-budget exhaustion trigger the slow Task Manager. Using current RGB observations, progress history, and runtime state, it verifies the active operation's outcome and updates scene belief and variable bindings. It classifies the outcome as successful, failed, or uncertain and selects whether to continue, advance, observe, replan, recover, or terminate execution. Advancement requires semantic outcome verification and the node-specific gripper condition; successors become eligible once all prerequisites are satisfied.

The Task Manager jointly returns an outcome, decision, state updates, and revision indicator:

$$(\nu_k, d_k, \Delta B_k, \Delta\beta_k, \rho_k) = \mathcal{T}_{\text{VLM}}(x, o_{\tau_k}, B_k, G_k, M_k, r_k, \hat{s}_k^{\text{prog}}, e_k), \tag{6}$$

where ν_k is the outcome, d_k the decision, ΔB_k and $\Delta\beta_k$ the belief and binding updates, ρ_k the suffix-revision indicator, and $\hat{s}_k^{\text{prog}}$ the recent progress and completion trends.

An apparently complete motion can leave a required object relation unsatisfied. The Task Manager therefore checks the active node's expected effects against observations and retained evidence. Ambiguous imagery can trigger further observation before advancement or retry. These decisions update the belief, graph, and attempt history used for subsequent planning, connecting outcome verification to skill selection.

3.5 Embodiment-State Recovery and Task Termination

When outcome verification indicates failure or uncertainty, the Task Manager uses visual evidence, progress, attempt history, and remaining budget to select a continuation. It can extend a slowly progressing attempt, obtain observations, revise an invalid strategy, or initiate recovery. To support recovery, the runtime records the arm and gripper configuration (q_k^0, u_k^0) at the start of each motor node. Recovery applies multi-step joint control toward this configuration while retaining scene belief and interaction memory, then clears local WAM state. The Task Manager can revise the goal or future plan, which undergoes projection before execution resumes.

Objects may move during an unsuccessful attempt, so recovery does not restore an earlier world state. Subsequent decisions use updated observations together with retained evidence and the recorded outcomes of previous attempts.

The runtime declares completion when every required node is complete and final visual verification confirms the global goal. Execution terminates unsuccessfully when no feasible continuation remains or the task-level execution and recovery budget is exhausted. These limits bound physical interaction and recovery attempts. Appendix A gives implementation details and pseudocode.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Tasks and protocol. RoboMemArena contains 26 tasks in transfer, occlusion, counting, and sequential execution, with 4, 11, 7, and 4 tasks, respectively; 68.9% of subtasks depend on history (Lei et al., 2026). Trajectories average 1,076 environment steps. Each WAM's configurations use 20 rollouts per task with matched initial states, random seeds, observations, and task-level execution budgets. RoboCerebra Ideal evaluates compositional manipulation in static, fully observable scenes (Han et al., 2025), following its official protocol with 10 rollouts per task.

We evaluate two UR3 tasks, as shown in Figure 3. Tabletop Tidying assigns two cubic objects to a white box and two non-cubic objects to a black box, with four subtasks per trial. Box Exploration comprises six subtasks that inspect two boxes and place an object in the empty one, requiring retained evidence after the boxes close. We collect 100 demonstrations per task. All methods use these task-specific data with their respective checkpoints and are evaluated over 20 trials per task.

Metrics. For N episodes, let K_i count required subtasks in episode i and $s_{ik} \in \{0, 1\}$ indicate whether subtask k satisfies its completion predicate. Full-task success and mean subtask completion are defined as

$$\mathrm{SR}_{\mathrm{task}} = \frac{1}{N} \sum_{i=1}^{N} \prod_{k=1}^{K_i} s_{ik}, \qquad \mathrm{SR}_{\mathrm{sub}} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{K_i} \sum_{k=1}^{K_i} s_{ik}. \tag{7}$$

Evaluation subtasks follow benchmark predicates independently of graph nodes. RoboMemArena metrics average rollouts within each task, then weight tasks equally. RoboCerebra SR measures the mean completion rate of key object-state transitions.

