FROM VLM PLANNERS TO ROBOTIC AGENTS: SELF-EVOLVING HARNESSES FOR HISTORY-DEPENDENT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models achieve strong performance across embodied manipulation tasks by unifying visual understanding, language conditioning, and action generation. However, long-horizon tasks can involve non-Markovian dependencies: successful decisions may require retaining task progress and past interaction outcomes, while partial observability and execution errors compound over time. Existing work has begun to address these challenges by augmenting foundation-model policies with memory and by introducing hierarchical architectures that separate high-level planning from low-level execution. We organize the high-level VLM as a robotic agent through an editable harness to maintain task context and improve planning through execution experience. We instantiate this approach as SERA (Self-Evolving Robotic Agent), with a fixed VLA executor providing low-level control. The harness maintains and updates task context using observations and execution feedback to guide subsequent planning. Self-evolution uses environment rollouts to propose and evaluate harness revisions, with the selected configuration fixed before held-out testing. On RoboMME, SERA achieves 51.38% overall success with autonomous selection, compared with 44.38% without self-evolution. Our exploration suggests that agentic planning is a promising direction for history-dependent robotic manipulation.

1 INTRODUCTION

Vision-language-action (VLA) models connect visual observations and language instructions to robot actions, providing a foundation for general-purpose manipulation (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024). Long-horizon tasks, however, can require information that the current observation does not reveal (Koo et al., 2025; Dai et al., 2026). For example, when a robot must repeat an operation a specified number of times, similar images may call for different decisions depending on how many repetitions have actually succeeded. An earlier observation may also contain a cue that is no longer visible. Such history-dependent decisions require preserving task-relevant information and tracking execution outcomes over successive interactions.

Existing research provides several foundations for these capabilities. Memory-augmented policies equip VLA models with mechanisms for retaining and using historical information, including compact temporal representations (Koo et al., 2025; Jang et al., 2025; Zhu et al., 2026), memory retrieval, and recurrent state updates (Shi et al., 2025a; Hu et al., 2026; Li et al., 2026). Hierarchical architectures introduce a high-level planning layer that issues intermediate decisions to low-level executors (Ahn et al., 2022; Belkhale et al., 2024; Duan et al., 2025; Sridhar et al., 2025) and can incorporate environment feedback (Huang et al., 2022). Agentic systems further organize reasoning, actions, and feedback into iterative decision processes (Yao et al., 2022; Shinn et al., 2023; Wang et al., 2023). Recent embodied harnesses explore external task management (Zhang et al., 2026b; Gu et al., 2026) and experience-driven adaptation (Chen et al., 2026; Wang et al., 2026; Ding et al., 2026). Together, these developments motivate studying how the high-level planner in a hierarchical architecture can manage task context and make use of execution experience.

Our focus is on two connected responsibilities of this planning process. Within an episode, issuing a command does not establish that the intended operation has succeeded. In the repetition exam-

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.

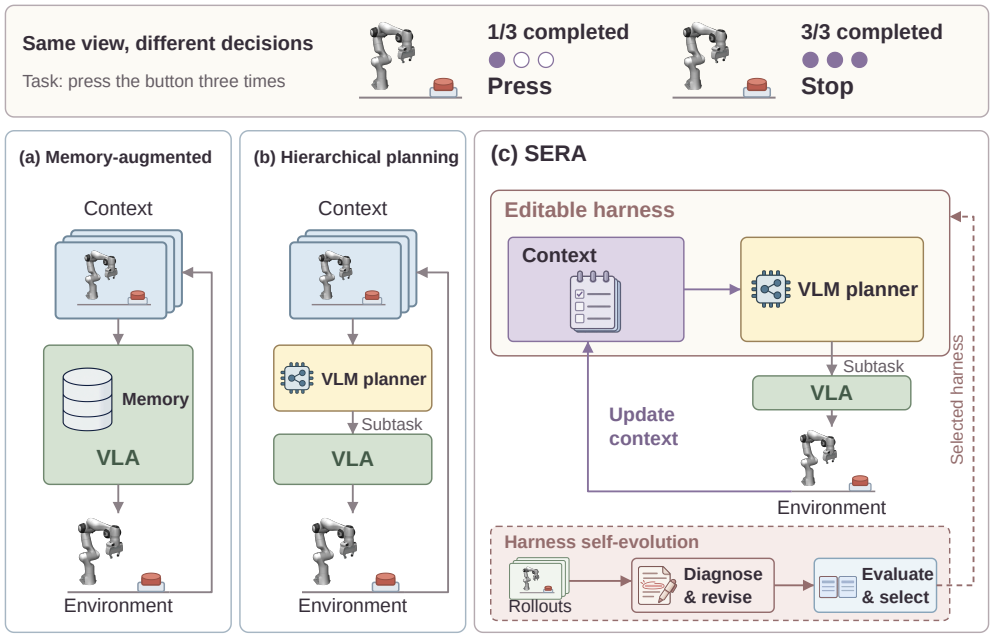

Figure 1: **From planner to agent for history-dependent manipulation.** Top: Similar observations can require different decisions depending on task progress. (a) Memory-augmented policies equip VLA models with memory mechanisms to retain and use historical information. (b) Hierarchical planning separates high-level planning from low-level execution. (c) SERA uses an editable harness to organize the VLM as an agent that maintains task context from observation and interaction history. Execution-driven self-evolution refines the harness using rollout feedback. The dashed arrow denotes the selected harness, fixed before testing.

ple, updating progress requires interpreting subsequent observations and feedback, while retaining information needed for later decisions. Access to history therefore needs to be accompanied by an explicit process for maintaining decision-relevant task context. Across attempts, an execution record must be translated into a reusable change to influence future planning. This raises a question: how can we support both ongoing context management and improvements to the planning process from execution experience? We explore an editable harness around the high-level VLM as a common mechanism for these responsibilities. The harness organizes context and feedback during execution, while providing an explicit object for revision during self-evolution. This also offers a way to investigate experience reuse when additional demonstrations and manual annotations are costly. Figure 1 illustrates this design direction.

