VISTA: ACTIVE EXPLORATION FOR SELF-IMPROVEMENT IN EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic embodied intelligence plays a foundational role in open-ended, dynamic physical environments. Rather than treating policy learning as an end-to-end mapping from instructions to actions, recent studies reveal that embodied systems depend not only on pre-trained foundation models, but also on harness. Prior approaches largely treat rollouts as static logs for post-hoc reflection, producing harness updates without autonomously testing and refining them through subsequent embodied interaction. To address this challenge, we propose VISTA, a meta-agent framework that transforms post-hoc rollout reflection into active self-exploration for embodied harness improvement. Specifically, VISTA maintains a harness DAG that records the lineage of proposed harnesses, which enables the meta-agent to retrieve prior hypotheses, build on effective mechanisms, and avoid repeatedly exploring failed directions. At each iteration, VISTA first evaluates the current harness to obtain rollout evidence and initiaze a candidate harness. Rather than directly committing an update from these historical trajectories, the meta-agent actively explores in the environment. It autonomously generates new rollouts, probes execution failures and recovery behaviors, and iteratively modifies the harness based on the resulting evidence. The refined harness is then evaluated afresh and added to the harness DAG. We evaluate VISTA on the LIBERO-Pro and RoboCasa benchmarks. Experimental results demonstrate that VISTA progressively improves task success and is more rollout-efficient than static reflection, achieving higher success rates with fewer simulator rollouts.

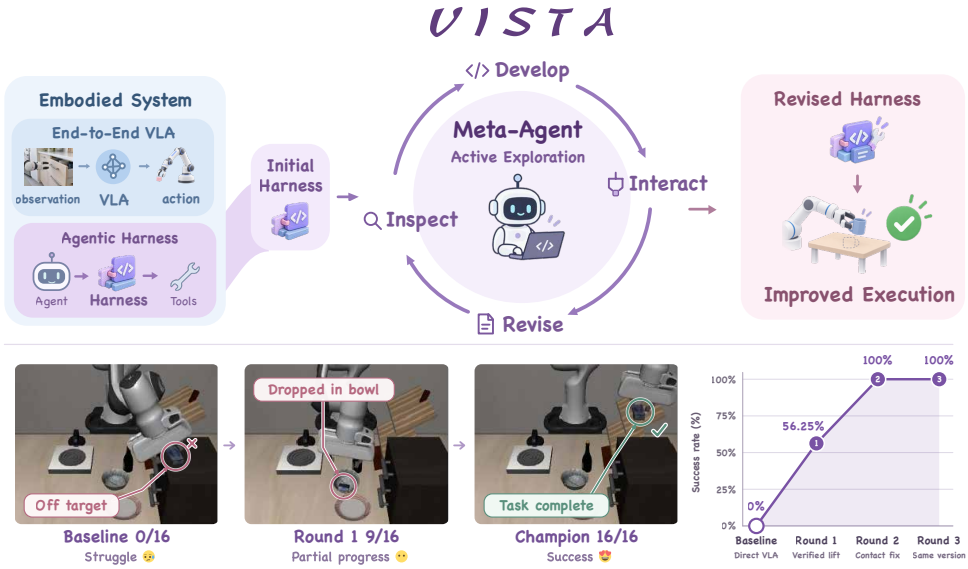


Figure 1: **VISTA actively explores the environment to improve embodied-agent harnesses.** Starting from an initial harness, a meta-agent inspects prior execution evidence, proposes revisions, and actively interacts with the environment to test hypotheses and refine candidate harnesses before formal evaluation.

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.

1 INTRODUCTION

Embodied artificial intelligence has emerged as a cornerstone for enabling autonomous systems to perceive, reason, and interact seamlessly within dynamic physical environments, offering transformative potential across household assistance, healthcare logistics, and industrial automation (Brohan et al., 2022; Collaboration et al., 2023). Catalyzed by multimodal foundation models and large-scale embodiment datasets, learning-based paradigms have demonstrated increasingly competitive capabilities in executing atomic manipulation tasks (Zitkovich et al., 2023; Octo Model Team et al., 2024; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025; Wu et al., 2023; Lyu et al., 2026; Ye et al., 2026; Li et al., 2026a). But scaling end-to-end policies to long-horizon, multi-stage tasks remains notoriously difficult. End-to-end models suffer severely from compounding execution errors over extended trajectories, out-of-distribution generalization under perturbations, and prohibitive sample complexity when attempting to fit multi-modal, long-tail action distributions across heterogeneous task phases (Ahn et al., 2022; Huang et al., 2022a).

To overcome these fundamental limitations, the convergence of frontier large language model (LLM) agents and robotics has established embodied agentic systems as a pivotal research frontier (Liang et al., 2023; Singh et al., 2022; Huang et al., 2023; Wang et al., 2023; Zhang et al., 2026). By coupling high-level symbolic reasoning and stateful tool orchestration with low-level control policies, agentic embodied architectures offer decisive advantages in managing long-horizon complexity: they facilitate modular hierarchical task decomposition, support active perception to resolve visual ambiguities, and enable closed-loop error diagnosis and autonomous recovery. The performance, reliability, and long-horizon coherence of such systems depend not only on their pre-trained foundation model weights, but also fundamentally on their harness (Yang et al., 2024; Shinn et al., 2023). To construct such harnesses, existing approaches predominantly rely on handcrafted heuristic rules (Liang et al., 2023; Singh et al., 2022) or static memory augmentation (Wang et al., 2023; Zhang et al., 2026), where execution workflows and fallback mechanisms are either hardcoded or rigidly replayed from reference demonstrations. However, these designs are fundamentally passive and brittle. Most critically, they lack the capacity for causal diagnosis and autonomous evolution.

To address this issue, we propose **VISTA**, a meta-agent framework that unifies active environmental exploration with closed-loop harness adaptation to achieve embodied self-improvement. Unlike conventional paradigms that passively propose harness from history rollouts, the meta agent empowers the embodied agent to actively deliberate and select prospective exploration paths to gather diagnostic physical evidence. Crucially, the outcomes of these exploratory interactions are translated into structured causal feedback, driving an outer-loop meta-agent to programmatically adjust and patch the operational harness. After this development phase, the proposed harness is evaluated afresh and added to a harness graph, which records the lineage of proposed harnesses, supporting the meta agent to actively retieve prior hypotheses, reuse of effective mechanisms, and exploration from earlier candidates for further harness evoluation.

We evaluate VISTA on LIBERO-Pro Zhou et al. (2025b) and RoboCasa Nasiriany et al. (2024). VISTA progressively improves task success as experience accumulates and achieves higher success rates with less token consumption than static reflection. These results support VISTA as an effective and efficient mechanism for converting embodied experience into improvements in the agent's executable harness. Our contributions are summarized as follows:

- We propose VISTA, a framework that departs from conventional handcrafted heuristics and static memory augmentation by framing the robot's operational harness as an evolvable, executable control plane, realizing autonomous embodied self-improvement.
- We introduce an active exploration mechanism where the meta-agent empowers the embodied system to actively gather diagnostic physical evidence, translating exploratory rollouts into structured causal feedback. Furthermore, we develop a Harness Graph to track the evolutionary lineage of candidate harnesses, facilitating hypothesis retrieval, component reuse, and principled branch exploration.
- We conduct comprehensive evaluations on the challenging LIBERO-Pro and RoboCasa benchmarks. Experimental results demonstrate that VISTA progressively improves manipulation success rates through accumulated experience, outperforming baselines while achieving higher token efficiency.