Models and comparisons. We fine-tune LingBot-VA (Li et al., 2026) for each benchmark and additionally evaluate FastWAM (Yuan et al., 2026) on RoboMemArena. For each WAM, whole-task and subtask checkpoints use identical training data and optimization settings. Subtask-based variants share the subtask checkpoint. LingBot-VA receives 256×256 agent- and wrist-view RGB and robot state. The Task Manager uses Qwen3-VL-32B-Instruct (Bai et al., 2025) without task-specific fine-tuning or privileged visual inputs. Table 1 uses the results for $\pi_{0.5}$ (Physical Intelligence et al., 2025), HiF-VLA (Lin et al., 2026), MemoryVLA (Shi et al., 2026), MemER (Sridhar et al., 2026), and PrediMem from RoboMemArena (Lei et al., 2026), and includes the concurrent Harness VLA framework (Zhang et al., 2026). RoboCerebra comparisons include $\pi_{0.5}$, OpenVLA (Kim et al., 2024), GPT-4o Planner + OpenVLA, and HPE (Han et al., 2025). UR3 comparisons use $\pi_{0.5}$, FastWAM, and LingBot-VA. Appendix B details these evaluation protocols.

4.2 MAIN RESULTS

Simulation. On RoboMemArena, HarnessWAM achieves state-of-the-art performance using LingBot-VA, as shown in Table 1. The full-task and subtask SRs reach 64.4% and 76.8%, respectively, compared with 38.5% and 55.2% for PrediMem.

With LingBot-VA, HarnessWAM yields absolute gains of 15.4% in full-task SR and 16.8% in subtask SR over whole-task execution. The gains reach 26.0% and 27.6% with FastWAM. Combining subtask-trained control with HarnessWAM improves completion in every task family for both backbones.

On RoboCerebra Ideal, HarnessWAM achieves 23.70% SR, compared with 21.92% for GPT-4o Planner + OpenVLA and 21.10% for HPE, as shown in Table 2.

Table 1: RoboMemArena success rates (%). Standalone WAMs use whole-task checkpoints; Harness-WAM uses subtask checkpoints. Superscripts give absolute SR gains over the paired WAM, in the same units. Bold and underline denote the best and second-best results.

Method	Transfer		Occlusion		Counting		Sequence		Average	
	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}
$\pi_{0.5}$	20.0	42.8	12.7	17.2	14.3	50.9	60.0	71.6	21.5	38.7
HiF-VLA	17.5	38.9	12.7	27.1	8.6	45.9	42.5	70.2	16.9	39.8
MemoryVLA	15.0	37.2	7.3	13.1	14.3	55.1	37.5	65.2	15.0	35.3
MemER	20.0	36.1	16.4	33.2	27.1	65.1	65.0	79.1	27.3	49.1
PrediMem	<u>22.5</u>	<u>45.2</u>	27.3	38.4	45.7	69.3	72.5	<u>89.5</u>	38.5	55.2
Harness VLA	**50.0**	**58.3**	0.0	0.0	12.1	50.0	**82.5**	84.5	23.6	35.4
FastWAM	1.3	5.4	12.3	24.7	11.4	28.3	27.5	55.8	12.7	27.5
+ HarnessWAM	8.8	21.3	36.4	53.7	<u>55.7</u>	<u>73.0</u>	45.0	61.7	$38.7^{+26.0}$	$55.1^{+27.6}$
LingBot-VA	11.3	12.5	<u>63.2</u>	<u>66.6</u>	37.1	68.2	68.8	74.8	<u>49.0</u>	<u>60.0</u>
+ HarnessWAM	<u>22.5</u>	26.5	**76.8**	**87.7**	**59.3**	**80.6**	<u>81.3</u>	**90.6**	$\mathbf{64.4}^{+15.4}$	$\mathbf{76.8}^{+16.8}$

Table 2: Success rate on RoboCerebra Ideal (%).

Method	Ideal SR ↑
$\pi_{0.5}$	1.88
OpenVLA (fine-tuned)	7.84
GPT-4o Planner + OpenVLA	<u>21.92</u>
HPE Framework	21.10
HarnessWAM	**23.70**

Physical manipulation. In Table 3, HarnessWAM reaches 95.0% full-task SR and 96.3% subtask SR for tidying, compared with 30.0% and 55.0% for LingBot-VA. For exploration, the corresponding rates are 90.0% and 95.8%, compared with 5.0% and 25.8%. Neither $\pi_{0.5}$ nor FastWAM completes an exploration trial under the shared training data. HarnessWAM thus supports repeated rule-based manipulation and subsequent actions conditioned on earlier observations.