We present SERA (Self-Evolving Robotic Agent), which organizes a VLM as an agent within an editable harness around a fixed VLA executor. During task execution, the harness supports sustained context management over long horizons, guiding the extraction, retention, and updating of task progress and execution outcomes from observations and feedback. This context supports the VLM's subsequent planning. However, a predefined harness may not anticipate all task requirements, motivating execution-driven self-evolution to identify and repair its limitations. Analysis and revision agents autonomously diagnose failures from execution traces and propose candidate repairs to planning guidance, assessment criteria, and control rules. A verifier evaluates these candidates through environment rollouts, providing feedback for further revision and harness selection. This process revises the harness without manually constructing additional demonstration datasets for supervised fine-tuning of the high-level VLM planner. The selected harness is fixed before held-out testing, while task context continues to be updated within each episode.

We evaluate SERA on RoboMME, a benchmark for history-dependent robotic manipulation (Dai et al., 2026). Autonomous SERA achieves 51.38% overall success, compared with 44.38% without self-evolution. These results support an empirical exploration of agentic planning and the aggregate effect of execution-driven harness revisions. Our contributions are:

- An agentic planning framework that organizes a high-level VLM within an editable harness for sustained task-context management over long horizons, tracking task progress and execution feedback to guide a fixed VLA executor.
- An execution-driven self-evolution procedure that autonomously proposes and evaluates revisions to planning guidance, assessment criteria, and control rules from rollout feedback, without constructing an SFT dataset.
- We evaluate SERA on RoboMME and achieve 51.38% overall success. Harness self-evolution improves success by 7.00 percentage points over the initial configuration.

2 Related Work

2.1 Memory-Augmented Policies

Generalist robot policies learn language-conditioned manipulation from diverse data (Brohan et al., 2022; Open X-Embodiment Collaboration, 2023; Octo Model Team et al., 2024). VLA models connect pretrained vision-language representations with action learning (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; NVIDIA et al., 2025; Physical Intelligence et al., 2025). Related embodied models, including PaLM-E, incorporate visual and continuous state inputs into language models for embodied reasoning and planning (Driess et al., 2023). These developments provide a foundation for language-conditioned manipulation, while history-dependent tasks additionally require information beyond the current observation. Recent work has also emphasized the importance of historical information in manipulation, with benchmarks such as RoboMME explicitly evaluating history-dependent behavior (Dai et al., 2026).

Memory-augmented policies incorporate such information through different representations and update mechanisms (Fang et al., 2025). HAMLET aggregates compact moment tokens across timesteps to condition action prediction (Koo et al., 2025). MemoryVLA retrieves perceptual and cognitive information from a memory bank and fuses it with current representations (Shi et al., 2025a), while ContextVLA compresses past observations into a context token for efficient multi-frame conditioning (Jang et al., 2025). Other designs address cross-subtask information transfer through event-triggered latent memory in WeaveLA (Zhu et al., 2026), hierarchical experience retrieval in ECHO (Hu et al., 2026), and temporal aggregation through frame- and chunk-level recurrent queries in ReMem-VLA (Li et al., 2026). These methods integrate historical information into action prediction through compact temporal representations, memory retrieval, or recurrent updates.

2.2 Hierarchical Robotic Planning

Hierarchical approaches connect high-level reasoning to executable robot behavior through intermediate decisions (Rana et al., 2023; Shi et al., 2025b; Zhang et al., 2026a). SayCan combines language-model planning with affordance estimates for pretrained skills (Ahn et al., 2022), and Inner Monologue incorporates environment feedback into subsequent planning (Huang et al., 2022). Code as Policies expresses robot behavior as programs invoking perception and control APIs (Liang et al., 2022). Other interfaces include the spatial value maps of VoxPoser, the relational keypoint constraints of ReKep, and the language motions of RT-H (Huang et al., 2023; 2024; Belkhale et al., 2024). RoboPARA further uses dependency graphs and resource constraints to organize parallel dual-arm execution (Duan et al., 2025).

Memory and hierarchical planning can also be combined. MemER trains a high-level VLM to nominate historical keyframes and predict language subtasks; selected keyframes inform subsequent decisions, while a low-level VLA executes the subtasks (Sridhar et al., 2025). Goal2Skill combines structured task memory with subtask verification and replanning in a VLM–VLA hierarchy (Liu et al., 2026).

2.3 Agentic Embodied Systems and Self-Evolution

Research on language models and agents studies reasoning, interaction, and feedback-driven refinement (Madaan et al., 2023). ReAct interleaves reasoning with actions and observations (Yao et al., 2022); Reflexion stores textual feedback for subsequent attempts (Shinn et al., 2023); and Voyager

accumulates reusable executable skills through interaction in Minecraft (Wang et al., 2023). Related methods improve prompts or agent code using execution feedback (Agrawal et al., 2025; Zhang et al., 2025). These works provide precedents for agent loops and experience reuse, although their evaluation settings differ from robotic manipulation.

Recent embodied systems place these ideas around learned executors (Ahn et al., 2024). Harness VLA combines a frozen VLA primitive with analytic primitives and task-specific and global memories (Zhang et al., 2026b). HarnessWAM maintains evidence-grounded task state and a task graph, and compiles plans into capability-constrained WAM skills (Gu et al., 2026). RoboMemory combines spatial, temporal, episodic, and semantic memories with a planner–critic loop (Lei et al., 2025). HELM couples episodic memory with a learned state verifier and recovery control around a frozen VLA (Zeng et al., 2026). Cortex combines task-context memory and skill constraints in a deployment harness (Peng et al., 2026).