2 Related Work

2.1 Foundation Models for Robotic Manipulation

Low-level action policies. Foundation models for robotic manipulation can be broadly categorized according to how their semantic knowledge is connected to physical control. One line of research extends pretrained vision–language models (VLMs) into low-level policies through vision–language–action (VLA) models, which generate continuous action chunks using diffusion or flow matching (Black et al., 2024; Chi et al., 2025), task-space keyframes or spatial-grid actions (Qu et al., 2025), and latent action codes learned from robot trajectories or videos (Bu et al., 2025; Chen et al., 2025a). Recent video–action and world–action models further couple action prediction with future visual dynamics (Li et al., 2026d; Pan et al., 2026). Robot foundation models extend these capabilities across heterogeneous tasks and embodiments (Doshi et al., 2024; O'Neill et al., 2024), while subsequent studies improve action initialization, tokenization, and adaptation to new tasks and embodiments (Xu et al., 2026). Despite their strong control performance, these approaches translate general pretrained knowledge into embodiment-specific motor signals and often require additional robot trajectories when the task family or platform changes (Zhong et al., 2025).

Semantic decision making and physical grounding. A complementary line of research treats foundation models as semantic decision makers rather than direct motor controllers. These models predict subgoals, keypoints, affordances, value maps, or spatial constraints, which are subsequently realized by downstream controllers (Driess et al., 2023; Huang et al., 2023). Although this preserves high-level semantic reasoning, it separates model intent from physical execution and depends on system-specific grounding pipelines (Hu et al., 2026; Duan et al., 2024b), as well as long-horizon planning, closed-loop feedback, and failure recovery (Galanti et al., 2026). Together, these two directions reveal a trade-off between direct physical control and semantic generalization: low-level policies offer precise execution but are closely tied to specific embodiments, whereas intermediate representations preserve semantic structure but limit physical adaptability. This trade-off motivates interfaces that combine semantic interpretability with fine-grained executable actions.

2.2 Embodied Agent Systems

Agentic control architectures. Building on these foundation-model paradigms, agentic embodied systems place a pretrained VLM within an iterative harness that translates model decisions into robot actions and feeds execution outcomes back to the model, thereby closing the perception–decision–execution loop (Lima et al., 2026; Liu et al., 2026). Existing systems mainly differ in the primitives through which decisions are executed. Some models generate programs over perception and control APIs (Chen et al., 2024a; Fu et al., 2026), while others select skills from predefined libraries (Li et al., 2026b; Rana et al., 2023), steer learned policies such as VLAs with language subgoals (Chen et al., 2026; Peng et al., 2026), or delegate decisions to symbolic planners and numerical optimizers (Chen et al., 2024b; 2025b). To support long-horizon interaction, these systems typically maintain task context and persistent memory (Anwar et al., 2025), perform feedback-based monitoring and reflection (Huang et al., 2022b), and detect failures for replanning and recovery (Duan et al., 2024a).

Interfaces for embodied execution. Within agentic robot systems, interface design strongly affects control performance. While high-level, human-designed abstractions can improve reliability, composing low-level perception and control APIs remains difficult (Fu et al., 2026). Many systems therefore expose skills, controllers, or VLAs as callable primitives (Shi et al., 2025; Zhang et al., 2026), leaving the agent to decide what to do while an opaque executor determines how to perform it (Galanti et al., 2026). Although this simplifies planning, it reduces the model's role in fine-grained physical control. Similar compact action interfaces are used by computer-use and game agents (Lin et al., 2025; Ouyang et al., 2026), simulated embodied agents (Li et al., 2023; 2026c), and navigation systems [46, 75] (Zhou et al., 2026). Real-world manipulation still lacks a comparably simple and reusable interface because its control is high-dimensional, embodiment-specific, and spatially precise, motivating semantic action spaces with deterministic physical grounding and cross-embodiment transfer across diverse tasks and settings.

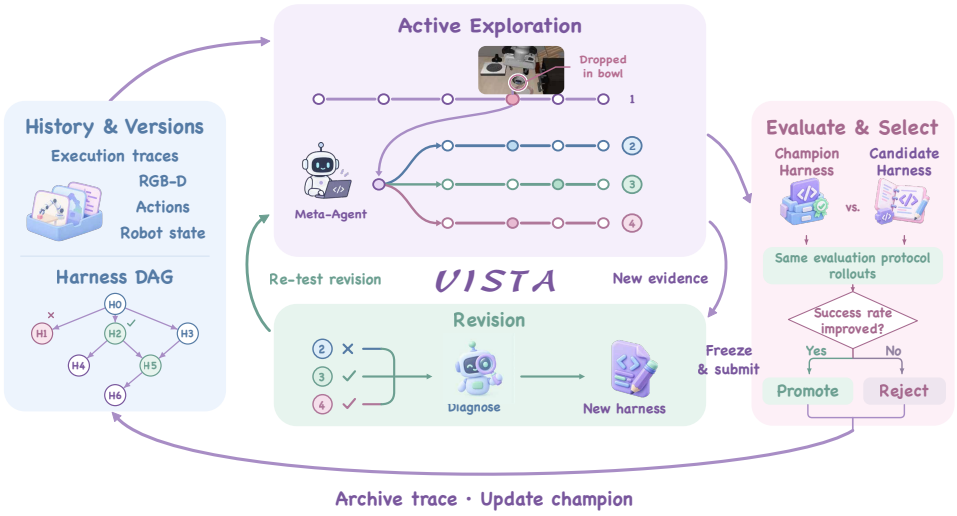


Figure 2: **Overview of VISTA.** The meta-agent develops an embodied harness from versioned implementations and multimodal execution evidence. Within a development iteration, it interleaves code revision, active exploration, and autonomous candidate revision. After submission, the frozen candidate undergoes an independent autonomous evaluation under the declared development conditions. The selection rule determines whether it replaces the incumbent; both promoted and rejected versions remain in the history. The inner loop acquires evidence and refines a candidate, while the outer loop selects and retains harnesses across development iterations.

3 METHOD

VISTA improves an embodied agent by evolving its harness, including the instructions, executable skills, and coordination logic through which the agent perceives, reasons, and acts. The online agent has access to a language-model planner, a VLA model, and public robot capabilities. VISTA keeps the underlying model parameters fixed and optimizes how these resources are composed into autonomous behavior. Figure 2 illustrates the resulting development loop. A meta-agent, implemented as a coding agent, retrieves a harness version and its execution evidence, identifies a behavior to improve, and develops a candidate. During development, it can actively investigate the environment, probe execution failures, and revise the harness in response to newly acquired evidence. It can also request autonomous tests of intermediate versions. The submitted candidate is then frozen and evaluated in fresh autonomous episodes. A host-controlled selection rule updates the incumbent, while the candidate and its evidence remain available for subsequent iterations. Improvement is thus carried forward through both executable harness versions and the experience accumulated while developing them.