Table 3: Real-world success rates (%) with successful/total counts in parentheses. Each method is evaluated over 20 trials per task. Bold and underline denote the best and second-best results.

Method	Tabletop Tidying		Box Exploration	
	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}
$\pi_{0.5}$	5.0 (1/20)	31.3 (25/80)	0.0 (0/20)	0.0 (0/120)
FastWAM	25.0 (5/20)	40.0 (32/80)	0.0 (0/20)	2.5 (3/120)
LingBot-VA	<u>30.0 (6/20)</u>	<u>55.0 (44/80)</u>	<u>5.0 (1/20)</u>	<u>25.8 (31/120)</u>
HarnessWAM	**95.0 (19/20)**	**96.3 (77/80)**	**90.0 (18/20)**	**95.8 (115/120)**

4.3 Ablation Study

Architectural controls. Table 4 compares orchestration strategies sharing a subtask-trained LingBot-VA checkpoint. WAM + Static Plan fixes the initial skill sequence without memory updates, reaching 52.5% full-task SR. This sequence cannot respond to new evidence. Free-form orchestration permits online adaptation through prompts generated every 50 steps from current RGB and recent decisions. However, these updates lack structured task state and executable constraints, yielding

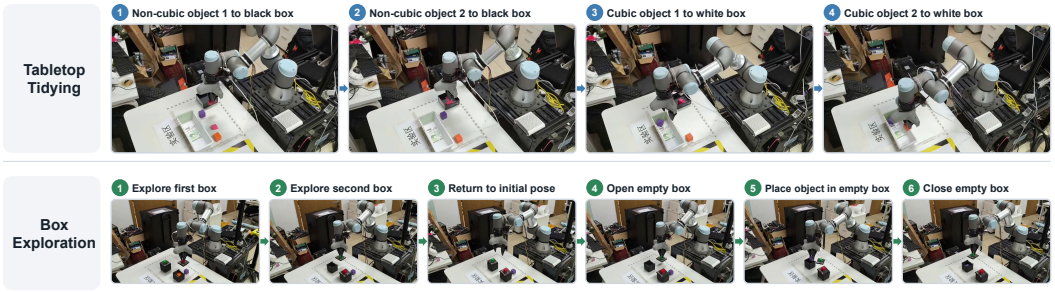

Figure 3: UR3 tasks. Tabletop Tidying sorts objects by shape. Box Exploration identifies an empty box through sequential inspection, then retains its identity for subsequent manipulation.

11.9% full-task SR. HarnessWAM couples state updates with constrained plan revision to incorporate new evidence into supported skill sequences, reaching 64.4% full-task SR.

Table 4: Architectural controls and ablations on RoboMemArena (%). All settings share the same subtask-trained LingBot-VA checkpoint. Bold and underline mark the best and second-best results.

Method	Transfer		Occlusion		Counting		Sequence		Average	
	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}	SR_{task}	SR_{sub}
(a) Architectural controls										
VLM + WAM (free-form)	0.0	3.9	23.2	44.3	7.9	28.8	0.0	16.9	11.9	29.7
WAM + Static Plan	15.0	16.7	60.0	73.6	<u>52.9</u>	**82.5**	68.8	89.6	52.5	69.7
(b) Planning and task state										
w/o task state	15.0	17.2	60.0	72.9	52.1	79.5	68.8	89.6	52.3	68.7
w/o executable projection	5.0	10.6	30.9	53.1	9.3	47.2	43.8	59.4	23.1	45.9
w/o suffix revision	20.0	23.5	72.7	**88.4**	50.7	77.9	62.5	76.0	57.1	73.7
(c) Execution control										
w/o progress events	13.8	17.5	59.1	66.9	27.1	63.3	55.0	67.3	42.9	58.4
Progress-only switching	**23.8**	<u>27.5</u>	<u>75.0</u>	85.5	51.4	77.4	67.5	**91.0**	<u>59.6</u>	<u>75.2</u>
w/o recovery	<u>22.5</u>	**28.5**	73.2	85.0	51.4	79.3	<u>70.0</u>	83.5	59.0	74.5
HarnessWAM	<u>22.5</u>	26.5	**76.8**	<u>87.7</u>	**59.3**	<u>80.6</u>	**81.3**	<u>90.6</u>	**64.4**	**76.8**

State and projection. Removing structured scene facts, history, and bindings reduces average full-task SR to 52.3%, with the largest decline on occlusion. Under occlusion, successful execution depends on evidence retained from earlier observations. Separately, disabling projection reduces full-task SR to 23.1% despite retaining task state and feedback. Access to task knowledge is therefore insufficient when proposed operations lack capability, argument, and dependency checks. Persistent evidence and executable constraints serve complementary roles in connecting task reasoning to WAM control under changing observations and physical conditions.