Execution-driven harness adaptation is also an emerging research direction. ETA studies experience-derived strategy and skill updates subject to runtime checks before promotion (Chen et al., 2026). SHAPER evolves textual skills and a context-code harness using rollout feedback and validation while keeping its planner and executor frozen (Wang et al., 2026). Zetta revises runtime critics, recovery playbooks, and executable tools from failed rollouts (Ding et al., 2026). Within this line of work, SERA explores an editable harness for high-level planning in history-dependent robotic manipulation. The harness supports within-episode context management and is revised using development-stage execution experience, with the selected version fixed before testing.

3 METHOD

3.1 PROBLEM FORMULATION

We consider history-dependent robotic manipulation tasks specified by a language instruction i. Each episode may provide initial context c_0, such as a demonstration or previously observed events. At a policy query time t, the robot receives a sensory observation o_t, and the policy predicts an action chunk A_t. The decision problem is non-Markovian with respect to the instantaneous observation: similar current observations can require different actions depending on earlier interactions. For example, the correct placement target may depend on the order in which locations were visited in an earlier demonstration, which cannot be inferred from the current observation alone.

We define the interaction history h_t and predicted action chunk A_t as

$$h_t = (c_0, o_0, a_0, \ldots, a_{t-1}, o_t), \qquad A_t = (a_t, \ldots, a_{t+K-1}), \tag{1}$$

where K is the prediction horizon and $a_j \in \mathcal{A}$ is a control action. The objective is to complete the task within a bounded interaction budget using a history-conditioned policy,

$$A_t \sim \pi(\cdot \mid i, h_t). \tag{2}$$

SERA maintains an explicit task context alongside recent observations to make past information available for high-level planning. This context is an approximate record of task state and is updated as new execution evidence arrives.

3.2 AGENTIC HARNESS FOR PLANNING

Framework overview. SERA organizes a high-level VLM as an agent through an editable harness, with a fixed VLA executor for low-level control. We denote the harness configuration by H: it specifies how inputs are assembled, task context is maintained, commands are checked, and execution feedback is used in subsequent decisions. Figure 2(a) shows the resulting planning–execution loop. Execution-driven self-evolution revises H using rollout feedback, without requiring additional training of the VLM planner (Section 3.3).

Episode initialization and task context. At initialization, the harness derives episode guidance g, including a task description, from the instruction i and initial context c_0. This guidance remains fixed within the episode. During execution, the harness updates the task context c_n using recent observations and execution feedback. The context contains a compact summary of task progress,

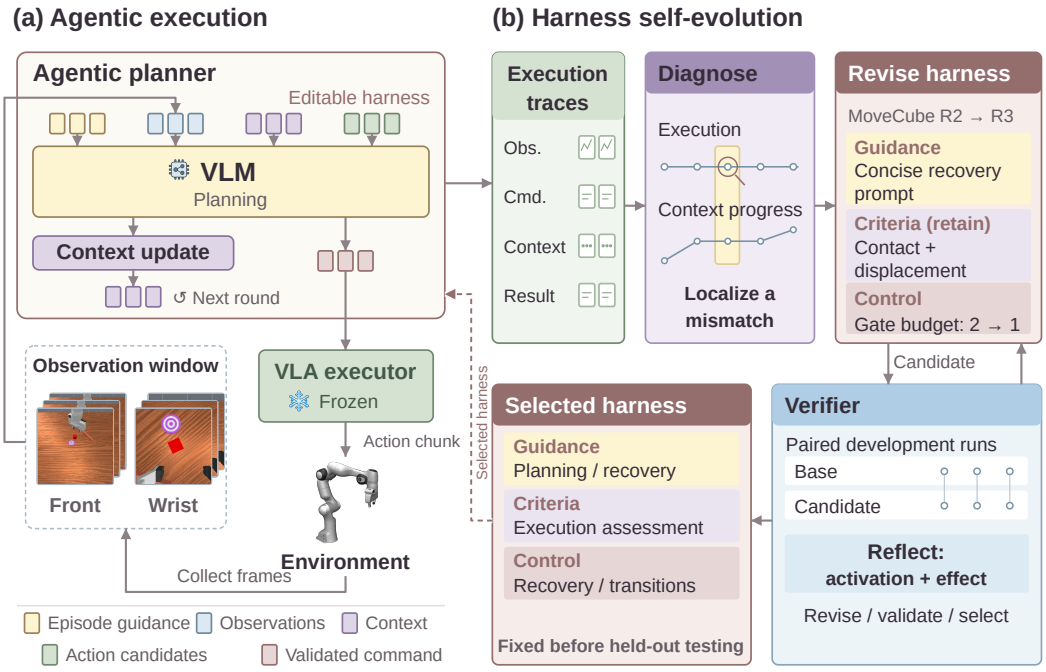

Figure 2: **Agentic execution and harness self-evolution.** **(a)** The harness maintains task context and uses a VLM to plan commands for a VLA executor, incorporating execution feedback in subsequent rounds. **(b)** Execution traces guide revisions to planning guidance, assessment criteria, and control rules. A verifier evaluates candidates and informs further revision and selection; the selected harness is fixed before testing. Tokens denote information categories and image stacks show observation sequences.

decision-relevant facts, execution outcomes, and unresolved uncertainty, together with control state for tracking lack of progress. By retaining relevant information after its supporting images leave the recent observation window, the context makes earlier events available to later planning. Initialization priors and context fields are detailed in Appendix B.2.

Context-aware planning. At planning round n, the harness provides the VLM with g, recent visual observations V_n, runtime context c_n, and the previously executed command u_{n-1}. To support command generation, it also supplies action candidates $A_{\text{cand},n}$ from the executor's supported command set. These candidates specify available commands and their required arguments. Candidate retrieval and command-format details are given in Appendix B.1.