3.1 HARNESS DAG

Let L_θ denote the online language model, π_ϕ the VLA, and $\mathcal{C}$ the public capability interface. The interface exposes observation, perception, policy inference, and bounded robot control. A harness is a package $H = (I, \mathcal{S}, P, M, D)$, where I specifies model-facing instructions, $\mathcal{S}$ contains skill descriptions and executable implementations, P provides hooks for request construction, tool selection, and feedback processing, M stores persistent procedural knowledge, and D declares tool bindings and capability dependencies. Skills may compose multiple perception or control operations, while policy hooks govern their use within the host's execution loop. Harness revisions can therefore change both the operations available to the agent and how it coordinates them. The model parameters, capability interface, and evaluation rule are fixed. For a task instance x, comprising a goal and scene initialization, and execution randomness ξ, let $S(H; x, \xi) \in \{0, 1\}$ denote official

autonomous task success. Our objective is

$$\max_{H \in \mathcal{H}_C} J(H), \qquad J(H) = \mathbb{E}_{x \sim \mathcal{T},\, \xi}\big[S(H; x, \xi)\big], \tag{1}$$

where $\mathcal{H}_C$ is the space of admissible harnesses and $\mathcal{T}$ is the task-instance distribution within the declared search scope. A frozen harness still induces closed-loop behavior: its code and persistent knowledge remain fixed during evaluation, while episode-local state and actions depend on incoming observations.

Harness development produces a directed acyclic graph (DAG) $G = (V, E)$ of immutable versions. We instantiate each harness as a Git-versioned package, with commits identifying snapshots and parent links defining lineage. Branches preserve alternative development directions, and explicit merges can combine histories. Execution records and development notes are linked to their corresponding versions. Each iteration begins from the incumbent by default, but the meta-agent may select another recorded base, including a previously rejected candidate, and inspect earlier implementations for reusable mechanisms. Retaining unsuccessful proposals makes their diagnostic findings available to later development.

3.2 Evidence-Guided Harness Revision

Each development record is tied to a frozen harness version and exact execution condition, including task initialization, repetition, randomness, and execution mode. A trajectory aligns planner decisions and tool calls with robot actions, images, and state, allowing the meta-agent to localize a failure from an aggregate outcome to a specific decision or physical execution interval.

Given a base harness and these records, the meta-agent forms a concrete hypothesis and revises the implicated components, such as instructions, observation processing, skill termination, or coordination logic. The resulting version and its supporting evidence are retained for later development. When recorded trajectories are insufficient to resolve the hypothesis, VISTA acquires additional embodied evidence through active exploration.

3.3 Active Diagnostic Exploration

Unlike static reflection, VISTA allows the meta-agent to collect new embodied evidence before submitting a candidate. It can launch a private simulator session from a frozen harness snapshot, inspect the current execution state, and invoke public sensing or control capabilities to test a specific hypothesis. These interactions provide diagnostic evidence for subsequent harness revisions rather than modifying the formal evaluation process.

After revision, the meta-agent can request fresh autonomous test rollouts to determine whether the candidate reproduces the desired behavior without intervention. Development therefore alternates among hypothesis formation, targeted interaction, harness revision, and autonomous testing until a candidate is submitted or the development budget is exhausted. Diagnostic traces and intermediate test results remain development evidence; promotion is determined only by the separate formal evaluation in 3.4.

3.4 Autonomous Validation and Selection

After submission, the host evaluates the frozen candidate through fresh episodes without meta-agent intervention. For a declared development matrix containing M task initializations and R repetitions per initialization, its empirical success is

$$\widehat{J}_{\mathrm{dev}}(H) = \frac{1}{MR} \sum_{i=1}^{M} \sum_{r=1}^{R} S(H; x_i, \xi_{i,r}). \tag{2}$$

Candidate and incumbent evaluations use matching task initializations, repetition settings, model configurations, and execution limits. Selection requires a complete, comparable matrix of valid autonomous outcomes. Matching incumbent evaluations may be reused. Under the success-based selection rule, the incumbent $H_k^\star$ is updated as

$$H_{k+1}^\star = \begin{cases} \widetilde{H}_k, & \widehat{J}_{\mathrm{dev}}(\widetilde{H}_k) > \widehat{J}_{\mathrm{dev}}(H_k^\star), \\ H_k^\star, & \text{otherwise}, \end{cases} \tag{3}$$

Algorithm 1 VISTA: active exploration for harness evolution

Require: Initial harness H_{init}, development matrix $\mathcal{D}_{\text{dev}}$, budgets
Evaluate H_{init} and initialize $H^\star \leftarrow H_{\text{init}}$ and development graph G
while budget remains and stopping rule is not met **do**
  if $H^\star$ requires confirmation **then**
    $\widetilde{H} \leftarrow H^\star$
  else
    Initialize development from a retained base in G (default: $H^\star$)
    while development budget remains and no candidate is submitted **do**
      Meta-agent autonomously selects and executes the next operation from evidence (history inspection, harness revision, live diagnosis, or autonomous testing)
      Add resulting versions and evidence to G
    end while
    Validate and freeze the submitted candidate $\widetilde{H}$
  end if
  Evaluate $\widetilde{H}$ autonomously on $\mathcal{D}_{\text{dev}}$ from fresh initializations
  Add $\widetilde{H}$ and its evaluation records to G
  if the evaluation is complete **then**
    Update $H^\star$ using Eq. equation 3 and update its confirmation state
  end if
end while
return $H^\star$

where $\widetilde{H}_k$ is the submitted candidate. Ties retain the incumbent. All frozen versions and their available records are retained, regardless of promotion.

Algorithm 1 summarizes the procedure. Search is bounded by the configured development and iteration budgets. When early stopping is enabled, the same selected package must achieve the required consecutive perfect formal evaluations; a confirmation iteration re-evaluates that package without another development attempt.

4 Experiments

4.1 Experiments Setup

Benchmarks. We conduct experiments on LIBERO-Pro (Zhou et al., 2025a) and RoboCasa365 Nasiriany et al. (2026). For LIBERO-Pro, we evaluate on across *Spatial*, *Object*, *Goal*, and *Libero-10* under instruction redirection. Each suite contains 10 tasks. For RoboCasa365, we consider the *Composite Seen* and *Composite Unseen* task categories. Each suite contains 16 tasks, and each tasks are evaluated on 16 seeds.