Online graph revision. We freeze future nodes and dependencies while reusing the full system's initial plans and retaining memory, binding, verification, and recovery. On occlusion tasks, subtask SR remains close to HarnessWAM at 88.4% versus 87.7%, while full-task SR falls from 76.8% to 72.7%. Memory and binding updates retain newly acquired evidence, while the fixed suffix leaves subsequent operations unchanged. Online revision adapts these operations and their dependencies to the updated task state, allowing new observations to guide the remaining execution.

Execution control. Progress events align Task Manager calls with skill execution; replacing them with periodic calls reduces full-task SR from 64.4% to 42.9%. At each intervention, the Task Manager checks the active skill's effects before authorizing advancement. Without semantic verification and

gripper-state gating, progress-only switching reaches 75.2% subtask SR and 59.6% full-task SR, as estimated completion may precede satisfaction of the next skill's prerequisites. When verification detects failure, recovery enables another attempt. Without recovery, failed or exhausted attempts terminate execution, lowering full-task SR to 59.0%.

Plan quality. Table 5 compares raw and projected VLM plans. Projection raises executability from 13.8% to 72.9%, dependency satisfaction from 58.1% to 92.9%, and binding accuracy from 21.3% to 88.3%. Syntactically valid plans can still select incorrect targets or violate skill prerequisites. Projection jointly checks bindings, dependencies, and physical constraints before WAM execution.

Table 5: Plan quality before and after executable-space projection (%).

Plan representation	Syntax ↑	Dependencies ↑	Binding ↑	Executability ↑
Raw VLM plan	60.8	58.1	21.3	13.8
Projected plan	**95.2**	**92.9**	**88.3**	**72.9**

4.4 Qualitative Results

Figure 4 shows a RoboMemArena Task 4 rollout. HarnessWAM opens and closes three drawers, retaining evidence about their contents. Once the drawers close, their contents are occluded, and the current observation alone no longer identifies the required destination. The Task Manager uses the retained evidence to bind the target variable to the non-empty top drawer and specify subsequent actions. This binding carries the inspection result into the placement instruction, allowing the WAM to execute a local skill with a resolved target. The robot then reopens this drawer and places the object inside. This trajectory instantiates the evidence acquisition and target binding illustrated in Figure 1, linking retained observations to subsequent manipulation.

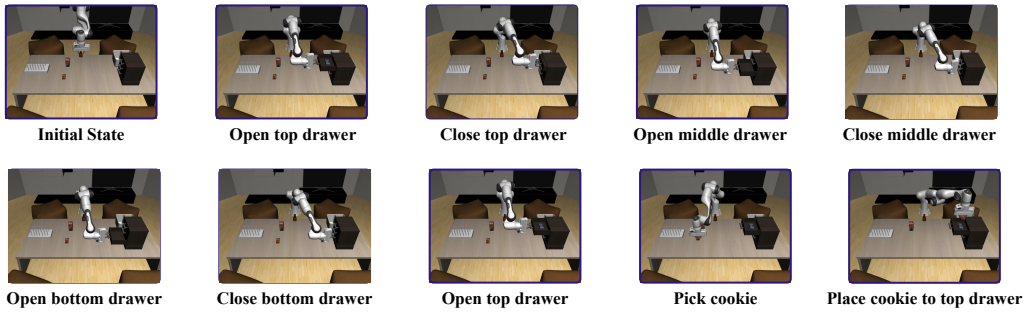

Figure 4: A RoboMemArena Task 4 rollout. Sequential drawer inspection supplies evidence for delayed target binding and subsequent object placement after the target contents become occluded.