The VLM uses the observations to assess the effect of the previous command, then proposes an updated context and a high-level command in the same call. Action-specific assessment criteria, when available, guide attention to relevant execution evidence. The harness checks the response and applies its execution-control rules before storing the resulting context and issuing the command. The combined update is

$$(c_{n+1}, u_n) = \mathcal{P}_H(g, V_n, c_n, u_{n-1}; A_{\text{cand},n}). \tag{3}$$

Here, $\mathcal{P}_H$ includes both the VLM call and the surrounding harness operations; u_n is the command actually passed to the executor. The first round uses an initial context and no preceding command.

Execution and feedback. The harness converts u_n into the executor's required input format. Let x_t denote the executor's native conditioning inputs, including its observations and any required task or history inputs. At the policy query time t_n for planning round n, the VLA with fixed parameters θ predicts an action chunk,

$$A_{t_n} \sim \pi_\theta(\cdot \mid x_{t_n}, u_n). \tag{4}$$

The execution loop applies a bounded prefix of this chunk over multiple control steps and collects the resulting observations. After this execution interval, the harness starts the next planning round

unless the episode terminates or exhausts its budget. Each round queries the VLA once; the same subtask can continue across rounds with newly predicted chunks. The next round combines this execution evidence with the retained context to decide whether to continue, advance, or recover a missing prerequisite. Harness control state tracks repeated uncertain or stagnant outcomes and can request reconsideration or another command selection. Context and control details appear in Appendix B.2; the runtime context is reset between episodes.

3.3 Execution-Driven Self-Evolution

A predefined harness may not anticipate all task requirements, leading to recurring errors in progress tracking, command selection, or recovery. Execution-driven self-evolution uses environment rollouts to identify and revise these limitations. Starting from H_0, analysis and revision agents propose candidate harness changes, and a verifier supplies feedback for further revision and selection (Figure 2(b)). This process changes the procedure that maintains and uses task context, beyond updating the context of an individual episode.

Failure diagnosis. An analysis agent examines recorded observations, task context, issued commands, and outcomes to identify where execution first diverged from the intended behavior. It considers physical execution and task-progress reasoning separately, tracing their relationship to the terminal failure. The diagnosis specifies a repair target, supporting evidence, uncertainty, and task scope, forming a hypothesis for subsequent rollout evaluation. Offline diagnostic materials and restrictions on deployable repair triggers are specified in Appendix B.3.

Candidate harness revisions. A revision agent translates the diagnosis into a candidate bundle with activation conditions and intended behavioral changes. Candidates can revise three parts of H: *planning guidance* directs the VLM's planning and recovery; *assessment criteria* specify evidence for judging execution outcomes; and *control rules* govern recovery and task transitions. Here, planning guidance refers to procedural instructions in the harness, while g denotes the episode guidance established at initialization. The harness incorporates candidate changes into prompt construction and runtime control, allowing the candidate to alter the execution loop in Section 3.2. The revision process operates on these external components without constructing an additional demonstration dataset for supervised fine-tuning of the high-level VLM planner.

Candidate revisions are expressed as structured updates to supported harness components. They can change the guidance supplied to the planner or activate registered runtime control mechanisms, while preserving the executor's command interface. This constrains the repair process to changes the execution loop can apply and inspect.

Verification and iterative revision. A verifier evaluates candidate harnesses through environment rollouts. For a given repair target, it compares candidate outcomes with cached baseline outcomes on the same evaluation episodes, recording repaired failures, regressed successes, and unchanged outcomes. A subsequent review examines whether the intervention activated and how it affected execution. These findings guide another revision, a different repair target, or rejection within the search budget. Candidates meeting the initial evaluation criteria can be assessed on additional episodes excluded from their diagnosis and revision. Panel construction and selection rules are described in Appendix B.3.

Selection and deployment. The autonomous selection procedure allows both candidates passing fresh validation and eligible fallbacks retained when the search budget is exhausted; tasks without a selected revision retain the base harness (Appendix B.3). The selected configuration $H^\star$ is fixed before held-out testing, and test outcomes do not enter its revision process. During testing, task context c_n continues to update, while $H^\star$ and executor parameters θ remain fixed.

4 Experiments

We evaluate SERA on RoboMME to examine agentic planning on history-dependent tasks and execution-driven harness repair, combining aggregate results with execution cases. We then discuss directions for improving execution reliability.

4.1 EXPERIMENTAL SETUP

Benchmark and metric. RoboMME (Dai et al., 2026) contains 16 history-dependent manipulation tasks in four suites: Counting, Permanence, Reference, and Imitation. We evaluate each SERA configuration on 50 episodes per task, totaling 800 episodes. Success rate (SR) is averaged equally across tasks within each suite and across all 16 tasks for the overall score. Appendix A gives the aggregation rule and per-task counts.

Baselines. We consider three categories of baselines in Table 1, following the evaluated configurations in RoboMME (Dai et al., 2026). **VLA baselines** directly predict actions from their inputs: $\pi_{0.5}$ (Physical Intelligence et al., 2025) uses the current observation without historical context, while $\pi_{0.5}$ w/ action history additionally includes past actions in its language input (Dai et al., 2026). **Memory-augmented policies** equip action models with mechanisms for retaining and using history. SAM2Act (Fang et al., 2025) uses a memory bank and memory attention to condition waypoint prediction on past observations; FrameSamp+Modul (Dai et al., 2026) samples historical frames and uses their visual features to modulate the action expert. **Hierarchical policies** separate high-level subgoal planning from low-level execution. SimpleSG+QwenVL generates language subgoals, whereas GroundSG+Gemini and GroundSG+QwenVL use grounded subgoals to specify execution targets (Dai et al., 2026). MemER (Sridhar et al., 2025) maintains selected historical keyframes as visual context for subgoal prediction. We evaluate **SERA**, which organizes a VLM planner through an editable harness for task-context management and execution-driven self-evolution. We use `gpt-5.6-sol` through Codex as the high-level VLM planner, with reasoning effort set to `high`.