Models. We use $\pi_{0.5}$ Physical Intelligence et al. (2025) as the VLA backbone for LIBERO-Pro and RLDX-1 (Kim et al., 2026) for RoboCasa365. Across both benchmarks, the embodied planner is instantiated with GPT-5.6 Sol, and the meta agent is implemented as a coding agent powered by Codex and GPT-6, providing stronger reasoning and code-editing capabilities for iterative harness development. Following the settings in RLDX-1 on RoboCasa365 evaluation, each episode is capped at a per-task step horizon (300–2900 steps), which bounds per-rollout cost, prevents unbounded planner exploration, and makes large-scale harness evolution tractable under fixed compute.

Baselines. We compare our method with end-to-end VLA models, including OpenVLA (Kim et al., 2024), π_0 (Black et al., 2024), $\pi_{0.5}$ (Physical Intelligence et al., 2025), and RLDX-1 (Kim et al., 2026), and agentic embodied systems, including HarnessVLA Zhang et al. (2026). For LIBERO-Pro, our method starts from is a minimal harness H_0 exposing only the VLA and a small set of primitive tools. For RoboCasa365, we re-implement the HarnessVLA (Zhang et al., 2026) harness on our own infrastructure as our initial harness, with prompts, tool schemas, and the per-task memory corpus identical to the reference implementation.

Table 1: Success rate (%) on four LIBERO-Pro suites. H_0 is the initial harness and H_{VISTA} is the final selected harness.

Method	Spatial	Object	Goal	LIBERO-10	Mean
OpenVLA (Kim et al., 2024)	0.0	0.0	0.0	0.0	0.0
π_0 (Black et al., 2024)	0.0	0.0	0.0	0.0	0.0
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	40.0	65.0	40.0	50.0	48.8
HarnessVLA (Zhang et al., 2026)	79.0	79.0	62.0	62.0	70.4
Initial harness H_0	19.0	49.0	16.0	7.0	23.0
Envolved harness H_{VISTA}	98.0	100.0	99.0	89.0	97.0
Δ over H_0	+79.0	+51.0	+83.0	+82.0	+74.0

Table 2: Success rate (%) on Composite Seen and Composite Unseen task categories in RoboCasa365. H_0 is initialize from HarnessVLA.

Method	Composite Seen	Composite Unseen
$\pi_{0.5}$	6.8%	0.9%
RLDX-1	17.4%	6.8%
Initial harness H_0	18.4%	4.7%
Envolved harness H_{VISTA}	25.8 %	11.3 %
Δ over H_0	+7.8	+6.6

4.2 Main Results

Overall Performance. We evaluate **VISTA** against state-of-the-art end-to-end policies and static harness baselines across two benchmarks. As shown in Table 1, monolithic end-to-end models like OpenVLA (Kim et al., 2024) and π_0 (Black et al., 2024) completely collapse, and $\pi_{0.5}$ (Physical Intelligence et al., 2025) attains only 48.8% on average. Although HarnessVLA (Zhang et al., 2026) reaches 70.4% via static memory augmentation, our evolved harness H_{VISTA} achieves a superior mean success rate of 97.0%, outperforming HarnessVLA by $+26.6\%$ and yielding a substantial $+74.0\%$ gain over the initial harness H_0. On the multi-stage household tasks of RoboCasa365 , H_{VISTA} progressively improves the success rate from 16.4% to 24.2% on Composite Seen, and from 4.7% to 11.3% on Composite Unseen, consistently outperforming both the static baseline H_0 and the specialized policy RLDX-1 Kim et al. (2026).

Evolution over Development. Figure 3 visualizes how autonomous success changes over development rounds. Each curve is a macro-average over all tasks in one LIBERO-Pro suite. VISTA produces substantial improvements across all evaluated suites. For the relatively easier suites, only two to three development rounds are sufficient to raise success rates close to saturation, while the more challenging suites continue to improve over additional iterations.

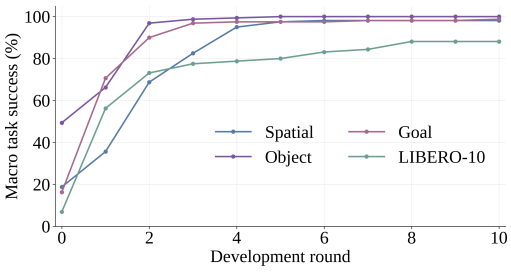


Figure 3: Evolution progress of success rate (%) on LIBERO-Pro suites.

Figure 4: Comparison of passive and active exploration on four LIBERO-Goal tasks. Δ is the mean improvement in success rate (percentage points) of the selected harness over the $\pi_{0.5}$ baseline; Token is the mean cumulative Meta tokens per task after each round.

Methods	R_1		R_2	
	Δ	Token	Δ	Token
Passive	+39.1	94.2M	+56.3	137.7M
Active	**+50.0**	**42.2M**	**+85.9**	**91.6M**

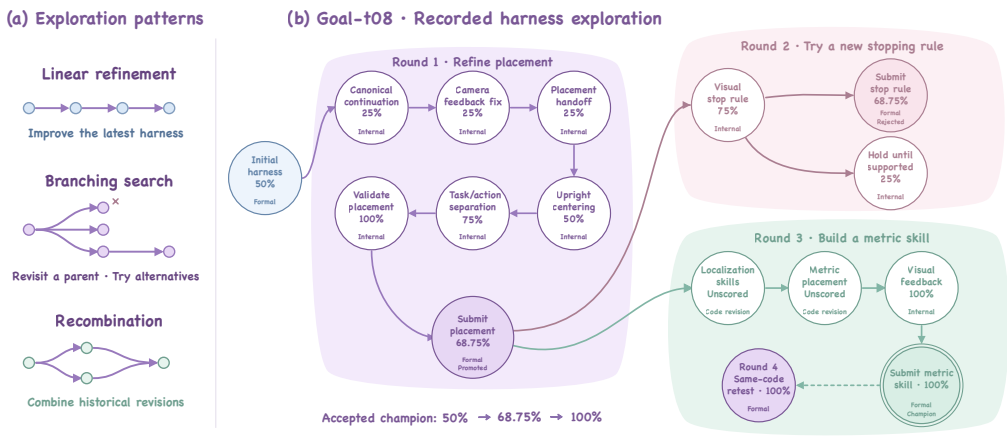


Figure 5: **Harness development trajectories.** (a) Representative development patterns observed during optimization, including linear refinement, branching exploration, and branch recombination. (b) A representative trajectory on LIBERO Goal-t08. *Internal* commit nodes denote small-scale active exploration and evaluation conducted by the Meta Agent during harness development, whereas *formal* commit nodes start evaluations on the complete 16-instance evaluation matrix and determine whether a candidate is promoted. Over the course of development, the harness improves from a 50% initial success rate to 100%.

4.3 Ablation

Active vs. passive exploration. To isolate the contribution of active exploration, we compare VISTA with a passive, rollout-only variant. The passive variant retains history inspection, harness editing, and autonomous candidate evaluation, but cannot open a live scene, probe a decision boundary, or execute diagnostic capabilities during development; it must diagnose failures from recorded rollouts alone. Both variants use the same Meta model, online Planner, frozen VLA, and formal evaluation matrix. To rule out budget as a confound, we match Meta resources per round: the passive variant may not submit until it has consumed the Meta tokens used by the corresponding active round, and the round is closed at 110% of that budget. Meta tokens include all input (cached and uncached) and output tokens of the Meta agent and its sub-agents.