5 Conclusion

HarnessWAM connects finite-horizon WAM control to persistent task-level deliberation. Evidence-grounded task graphs and capability-conditioned projection organize local skills, while event-triggered verification determines how observed outcomes change subsequent execution. Plan revision and recovery preserve the evidence and history needed to continue after failure. Results on RoboMemArena, RoboCerebra Ideal, and two UR3 tasks support this framework across history-dependent and compositional manipulation. Ablations identify contributions from planning, task state, verification, and recovery, while plan diagnostics show how constraint checks improve represented consistency. Performance remains limited by the WAM's skill repertoire and the accuracy of VLM state estimation. These limitations motivate broader skill coverage and calibrated uncertainty estimates to guide task-level deliberation under partial observability and uncertain action outcomes.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing to improve the grammar, clarity, and readability of the manuscript. All AI-assisted edits were reviewed by the authors to ensure that they preserved the intended scientific meaning. The authors take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

Section 3 specifies the task representation, executable-space projection, and execution control. Appendix A provides the representation schemas, progress-estimation objectives, training hyper-parameters, Task Manager outputs, and inference algorithm. Section 4.1 describes the models, demonstration data, whole-task and subtask checkpoint configurations, rollout counts, and evaluation metrics. Appendix B details the evaluation protocols, baseline configurations, architectural controls, component ablations, plan-quality metrics, and physical-task completion criteria.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Zi ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress. arXiv preprint arXiv:2410.04640, 2024. URL https://arxiv.org/abs/2410.04640.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. arXiv preprint arXiv:2204.01691, 2022. URL https://arxiv.org/abs/2204.01691.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report. arXiv preprint arXiv:2511.21631, 2025. URL https://arxiv.org/abs/2511.21631.

Ian M. Bullock, Joshua Z. Zheng, Sara De La Rosa, Charlotte Guertler, and Aaron M. Dollar. Grasp Frequency and Usage in Daily Household and Machine Shop Tasks. *IEEE Transactions on Haptics*, 6(3):296–308, 2013. doi: 10.1109/toh.2013.6. URL http://dx.doi.org/10.1109/TOH.2013.6.

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, Ru Ying, Ran Zheng, and Yao Mu. AHA-WAM:Asynchronous Horizon-Adaptive World-Action Modeling with Observation-Guided Context Routing. arXiv preprint arXiv:2606.09811, 2026. URL https://arxiv.org/abs/2606.09811.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. SARM: Stage-Aware Reward Modeling for Long Horizon Robot Manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 17091–17116, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1c92edb990a05f2269f0cc3afbb4c952-Paper-Conference.pdf.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning Universal Policies via Text-Guided Video Generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 9156–9172. Curran Associates, Inc., 2023. doi: 10.52202/075280-0403. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1d5b9233ad716a43be5c0d3023cb82d0-Paper-Conference.pdf.

Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-Supervised Visual Planning with Temporal Skip Connections. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 344–356. PMLR, 2017. URL https://proceedings.mlr.press/v78/frederik-ebert17a.html.

Frederik Ebert, Sudeep Dasari, Alex X. Lee, Sergey Levine, and Chelsea Finn. Robustness via Retrying: Closed-Loop Robotic Manipulation with Self-Supervised Learning. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 983–993. PMLR, 2018a. URL https://proceedings.mlr.press/v87/ebert18a.html.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control. arXiv preprint arXiv:1812.00568, 2018b. URL https://arxiv.org/abs/1812.00568.

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. URL http://dx.doi.org/10.1146/ANNUREV-CONTROL-091420-084139.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL http://dx.doi.org/10.1038/S41586-025-08744-2.

Songhao Han, Boxiang Qiu, Yue Liao, Siyuan Huang, Chen Gao, Shuicheng Yan, and Si Liu. RoboCerebra: A Large-scale Benchmark for Long-horizon Robotic Manipulation Evaluation. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-4031. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/aed3899aa23ae06868aeb2e2a856485d-Paper-Datasets_and_Benchmarks_Track.pdf.

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

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. arXiv preprint arXiv:2307.05973, 2023. URL https://arxiv.org/abs/2307.05973.

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

Huashuo Lei, Wenxuan Song, Huarui Zhang, Jieyuan Pei, Jiayi Chen, Haodong Yan, Han Zhao, Pengxiang Ding, Zhipeng Zhang, Lida Huang, Donglin Wang, Yan Wang, and Haoang Li. RoboMemArena: A Comprehensive and Challenging Robotic Memory Benchmark. arXiv preprint arXiv:2605.10921, 2026. URL https://arxiv.org/abs/2605.10921.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control. arXiv preprint arXiv:2601.21998, 2026. URL https://arxiv.org/abs/2601.21998.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng YAN, Maoqing Yao, and Guanghui Ren. Genie Envisioner: A Unified World Foundation Platform for Robotic Manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 88446–88463, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8e773e9e463e97b930c47621b6777159-Paper-Conference.pdf.

Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. HiF-VLA: Hindsight, Insight and Foresight through Motion Representation for Vision-Language-Action Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20732–20742, 2026.

Federico L. Moro, Nikos G. Tsagarakis, and Darwin G. Caldwell. On the Kinematic Motion Primitives (kMPs) - Theory and Application. *Frontiers in Neurorobotics*, 6:10, 2012. doi: 10.3389/fnbot.2012.00010. URL https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2012.00010.

J.D. Morrow and P.K. Khosla. Manipulation task primitives for composing robot skills. In *Proceedings of International Conference on Robotics and Automation*, volume 4 of *ROBOT-97*, pp. 3354–3359. IEEE, 1997. doi: 10.1109/robot.1997.606800. URL http://dx.doi.org/10.1109/ROBOT.1997.606800.

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

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning. arXiv preprint arXiv:2307.06135, 2023. URL https://arxiv.org/abs/2307.06135.

Marco Santello, Martha Flanders, and John F. Soechting. Postural Hand Synergies for Tool Use. *The Journal of Neuroscience*, 18(23):10105–10115, 1998. doi: 10.1523/jneurosci.18-23-10105.1998. URL http://dx.doi.org/10.1523/JNEUROSCI.18-23-10105.1998.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-Cognitive Memory in Vision-Language-Action Models for Robotic Manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 18567–18602, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1fa3d6ccbcd15f7285fee666b2bd57be-Paper-Conference.pdf.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up Memory for Robotic Control via Experience Retrieval. In *International Conference on Learning Representations*, volume 2026, pp. 97142–97166, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/9da515b1ad19d032a7398f00f5ff9b0c-Paper-Conference.pdf.

Xiaoquan Sun, Ruijian Zhang, Chen Cao, Yihan Sun, Jiahui Chen, Zetian Xu, Bo Chen, Haijier Chen, Zhen Yang, Jiarun Zhu, Yijun Hong, JingZhe Xu, Jingrui Pang, Mingqi Yuan, and Jiayu Chen.

HiMem-WAM: Hierarchical Memory-Gated World Action Models for Robotic Manipulation. arXiv preprint arXiv:2606.10363, 2026. URL https://arxiv.org/abs/2606.10363.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding, Localization, and Dense Features. arXiv preprint arXiv:2502.14786, 2025. URL https://arxiv.org/abs/2502.14786.

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, Jiabing Yang, Zhaowen Li, Yan Huang, Liang Wang, and Peng Su. DIM-WAM: World-Action Modeling with Diverse Historical Event Memory. arXiv preprint arXiv:2606.27677, 2026. URL https://arxiv.org/abs/2606.27677.

Youpeng Wen, Junfan Lin, Yi Zhu, Jianhua Han, Hang Xu, Shen Zhao, and Xiaodan Liang. VidMan: Exploiting Implicit Dynamics from Video Diffusion Model for Effective Robot Manipulation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 41051–41075. Curran Associates, Inc., 2024. doi: 10.52202/079017-1298. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/481c70828a4ff20d31a646cc6cc95f3d-Paper-Conference.pdf.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

Sizhe Yang, Juncheng Mu, Tianming Wei, Chenhao Lu, Xiaofan Li, Linning Xu, Zhengrong Xue, Zhecheng Yuan, Dahua Lin, Jiangmiao Pang, and Huazhe Xu. MemoryWAM: Efficient World Action Modeling with Persistent Memory. arXiv preprint arXiv:2606.20562, 2026. URL https://arxiv.org/abs/2606.20562.

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. URL https://arxiv.org/abs/2210.03629.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World Action Models are Zero-shot Policies. arXiv preprint arXiv:2602.15922, 2026. URL https://arxiv.org/abs/2602.15922.

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. URL https://arxiv.org/abs/2603.16666.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Zhengru Fang, Yi Nie, 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, 2026. URL https://arxiv.org/abs/2607.08448.

Ganlong Zhao, Zijia Tang, Xingping Chen, Zhanghui Kuang, Ye Tian, and Guanbin Li. FLARE: A Failure-Aware Framework for Autonomous Correction and Recovery in Visual-Language Robotic Manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22391–22401, 2026.