4.2 AGENTIC PLANNING

How does agentic planning perform on manipulation tasks requiring historical information?

Table 1 shows that the complete SERA framework achieves 51.38% overall success, compared with 42.38% for MemER and 44.51% for FrameSamp+Modul. SERA obtains the highest scores among the listed methods on Reference and Permanence, which require using earlier observations to resolve task-relevant information beyond the current view. These results suggest the potential of organizing a high-level VLM as an agent for history-dependent manipulation.

Table 1: **RoboMME main results.** Task-averaged success rates (%; higher is better). Memory-augmented policies equip action models with mechanisms to retain and use history; hierarchical policies separate subgoal planning from low-level execution, including systems that also use memory. SERA is grouped with hierarchical policies. Baselines use official leaderboard results; SERA uses autonomous selection. **Bold** and underlined values denote the best and second-best scores across all listed methods.

Method	Counting	Permanence	Reference	Imitation	Average ↑
VLA baselines					
$\pi_{0.5}$	28.78	17.00	17.17	8.78	17.93
$\pi_{0.5}$ w/ action history	29.09	22.75	14.59	11.17	19.40
Memory-augmented policies					
SAM2Act	35.33	26.00	16.83	7.33	21.37
FrameSamp+Modul	**65.22**	25.11	36.33	**51.39**	44.51
Hierarchical policies					
SimpleSG+QwenVL	44.61	19.61	25.22	26.56	29.00
GroundSG+Gemini	12.75	15.75	11.25	6.50	11.56
GroundSG+QwenVL	38.00	39.34	31.56	21.89	32.70
MemER	48.83	53.17	38.00	29.50	42.38
SERA (Ours)	49.00	**57.00**	**69.50**	30.00	**51.38**

Within the Reference suite, VideoPlaceOrder illustrates how demonstration understanding and sustained context management support execution (Figure 3b), even without self-evolution. In this

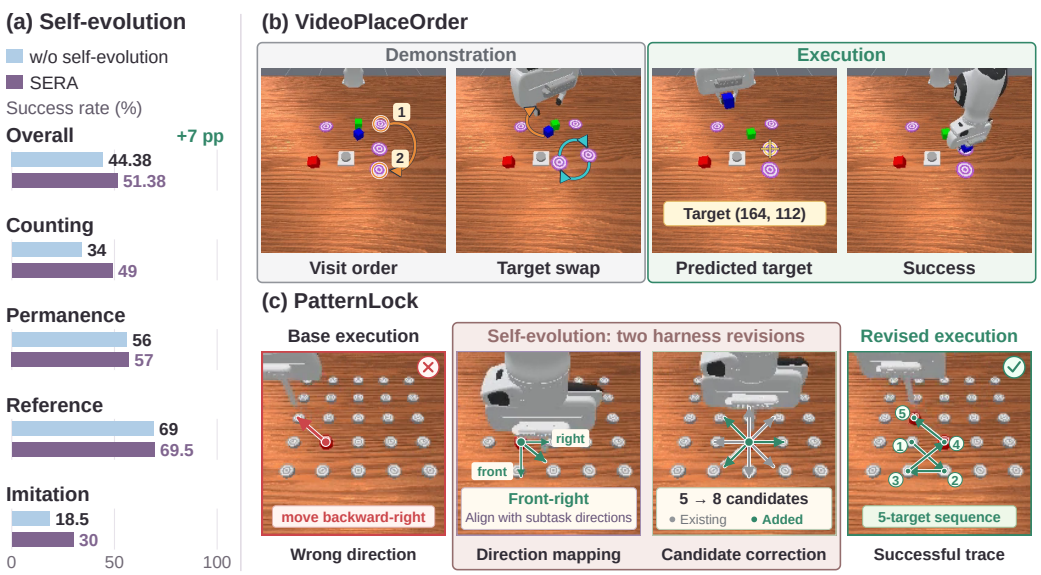

Figure 3: **Self-evolution results and complementary execution cases.** (a) RoboMME success rates before and after harness revision. (b) VideoPlaceOrder: the base agent analyzes a demonstration to identify the intended target despite a swap, then uses retained task context to predict its current location and successfully place the cube. (c) PatternLock: self-evolution diagnoses direction-mapping errors and missing action candidates, revises the direction guidance and candidate set, and the revised agent successfully completes the sequence. VideoPlaceOrder shows a test episode; PatternLock shows a self-evolution repair example. Direction guides and candidate colors are schematic; paths show motion and target order.

episode, the instruction specifies the second target visited in the demonstration. The robot must identify that historical target and place the blue cube at its current location. The agent identifies the intended target by interpreting the demonstration's visit order and subsequent target swap. The harness retains this information across grasp attempts while task context is updated with execution feedback. Once grasping is confirmed, the agent combines the retained target information with the current observation to locate the destination and complete placement. The execution trace separates two kinds of information: the destination inferred from the demonstration remains available through initialization guidance, while task context records whether grasping has succeeded. Subsequent planning therefore combines a persistent task goal with changing execution state. The full demonstration is not supplied again at each planning round in this run. We next examine whether execution experience can improve planning guidance and control rules.

4.3 Execution-Driven Self-Evolution

Can execution-driven self-evolution repair limitations in the initial harness and improve task performance?

An initial harness may contain incomplete planning guidance, imperfect assessment criteria, or unsuitable control rules. Feedback obtained from execution provides an opportunity to identify where these limitations affect task performance and to revise the corresponding parts of the harness. With execution-driven self-evolution, SERA improves overall success from 44.38% to 51.38%, a gain of 7.00 percentage points (Figure 3a). Revisions address inconsistent spatial interpretation, insufficient checks before advancing task stages, and missing action candidates, improving performance on tasks such as SwingXtimes and PatternLock.