Table 4 shows that active exploration yields larger gains at a lower cost. After the first round, active exploration improves success by 50.0 points using 42.2M Meta tokens per task, whereas the passive variant improves by 39.1 points while consuming 94.2M. The gap widens in the second round (+85.9 vs. +56.3 points) even though the passive variant still consumes 1.5× more tokens. The advantage is most pronounced on tasks whose failures cannot be disambiguated from recorded trajectories: on Goal-00, the passive variant remains at 0/16 after 287M tokens.

4.4 Analysis: Inside Harness Evolution

Development Trajectories. During optimization, the Meta Agent maintains a Git-like lineage to explore and revise harnesses. We analyze the recorded lineages to characterize how harnesses evolve across successive revisions. Depending on the task, we observe different development patterns. In some cases, optimization proceeds linearly by refining the current harness. In others, the optimizer branches from an earlier revision when the current hypothesis appears ineffective and explores an alternative direction. We also observe cases where mechanisms developed in different branches are recombined into a new harness. Figure 5(b) shows a representative trajectory on LIBERO Goal-t08. Starting from an initial harness with 50% formal success, the first development round refines the placement behavior and promotes a candidate with 68.75% success. The next round explores an alternative stopping rule, but the submitted candidate does not improve over the incumbent and is therefore rejected. A later branch introduces a metric placement skill and reaches 100% formal success, after which the same frozen harness is re-evaluated and again obtains 100%. This trajectory illustrates that individual development attempts need not improve monotonically: VISTA can retain

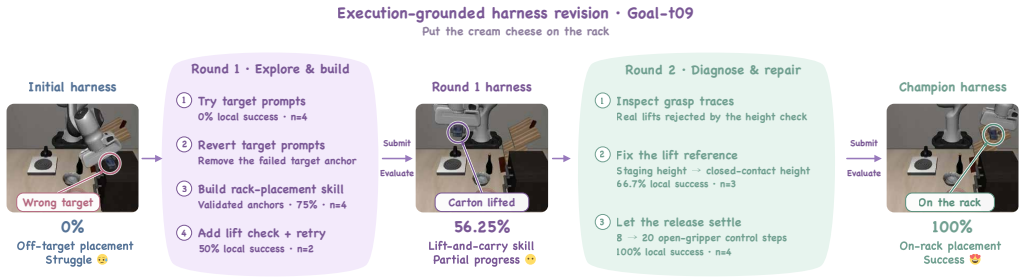


Figure 6: **A representative harness development process on Goal-t09.** The initial harness achieves $0/16$ successes. The first development round adds lift verification, a bounded re-localization and grasp retry, and an explicit transport-and-release procedure, improving performance to $9/16$. Analysis of the remaining failures then leads to a revision of the lift-state interpretation and release behavior, after which the final harness achieves $16/16$ on the development set.

the best-so-far harness while continuing to explore alternative hypotheses and reuse evidence from unsuccessful branches.

Acquired Harness Mechanisms. We examine the artifacts produced during harness development to understand what is accumulated through optimization. The revisions span perception interfaces, state verification, action composition, and the allocation of control between executable skills and the frozen VLA. These categories are not mutually exclusive: a single revision may change both how an object is localized and how the outcome of the subsequent manipulation is verified. For example, across the developed harnesses, the optimizer introduces observation-dependent manipulation targets, explicit grasp and execution-state verification, bounded recovery procedures, and programmatic action sequences that complement VLA control.

Figure 6 shows a representative example from Goal-t09. The initial harness succeeds in $0/16$ episodes. In the first development round, the meta-agent analyzes the failed trajectories and makes several concrete changes: it requires a verified lift before transport, re-segments the carton before a bounded grasp retry, and introduces an explicit transport-and-release procedure for placing the carton onto the rack. The resulting harness reaches $9/16$. The remaining failures expose a mismatch in the lift verification logic: lift was measured relative to an approach pose rather than the closed-contact state from which the carton was actually lifted. The second round therefore revises the detector to measure displacement over a continuous closed-contact segment and resets the accumulated evidence when contact is lost, together with further adjustments to the release behavior. After these revisions, the harness reaches $16/16$ on the development set. The accepted changes from the first round remain part of the harness in the second round, so later development builds directly on mechanisms acquired earlier rather than restarting from the initial system.

5 Conclusion

We introduced VISTA, a framework for improving embodied agents by evolving their operational harness while keeping the underlying language model and VLA fixed. Rather than treating completed rollouts as a static source of feedback, VISTA enables a meta-agent to actively acquire diagnostic evidence, revise executable harness components, and test intermediate solutions before submitting a frozen candidate for autonomous evaluation. Across LIBERO-Pro and RoboCasa, this process progressively improves task success over successive development rounds. These results demonstrate that active, execution-grounded harness development provides a practical mechanism for converting embodied interaction into persistent improvements in agent behavior.

Limitations and Future Works. While VISTA demonstrates substantial gains across simulated benchmarks, our current evaluations are conducted strictly within high-fidelity simulation environments. physical real-robot validation has not yet been performed. Real-world deployment inevitably introduces sensor noise, unpredictable actuation latency, and unmodeled contact dynamics. In future work, we plan to validate VISTA on physical robotic platforms (e.g., Franka Emika arms and mobile manipulators), integrating real-time tactile feedback and conservative kinematic compliance to guarantee hardware safety during real-world exploration.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2838–2845. IEEE, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Fusai, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, et al. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Junting Chen, Yao Mu, Qiaojun Yu, Tianming Wei, Silang Wu, Zhecheng Yuan, Zhixuan Liang, Chao Yang, Kaipeng Zhang, Wenqi Shao, et al. Roboscript: Code generation for free-form manipulation tasks across real and simulation. *arXiv preprint arXiv:2402.14623*, 2024a.

Siyi Chen, Hugo Hadfield, Alex Zook, Mikaela Angelina Uy, Chan Hee Song, Erwin Coumans, Xuning Yang, Faisal Ladhak, Qing Qu, Stan Birchfield, et al. Volo: A physical orchestrator for open-vocabulary long-horizon manipulation. *arXiv preprint arXiv:2606.07723*, 2026.

Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19752–19763. IEEE, 2025a.

Yongchao Chen, Jacob Arkin, Charles Dawson, Yang Zhang, Nicholas Roy, and Chuchu Fan. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. In *2024 IEEE International conference on robotics and automation (ICRA)*, pp. 6695–6702. IEEE, 2024b.

Yongchao Chen, Yilun Hao, Yang Zhang, and Chuchu Fan. Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 19248–19254. IEEE, 2025b.

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

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Vuong, et al. PaLM-e: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488. PMLR, 2023. URL https://proceedings.mlr.press/v202/driess23a.html.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024a.

Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*, 2024b.

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

Liane Galanti, Dhruv Shah, and Tri Dao. Addressing the orchestration gap in generalist robots via physical agency. *arXiv preprint arXiv:2607.21725*, 2026.

Jiaheng Hu, Mohit Shridhar, Caden Lu, Dhruv Shah, Hao-Tien Lewis Chiang, Jie Tan, and Annie Xie. What matters in orchestrating robot policies: A systematic study of hierarchical vla agents. *arXiv preprint arXiv:2606.10267*, 2026.

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*, 2022a. URL https://arxiv.org/abs/2207.05608.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022b.

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. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 540–562. PMLR, 2023. URL https://proceedings.mlr.press/v229/huang23b.html.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, et al. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, 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.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pp. 80–93. PMLR, 2023.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026a.

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

Siyao Li, Jiawei Gu, Shuai Liu, Kairui Hu, Zekun Li, Linjie Li, Chengcheng Tang, Po-Chen Wu, Ivan Shugurov, Lingni Ma, et al. Humanclaw: Can vision-language models act through a body? *arXiv e-prints*, pp. arXiv–2607, 2026c.

Ziang Li, Dongzhou Cheng, Yibin Wang, Shiyue Wang, Xiaoyang Xu, Lingxuan Weng, Juan Wang, and Jiaqi Wang. Light-wam: Efficient world action models with state-fusion action decoding. *arXiv preprint arXiv:2606.08242*, 2026d.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591. URL https://arxiv.org/abs/2209.07753.

Oscar Lima, Marc Vinci, Martin Günther, Marian Renz, Alexander Sung, Sebastian Stock, Johannes Brust, Lennart Niecksch, Zongyao Yi, Felix Igelbrink, et al. Agentic ai for robot control: Flexible but still fragile. In *Proceedings of the AAAI Symposium Series*, volume 8, pp. 465–473, 2026.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19498–19508. IEEE, 2025.

Haowen Liu, Xirui Li, Shaoxiong Yao, Peng Shi, Tianyi Zhou, Jia-Bin Huang, Furong Huang, and Jiayuan Mao. Guava: An effective and universal harness for embodied manipulation. *arXiv preprint arXiv:2606.18363*, 2026.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots, 2024. URL https://arxiv.org/abs/2406.02523.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2603.04356.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Mingyu Ouyang, Siyuan Hu, Kevin Qinghong Lin, Hwee Tou Ng, and Mike Zheng Shou. Gameworld: Towards standardized and verifiable evaluation of multimodal game agents. In *European Conference on Computer Vision*, pp. 41–59. Springer, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Bikang Pan, Fan Liu, Haotao Lu, Jingya Wang, and Ye Shi. Selfwam: A self-grounded unified world action model for fast robot control. *arXiv preprint arXiv:2608.00725*, 2026.

Jiaqi Peng, Xiqian Yu, Delin Feng, Yuqiang Yang, Wenzhe Cai, Jing Xiong, Ganlin Yang, Jinliang Zheng, Jiafei Cao, Xueyuan Wei, et al. Cortex: A bidirectionally aligned embodied agent framework for long-horizon manipulation. *arXiv preprint arXiv:2607.05377*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Galliker, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

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.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation, 2023.

Kechun Xu, Zhenjie Zhu, Anzhe Chen, Rong Xiong, and Yue Wang. Apt: Action expert pre-training improves instruction generalization of vision-language-action policies. *arXiv preprint arXiv:2606.12366*, 2026.

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

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

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. Version 4. Code: https://github.com/RLinf/RPent.

Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, et al. A survey on vision-language-action models: An action tokenization perspective. *arXiv preprint arXiv:2507.01925*, 2025.

Jian Zhou, Xunyi Zhao, Gengze Zhou, Zerui Li, Sihao Lin, Jiajun Liu, and Qi Wu. Embodied agents take control: Minimal-interface zero-shot agents rival industrial-scale policies in vision-and-language navigation. *arXiv preprint arXiv:2607.26148*, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025a. URL https://arxiv.org/abs/2510.03827.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2025b. URL https://arxiv.org/abs/2510.03827.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Vuong, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

APPENDIX

A AI ASSISTANCE DISCLOSURE

In this work, we used generative AI tools for assisting with translation, implementing partial code, supporting qualitative analysis of failure cases, and interpreting experimental results. We have not used generative AI tools for generating synthetic datasets, developing core theoretical or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proofs, assisting with proof writing, nor proposing or refining the main research hypothesis. Formulating mathematical proofs is not applicable to this work.

We have reviewed all AI-assisted work. All AI-generated code modules were independently inspected, debugged, and experimentally validated by two authors to guarantee functional correctness. AI-translated and AI-drafted text content was manually revised for technical accuracy, logical consistency, and academic tone. Qualitative observations suggested by generative AI were cross-checked against raw experimental outputs and human annotation statistics before being included in the manuscript. We take responsibility for the final content of this work, including all text, claims, metrics, and artifacts produced with the aid of generative AI.

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 INFRASTRUCTURE

VISTA separates harness development from autonomous evaluation through three interfaces. The meta-agent can read previous execution records, edit a candidate package, run live diagnostics, and request autonomous tests. The host freezes package versions, starts simulator processes, records traces, and applies the final selection rule. Figure A1 shows the three execution paths. Historical inspection reads completed records and does not recreate a previous simulator state. Live development starts a private scene from a frozen harness snapshot. Autonomous evaluation starts fresh episodes with a frozen candidate and no meta-agent intervention. All paths write records with explicit episode, request, and harness identities.

A live scene is not hot-swapped when the worktree changes. To test a later revision, VISTA starts a new session with the new frozen package. Live observations are stored as diagnostic evidence and are not silently added to the planner's context. Historical depth is available only for trace modes that archive metric depth and calibration; `full_rgb` records contain RGB but no historical depth.

B.2 META-AGENT SKILL AND TOOL SURFACE

The meta-agent follows the `evolve-harness` skill. This skill describes how to inspect history, formulate a concrete hypothesis, revise the harness, use live sessions for diagnosis, and validate the resulting change. It is a development protocol; it does not change the planner or VLA weights. Table A1 summarizes the main interfaces exposed to the meta-agent.

The harness package itself exposes a separate runtime surface to the online agent. In the LIBERO reference package, this surface includes native file and termination tools, image reading, an observation skill, and a VLA-step skill. A package may additionally define a `Policy` with `prepare_request`, `process_tool_result`, and optional `decide` hooks. These hooks operate inside the ordinary host planner loop: they can transform the visible request or choose among bound tools, but they do not replace the host loop or change the model weights and budgets.

The distinction between the two surfaces is deliberate. Meta tools control development and inspection; harness tools determine what the online agent can use during an autonomous episode. A live call made by the meta-agent therefore does not imply that the corresponding operation was available to, or selected by, the online agent in the formal evaluation.