Figure 3c illustrates how these issues are identified and addressed through harness self-evolution on PatternLock. Analysis of failed execution reveals an incorrect mapping between visual directions and executor commands, together with a candidate set that omits a required action. Self-evolution

corrects the direction guidance and candidate restriction, enabling the agent to complete the demonstrated sequence using existing executor actions.

The revision history also shows how verifier feedback guides subsequent changes. On the same six-episode revision panel, the initial harness succeeds in two episodes. A first revision targeting direction interpretation succeeds in three, but inspection of a remaining failure reveals that a required direction is absent from the retrieved candidates. A second revision addresses this remaining issue by making all eight existing directional commands available to the planner. With this revised candidate set and the direction guidance, the harness succeeds in five of the six episodes. These are pre-test revision results, illustrating how an incomplete repair can be refined through rollout feedback; they do not constitute an isolated comparison of the two repair components.

4.4 Further Improvements and Discussion

What further improvements could help our agentic planner translate high-level decisions into more reliable task execution?

One direction is to extend the harness with mechanisms that can intervene more directly in execution when replanning is insufficient. Exploratory RoboMemArena (Lei et al., 2026) evaluations motivate this direction. In a task requiring two pours followed by placing the container in a rack, the planner requests the second pour, but the executor places the container in the rack. Although task context retains the unfinished requirement, subsequent regrasp attempts fail. This observation suggests that maintaining the intended task state could be complemented by mechanisms for correcting execution deviations.

Exploratory gripper interventions provide an initial indication of this possibility. Some development runs complete the task with the intervention when matched runs without it fail, while another remains unsuccessful. These interventions operate beyond the scope of our current planner-side harness. The mixed outcomes observed across these runs indicate that achieving reliable recovery remains an open problem for this setting. Future work could investigate how an agentic planner can select and assess execution-level interventions, connecting its task understanding with more effective recovery.

5 Conclusion

We presented SERA, a self-evolving robotic agent for history-dependent robotic manipulation that organizes a high-level VLM within an editable harness around a fixed VLA executor. The harness manages task context during execution, while self-evolution uses development-time execution experience to revise planning guidance, assessment criteria, and control rules. The selected harness is fixed before held-out testing. On RoboMME, autonomous SERA achieves 51.38% success, compared with 44.38% without self-evolution. These results support further investigation into how the organization and adaptation of the high-level planner can contribute to VLA-based robotic control over long execution horizons.

Limitations and future work. VLM reasoning introduces planning latency into the execution loop. Repeated inference over observations and task context can accumulate substantial overhead during long-horizon execution, potentially limiting timely responses in dynamic tasks. The resulting trade-off between reasoning quality and execution responsiveness remains to be systematically evaluated. Future work will investigate more efficient context processing and reasoning strategies to reduce planning latency while preserving decision quality.

AI use statement

We used large language models as assistive tools to help collect related literature, assist with code writing and experiment organization, and support manuscript drafting and language editing through discussions with the authors. The authors carefully reviewed and checked AI-assisted work at each stage. The core research ideas were developed by the authors, who made the final decisions and take responsibility for the final content of the paper.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning, 2025. URL https://arxiv.org/abs/2507.19457v2.

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, 2022. URL https://arxiv.org/abs/2204.01691v2.

Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Sean Kirmani, Isabel Leal, Edward Lee, Sergey Levine, Yao Lu, Sharath Maddineni, Kanishka Rao, Dorsa Sadigh, Pannag Sanketi, Pierre Sermanet, Quan Vuong, Stefan Welker, Fei Xia, Ted Xiao, Peng Xu, Steve Xu, and Zhuo Xu. AutoRT: Embodied Foundation Models for Large Scale Orchestration of Robotic Agents, 2024. URL https://arxiv.org/abs/2401.12963v1.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quan Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action Hierarchies Using Language, 2024. URL https://arxiv.org/abs/2403.01823v2.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control, 2024. URL https://arxiv.org/abs/2410.24164v4.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale, 2022. URL https://arxiv.org/abs/2212.06817v1.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, 2023. URL https://arxiv.org/abs/2307.15818v1.

Yitong Chen, Zezheng Huai, Sixian Li, Yubang Wang, Haozhe Zhang, Yifei Zhang, Hechang Chen, Jingjing Gong, Yu-Gang Jiang, and Xipeng Qiu. ETA: A New Agentic Paradigm for Embodied Tasks, 2026. URL https://arxiv.org/abs/2608.03924v1.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and Understanding Memory for Robotic Generalist Policies, 2026. URL https://arxiv.org/abs/2603.04639v3.

Xin Ding, Liang Mi, Mingzhe Huang, Zixuan Wang, Chao Zhang, Zixu Hao, Fu Chen, Xiangyu Li, Yikai Zheng, Yaoyu Guo, Weijun Wang, Kun Li, Hao Wu, Yunxin Liu, and Ting Cao. Zetta ζ: An Efficient Closed-Loop Embodied Harness for Self-Evolving Physical Intelligence, 2026. URL https://arxiv.org/abs/2608.16590v1.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An Embodied Multimodal Language Model, 2023. URL https://arxiv.org/abs/2303.03378v1.

Shiying Duan, Pei Ren, Nanxiang Jiang, Zhengping Che, Jian Tang, Zhaoxin Fan, Yifan Sun, and Wenjun Wu. RoboPARA: Dual-Arm Robot Planning with Parallel Allocation and Recomposition Across Tasks, 2025. URL https://arxiv.org/abs/2506.06683v4.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. SAM2Act: Integrating Visual Foundation Model with A Memory Architecture for Robotic Manipulation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 15925–15942. PMLR, 2025. URL https://proceedings.mlr.press/v267/fang25c.html.