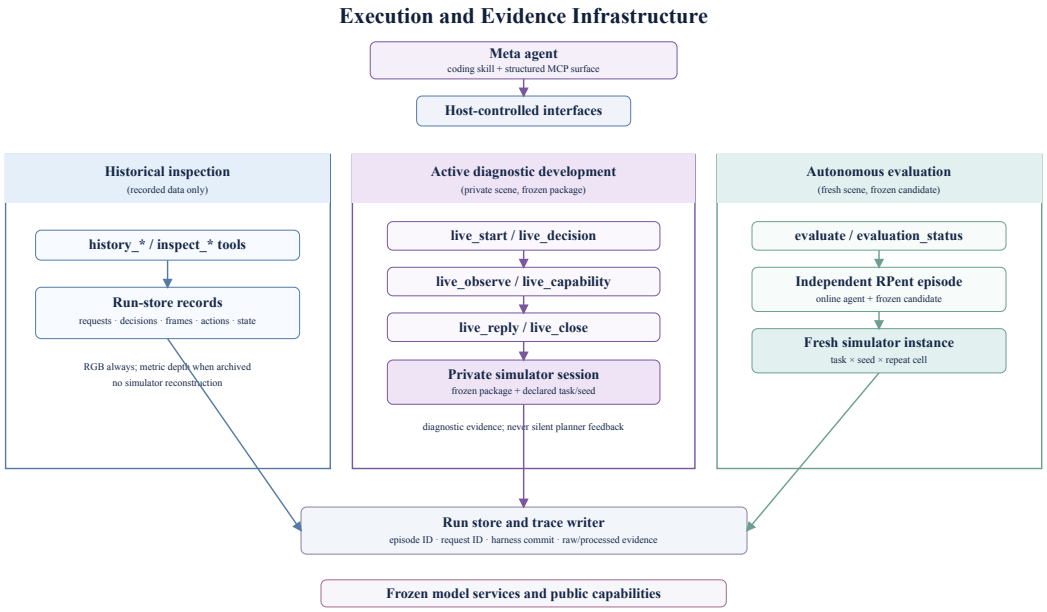


Figure A1: **Execution and evidence infrastructure.** The meta-agent communicates through a structured MCP interface. Historical inspection reads completed records; live development interacts with a private scene created from a frozen harness snapshot; and autonomous evaluation runs fresh episodes with no meta-agent intervention. All lanes use request, episode, and harness identities. Evidence is written to the run store, while package freezing and formal selection remain host-controlled. Historical depth is available only for trace modes that archive metric depth and calibration.

B.3 Git-Backed Harness History

Each frozen harness is stored as a Git commit. Ordinary revisions create parent edges, and an explicit merge creates a commit with multiple parents. The run store attaches evaluation requests, episode records, live diagnostics, and selection decisions to the corresponding commit.

Rejected candidates remain reachable through retained references and may be selected as later development bases. The incumbent used for formal comparison is recorded separately from the parent selected for revision. Thus, Git records code lineage, while the run store records the evidence and decisions associated with each version. Reusing a file or an idea does not create a Git ancestry edge.

C Additional Quantitative Results

C.1 Complete Per-Task Results

Tables A2 and A3 report complete per-task results on the development instances of LIBERO-Pro and RoboCasa, respectively. H0 denotes the unmodified harness, while Evolved denotes the retained champion. Both are evaluated on a 16-cell development matrix for each task.

LIBERO-Pro. Table A2 shows improvements over H0 on all 40 tasks across the four suites. The suite-average success rate increases from 19.0% to 98.0% on Spatial, from 49.0% to 100.0% on Object, from 16.0% to 99.0% on Goal, and from 7.0% to 89.0% on LIBERO-10. The retained champions achieve 100.0% success in the formal development evaluation on 34 of the 40 tasks, including every task in Object. The task-level breakdown also reveals differences that are less apparent from the suite averages. Six tasks remain below 100% success: two in Spatial, one in Goal, and three in LIBERO-10. In particular, LIBERO-10 T4 and T9 remain at 63.0% and 44.0%, respectively, despite improving substantially over their zero-success initial harnesses.

Table A1: Meta-agent interfaces in the VISTA implementation. Tools are grouped by their role in harness development.

Interface	Representative tools	Returned evidence or effect
History inspection	`history_summary`, `history_episodes`, `history_episode`, `history_decisions`, `history_decision`, `history_timeline`, `history_frame`, `inspect_decision`, `inspect_harness`	Versioned harness metadata, task/seed identities, exact planner decisions, tool arguments, raw and processed feedback, physical step ranges, images, and package contents.
Package development	Coding-agent file tools, `select_parent`, `budget`, `submit`	Candidate edits, saved drafts, selected Git parents, budget state, package validation, commit identity, and submission notes.
Active diagnostics	`live_start`, `live_decision`, `live_observe`, `live_capability`, `live_reply`, `live_close`	A private frozen-package session, decision-boundary observations, public capability probes, optional decision replacement, and diagnostic episode records.
Autonomous testing	`evaluate`, `evaluation_status`	A frozen candidate and fresh autonomous episodes for specified task–seed–repeat cells. Autonomous tests requested during development are diagnostic evidence only. Promotion is based on a separate host-initiated evaluation after submission.

Table A2: Per-task success rate (%) on LIBERO-Pro. $\pi_{0.5}$ denotes the direct-deploy baseline, H0 denotes the unmodified harness, and Evolved denotes the per-task champion after evolution. Δ is the gain over H0 in percentage points.

Suite	Method	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	Avg.
Spatial	π_{05}	0.0	100.0	0.0	0.0	0.0	50.0	100.0	50.0	0.0	100.0	40.0
	H0	0.0	94.0	0.0	0.0	0.0	56.0	19.0	0.0	13.0	6.0	19.0
	Evolved	100.0	100.0	100.0	100.0	94.0	100.0	100.0	88.0	100.0	100.0	98.0
	Δ	+100.0	+6.0	+100.0	+100.0	+94.0	+44.0	+81.0	+88.0	+88.0	+94.0	+79.0
Object	π_{05}	100.0	100.0	75.0	0.0	0.0	25.0	75.0	75.0	100.0	100.0	65.0
	H0	94.0	81.0	69.0	0.0	56.0	13.0	6.0	19.0	75.0	81.0	49.0
	Evolved	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	Δ	+6.0	+19.0	+31.0	+100.0	+44.0	+88.0	+94.0	+81.0	+25.0	+19.0	+51.0
Goal	π_{05}	0.0	100.0	25.0	25.0	100.0	0.0	75.0	50.0	25.0	0.0	40.0
	H0	0.0	6.0	31.0	0.0	31.0	0.0	25.0	19.0	50.0	0.0	16.0
	Evolved	100.0	100.0	88.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.0
	Δ	+100.0	+94.0	+56.0	+100.0	+69.0	+100.0	+75.0	+81.0	+50.0	+100.0	+83.0
LIBERO-10	π_{05}	25.0	75.0	100.0	0.0	0.0	75.0	100.0	50.0	75.0	0.0	50.0
	H0	0.0	25.0	38.0	0.0	0.0	0.0	6.0	0.0	0.0	0.0	7.0
	Evolved	100.0	100.0	100.0	88.0	63.0	100.0	100.0	100.0	100.0	44.0	89.0
	Δ	+100.0	+75.0	+63.0	+88.0	+63.0	+100.0	+94.0	+100.0	+100.0	+44.0	+83.0