Zhaopeng Gu, Bingke Zhu, Tianxi Lin, Guibo Zhu, Yingying Chen, Kai Wang, Tingyu Yuan, Chaoyang Zhao, Zhaowen Li, Peng Su, and Jinqiao Wang. HarnessWAM: Bridging Prediction and Deliberation in World Action Models, 2026. URL https://arxiv.org/abs/2608.09516v1.

Yanbin Hu, Jin Cui, Jiayi Lu, Ruixuan Yang, Jun Ye, Boran Zhao, Xingyu Chen, Xuguang Lan, and Pengju Ren. ECHO: Continuous Hierarchical Memory for Vision-Language-Action Models, 2026. URL https://arxiv.org/abs/2605.10993v1.

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, 2022. URL https://arxiv.org/abs/2207.05608v1.

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, 2023. URL https://arxiv.org/abs/2307.05973v2.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation, 2024. URL https://arxiv.org/abs/2409.01652v2.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. ContextVLA: Vision-Language-Action Model with Amortized Multi-Frame Context, 2025. URL https://arxiv.org/abs/2510.04246v1.

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, 2024. URL https://arxiv.org/abs/2406.09246v3.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch your Vision-Language-Action Model into a History-Aware Policy, 2025. URL https://arxiv.org/abs/2510.00695v3.

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, 2026. URL https://arxiv.org/abs/2605.10921v1.

Mingcong Lei, Honghao Cai, Yuyuan Yang, Yimou Wu, Jinke Ren, Zezhou Cui, Liangchen Tan, Junkun Hong, Gehan Hu, Shuangyu Zhu, Shaohan Jiang, Ge Wang, Junyuan Tan, Zhenglin Wan, Zheng Li, Zhen Li, Shuguang Cui, Yiming Zhao, and Yatong Han. RoboMemory: A Brain-inspired Multi-memory Agentic Framework for Interactive Environmental Learning in Physical Embodied Systems, 2025. URL https://arxiv.org/abs/2508.01415v7.

Hang Li, Fengyi Shen, Dong Chen, Liudi Yang, Xudong Wang, Jinkui Shi, Zhenshan Bing, Ziyuan Liu, and Alois Knoll. ReMem-VLA: Empowering Vision-Language-Action Model with Memory via Dual-Level Recurrent Queries, 2026. URL https://arxiv.org/abs/2603.12942v1.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control, 2022. URL https://arxiv.org/abs/2209.07753v4.

Zhen Liu, Xinyu Ning, Zhe Hu, Xinxin Xie, Weize Li, Zhipeng Tang, Chongyu Wang, Zejun Yang, Hanlin Wang, Yitong Liu, and Zhongzhu Pu. Goal2Skill: Long-Horizon Manipulation with Adaptive Planning and Reflection, 2026. URL https://arxiv.org/abs/2604.13942v1.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback, 2023. URL https://arxiv.org/abs/2303.17651v2.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots, 2025. URL https://arxiv.org/abs/2503.14734v1.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy, 2024. URL https://arxiv.org/abs/2405.12213v1.

Open X-Embodiment Collaboration. Open X-Embodiment: Robotic Learning Datasets and RT-X Models, 2023. URL https://arxiv.org/abs/2310.08864v1.

Jiaqi Peng, Xiqian Yu, Delin Feng, Yuqiang Yang, Wenzhe Cai, Jing Xiong, Ganlin Yang, Jinliang Zheng, Jiafei Cao, Xueyuan Wei, Jiangmiao Pang, Yuan Shen, and Tai Wang. Cortex: A Bidirectionally Aligned Embodied Agent Framework for Long-horizon Manipulation, 2026. URL https://arxiv.org/abs/2607.05377v1.

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, 2025. URL https://arxiv.org/abs/2504.16054v1.

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 Task Planning, 2023. URL https://arxiv.org/abs/2307.06135v1.

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, 2025a. URL https://arxiv.org/abs/2508.19236v2.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi Robot: Open-Ended Instruction Following with Hierarchical Vision-Language-Action Models, 2025b. URL https://arxiv.org/abs/2502.19417v1.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366v4.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling Up Memory for Robot Control via Experience Retrieval, 2025. URL https://arxiv.org/abs/2510.20328v1.

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, 2023. URL https://arxiv.org/abs/2305.16291v2.

Peidong Wang, Zhiming Ma, Ying Chang, Xufang Luo, Xiaocui Yang, Shi Feng, Yuqing Yang, and Dongsheng Li. Self-Evolving Embodied Agents via Skill-Harness Evolution, 2026. URL https://arxiv.org/abs/2608.11350v1.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2022. URL https://arxiv.org/abs/2210.03629v3.

Zijian Zeng, Fei Ding, Huiming Yang, and Xianwei Li. HELM: Harness-Enhanced Long-horizon Memory for Vision-Language-Action Manipulation, 2026. URL https://arxiv.org/abs/2604.18791v1.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents, 2025. URL https://arxiv.org/abs/2505.22954v1.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, Jiaming Zhou, Yishu Shen, Yudong Jin, Fangyi Xu, Shuailei Ma, Jiaqi Liao, Guanxing Lu, Zifan Shi, Yongkun Wen, Yujie Zhao, Weixuan Tang, Xinyang Wang, Chaojian Li, Jiapeng Zhu, Ka Leong Cheng, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Native Video-Action Pretraining for Generalizable Robot Control, 2026a. URL https://arxiv.org/abs/2607.08639v2.

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, 2026b. URL https://arxiv.org/abs/2607.08448v4.

Shoujing Zhu, Zhenyang Liu, Fungmiu Wang, Jiafeng Wang, Bo Yue, Guiliang Liu, Simo Wu, Xiangyang Xue, and Taiping Zeng. WeaveLA: Event Driven Cross-Subtask Latent Memory Weaving for Repetitive Robot Manipulation, 2026. URL https://arxiv.org/abs/2606.17463v1.