RoboCasa. Table A3 reports the corresponding per-task results on Composite-Seen and Composite-Unseen. On Composite-Seen, the evolved harness increases aggregate success from 59/256 (23.0%) to 66/256 (25.8%), with gains concentrated in GetToastedBread and RinseSinkBasin. On Composite-Unseen, which is still under active development, the current snapshot improves from 12/256 (4.7%) to 29/256 (11.3%). Unlike Composite-Seen, the gains are distributed

Table A3: Complete per-task results on RoboCasa composite suites. The $\pi_{0.5}$ column reports the official RC365 evaluation (per-task success rate, %). H0 and Evolved report successes out of the 16-cell matrix. Composite-Unseen evolution is still in progress and reflects the current snapshot.

Composite-Seen

Task	$\pi_{0.5}$ (%)	H0	Evolved	Δ
DeliverStraw	0	1/16	1/16	+0
GetToastedBread	0	0/16	5/16	+5
KettleBoiling	2	3/16	3/16	+0
LoadDishwasher	2	7/16	7/16	+0
PackIdenticalLunches	4	3/16	3/16	+0
PreSoakPan	26	5/16	5/16	+0
PrepareCoffee	2	0/16	0/16	+0
RinseSinkBasin	4	5/16	7/16	+2
ScrubCuttingBoard	2	8/16	8/16	+0
SearingMeat	0	0/16	0/16	+0
SetUpCuttingStation	10	4/16	4/16	+0
StackBowlsCabinet	26	10/16	10/16	+0
SteamInMicrowave	2	2/16	2/16	+0
StirVegetables	6	5/16	5/16	+0
StoreLeftoversInBowl	0	0/16	0/16	+0
WashLettuce	22	6/16	6/16	+0
Avg / Total	**6.8**	**59/256**	**66/256**	**+7**

Composite-Unseen (in progress)

Task	$\pi_{0.5}$ (%)	H0	Evolved	Δ
ArrangeBreadBasket	2	2/16	3/16	+1
ArrangeTea	0	0/16	0/16	+0
BreadSelection	2	0/16	3/16	+3
CategorizeCondiments	0	0/16	3/16	+3
CuttingToolSelection	8	1/16	2/16	+1
GarnishPancake	0	5/16	5/16	+0
GatherTableware	0	0/16	1/16	+1
HeatKebabSandwich	0	0/16	0/16	+0
MakeIceLemonade	0	1/16	1/16	+0
PanTransfer	0	0/16	1/16	+1
PortionHotDogs	0	0/16	1/16	+1
RecycleBottlesByType	0	0/16	0/16	+0
SeparateFreezerRack	0	0/16	0/16	+0
WaffleReheat	0	2/16	4/16	+2
WashFruitColander	2	0/16	2/16	+2
WeighIngredients	0	1/16	3/16	+2
Avg / Total	**0.9**	**12/256**	**29/256**	**+17**

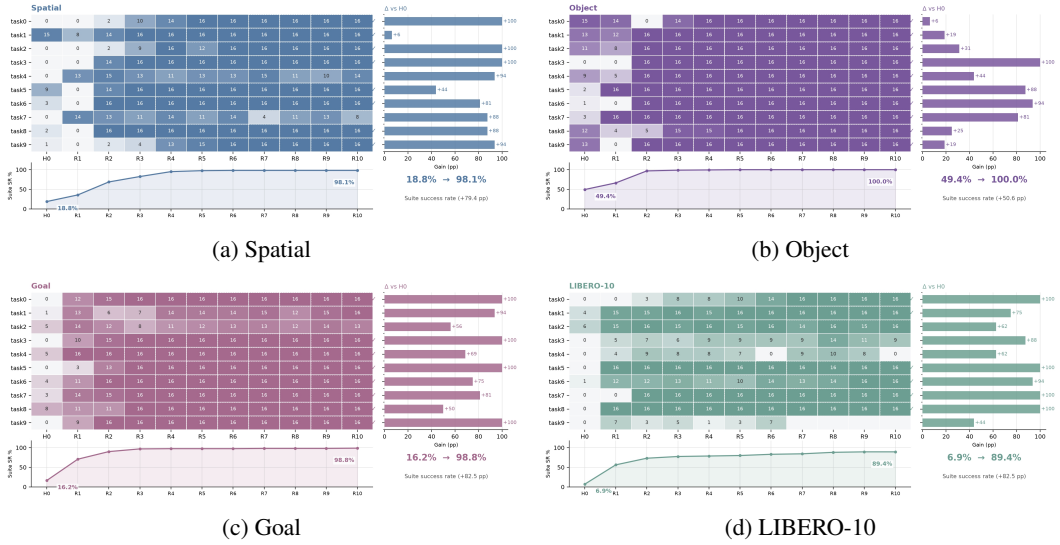
across a broader set of tasks: 10 of the 16 tasks improve over H0, with the largest current increases on BreadSelection and CategorizeCondiments. These results should be interpreted as an intermediate snapshot rather than the final Composite-Unseen result.

C.2 Complete Evolution Details

Figure A2 presents the recorded development trajectories for all four LIBERO-Pro suites. The heatmaps show the performance of the candidate evaluated in each round, whereas the curves below aggregate the best-so-far champion scores across tasks. The right-hand panels summarize the gain of each retained champion over H0. This distinction separates the quality of individual proposals from the performance accumulated through candidate selection.

Across the four suites, the largest aggregate improvements occur during the early rounds, followed by smaller, task-dependent gains. Object reaches perfect development performance, while Spatial and Goal approach saturation. LIBERO-10 improves more gradually and retains a larger performance gap at the reporting endpoint. The per-task trajectories further show that progress is not synchronized across tasks: some reach high success rates within the first few rounds, whereas others continue to receive revisions throughout the displayed development period. For example, Goal T1 reaches 100.0% only in R10.

Individual candidate scores are often non-monotonic, even when the retained champion remains unchanged. For example, Spatial T7 reaches 87.5% in R1, but several subsequent candidates obtain lower scores without replacing that champion. A plateau in the aggregate curve therefore does not necessarily indicate that development has stopped; it may instead reflect continued exploration without a successful replacement. Likewise, the non-decreasing champion curves follow from best-so-far selection and do not imply that every proposed revision improves performance.



(a) Spatial (b) Object

(c) Goal (d) LIBERO-10

Figure A2: Per-task evolution heatmaps on LIBERO-Pro. Rows are tasks (task0–task9), columns are evolution rounds (H0 = unmodified harness). Cell values are successes out of the 16-cell formal matrix for the candidate evaluated that round. Right panels: per-task gain over H0 (percentage points). Bottom: suite-aggregate success rate using the best-so-far champion per task.