A Additional Evaluation Details

A.1 Metric Aggregation and Evaluation Scope

Let $N = 16$ be the number of tasks and $n_i = 50$ the number of evaluation episodes for task i. Let $b_{ij} \in \{0, 1\}$ indicate full task success for episode j of task i. We report

$$\mathrm{SR} = \frac{100}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} b_{ij}. \tag{5}$$

Suite-level scores use the same rule restricted to the four tasks in that suite. Since each task has the same number of episodes, task averaging and episode pooling coincide. Scores are computed before rounding for display.

On RoboMME, autonomous SERA selects intervention bundles before test evaluation. The selection includes candidates passing development and fresh-validation checks, as well as terminal development fallbacks allowed by the procedure in Section 3.3. Tasks without a selected intervention retain the base result. Selected candidates remain in the aggregate even when their test score decreases; they are not replaced after observing test outcomes. The comparison therefore reflects the selection procedure as applied, rather than a per-task maximum over base and adapted test results.

A.2 RoboMME Per-Task Results

Table 2 compares our local reproduction of MemER (Sridhar et al., 2025) with the final SERA configuration. Each method is evaluated on 50 episodes per task. The MemER counts come from a single local evaluation of the released checkpoints; Table 1 uses the public leaderboard results.

Table 2: **Per-task RoboMME results.** Successful episodes out of 50 per task. MemER is our local reproduction using released checkpoints, distinct from the public leaderboard results in Table 1. SERA uses autonomous harness selection. Bold denotes the higher count in each row, including ties.

Suite	Task	MemER (reproduced)	SERA
Counting	BinFill	**30/50**	24/50
Counting	PickXtimes	**41/50**	35/50
Counting	SwingXtimes	**36/50**	**36/50**
Counting	StopCube	2/50	**3/50**
Permanence	VideoUnmask	43/50	**48/50**
Permanence	VideoUnmaskSwap	18/50	**46/50**
Permanence	ButtonUnmask	**41/50**	20/50
Permanence	ButtonUnmaskSwap	**12/50**	0/50
Reference	PickHighlight	**35/50**	27/50
Reference	VideoRepick	12/50	**17/50**
Reference	VideoPlaceButton	14/50	**49/50**
Reference	VideoPlaceOrder	18/50	**46/50**
Imitation	MoveCube	**45/50**	30/50
Imitation	InsertPeg	**3/50**	0/50
Imitation	PatternLock	10/50	**30/50**
Imitation	RouteStick	**3/50**	0/50
Overall		363/800	**411/800**

B HARNESS IMPLEMENTATION DETAILS

B.1 ROBOMME COMMAND INTERFACE

The command catalog is extracted from released executor training labels. Each entry specifies a skill identifier, an exact language template, its grounding-point count, and recorded task and transition statistics. The base retrieval scheme first restricts candidates to the known task's entries when available. It combines rankings for the preceding skill, recorded successors, and skill frequency using reciprocal-rank fusion, supplying the top five candidates or all candidates if fewer are available. This scheme does not use semantic query similarity, even though the retrieval interface accepts text.

The VLM selects a catalog entry and copies its template. Each required grounding point includes an object description, a source front-camera frame, and integer coordinates normalized to $[0, 999]$ along each image axis. The harness checks the template, point count, coordinate bounds, and frame validity before rendering the command. For example, the catalog contains `pick up the peg at <point_1>`; a subsequent hooking template requires points for the cube and its destination. These are catalog examples, not measured trajectory outcomes. Zero-point templates require no grounding points.

The VLA retains its native task-goal and observation interface. In the RoboMME driver, this includes current front/wrist images, proprioceptive state, and the executor's image/state buffer. This buffer is distinct from the recent visual window supplied to the high-level planner. The command interface does not establish physical success; execution evidence is processed in the next planning round.

B.2 CONTEXT STATE AND RUNTIME CONTROL

For RoboMME, episode initialization processes the instruction and initial context using benchmark-provided task-specification priors. Synchronized front- and wrist-camera observations supply scene-level and local interaction evidence during execution.

The RoboMME base planner emits a rewritten progress summary bounded to 1,600 characters. Its prompt requests the current phase, completed steps, task state, physical state, execution state, and material uncertainty. Structured execution fields also describe the held object, object support, and gripper state. These values are model estimates. The description and guidance obtained at episode initialization remain fixed, while the runtime summary and control state are updated and reset between episodes.

Execution status distinguishes progressing, completed, uncertain, stagnant, and regressed outcomes, with a separate initial status. Harness-maintained counters track repeated lack of progress and stagnation. In the catalog-only base configuration, they select normal planning, reconsideration of the current state, or renewed selection among known commands. The counters are not counts of verified task-completion events. The action-specific visual check is advisory within the same VLM call; it has no independent Python-side veto over the visual judgment. Subsequent runtime control determines the effective command and context stored for the next round.

B.3 DEVELOPMENT SELECTION AND REVISION SCOPE

Offline failure analysis may use benchmark-internal traces or expert demonstrations in addition to recorded planner and execution data. These are diagnostic materials: deployed revisions must express their activation conditions using task identity and runtime-observable information.

The autonomous selection protocol permits both candidates passing development and fresh-validation checks and terminal development fallbacks. In the reported protocol, a fallback with one additional successful episode and no observed regression can be retained for selection after search-budget exhaustion. A task with no selected intervention retains its base result. The deployed version records the selection route; the aggregate is not restricted to candidates passing fresh validation. A panel containing only baseline failures cannot establish retention of previously successful episodes.

Candidate manifests describe task scope, activation conditions, components, and revision lineage. The checked runtime loads prompt guidance, criterion overlays, and control rules into their corre-

sponding harness operations. The manifest format also defines broader extension types, but their presence in the schema does not establish that they were executed: in particular, the checked runtime rejects unregistered Python intervention plugins. We do not treat this interface as evidence of unrestricted online code modification.