EMHO: EMBODIED AGENT HARNESS OPTIMIZATION VIA EXPERIENCE TRACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving embodied agents often focuses on optimizing the underlying model through training, while the surrounding agent harness that controls planning, context, and tool use is typically engineered. We ask whether this harness can instead improve itself directly from experience traces under sparse environmental feedback. We propose **EMbodied Agent Harness Optimization (EMHO)**, a self-evolving framework that keeps the embodied model frozen and iteratively revises its harness by analyzing execution trajectories and prior harness history. EMHO optimizes beyond skills or recovery prompts, modifying how the agent monitors progress, uses vision tools, grounds observations, and responds to failures. To support multiple subtasks with a single harness, we introduce **EMHO-Merge**, which addresses trade-offs in jointly optimizing a single shared harness across subtasks by using episode-level gains and losses to guide evidence-supported refinement of when and how revised behaviors are applied. We evaluate EMHO on EmbodiedBench across navigation and manipulation tasks, and EMHO consistently improves task success for both Qwen 9B and 27B models. Qualitative analysis shows that EMHO goes beyond recovering from failures and unproductive actions to reshape how the embodied agent interprets and interacts with its environment.

1 INTRODUCTION

Recent progress in embodied agents has largely been driven by advances in the underlying vision-language models (Driess et al., 2023; Physical Intelligence et al., 2025; Liu et al., 2024; Kim et al., 2025). Yet a better embodied agent does not always require a better model. Even when the underlying vision-language model is frozen, changes to the surrounding agent system can substantially alter what the agent observes, how it reasons, which actions it attempts, and how it responds when those actions fail. This surrounding system is referred to as *harness*: the logic that organizes planning, context and memory, tool selection, observation presentation, and failure recovery.

Designing a strong embodied-agent harness, however, remains largely manual because failures are difficult to diagnose from interaction traces. An episode may span many observations, plans, actions, and tool calls, while the final outcome provides little indication of which earlier decision caused the failure. This contrasts with coding and command-line environments, where failed tests, exceptions, or localized execution errors can directly guide self-improvement (Zhang et al., 2026; Lee et al., 2026). Embodied agents instead often receive only sparse action-level validity feedback and a delayed episode-level success or failure signal, making it more difficult and costly to identify recurring weaknesses and determine how the harness should be revised.

Recent work on embodied self-improvement has focused on refining specific components of the agent, such as reusable skills or recovery mechanisms (Ju et al., 2026; Yang et al., 2026; Lu et al., 2026). While these approaches can improve particular behaviors or responses to failure, an embodied agent's behavior is determined by a broader set of interacting mechanisms, including planning, context and memory construction, external vision tool usage, visual grounding and failure recovery. When a weakness arises from one of these surrounding mechanisms, updating only a skill or recovery component may leave the underlying source of the problem unchanged. This raises a question: *can an embodied agent improve this broader harness itself from sparse and weakly diagnostic interaction feedback?* We introduce **EMbodied Agent Harness Optimization** (**EMHO**), a self-evolving framework that keeps the embodied model frozen and iteratively optimizes the surrounding

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.

harness. In the inner loop, the agent executes embodied tasks and records trajectories containing observations, plans, actions, tool usage, environmental feedback, and final outcomes. In the outer loop, a harness optimizer analyzes these trajectories together with the history of previous harness modifications and their associated outcomes, identifies recurring behavioral weaknesses, and proposes targeted edits to the harness.

To further explore whether one harness can perform well across related subtasks, we extend EMHO from subtask-specific optimization to joint optimization over a combined search set. When optimizing a single harness across multiple subtasks, a revision that improves performance on one subtask may reduce performance on another. We introduce **EMHO-Merge** to use these gains and losses to guide further refinement. It compares each candidate with its parent harness to identify episodes that change from failure to success and those that change from success to failure. By examining the corresponding trajectories before and after the revision, it looks for observable conditions that distinguish when the revised behavior helps from when it hurts. When supported by this evidence, the optimizer adjusts when or how the behavior is applied, aiming to retain the gains while reducing the losses.

We evaluate EMHO on EmbodiedBench (Yang et al., 2025) across navigation and manipulation tasks. EMHO substantially improves task success over the initial harness, with gains of up to 23.3% in navigation and 33.3% in manipulation. When optimizing a single shared harness across subtasks, EMHO-Merge achieves higher average search set success than EMHO on the pooled subtasks in both navigation and manipulation. Beyond improving failure recovery, our analysis shows that EMHO reshapes how the agent perceives and acts in the environment, changing how it monitors progress, orchestrates perceptual tools, and grounds visual observations into subsequent actions.

Our contributions are summarized as follows:

- **Self-evolving harness for embodied agents via EMHO.** We suggest EMHO, which evolves the broader harness around a frozen model from interaction traces under sparse and weakly diagnostic environmental feedback, moving beyond adaptation restricted to skills or recovery instructions.
- **Shared harness optimization via EMHO-Merge.** We extend EMHO to optimize a single shared harness across subtasks toward more general-purpose embodied behavior. EMHO-Merge explicitly contrasts episode-level gains and losses and uses paired trajectories to guide evidence-supported repairs, aiming to retain useful adaptations while mitigating trade-offs during joint optimization.
- **EMHO's ability to reshape agents' behavior.** EMHO improves task success, and qualitative analysis shows that the evolved harness improves failure recovery and reshapes how the agent monitors progress, uses perceptual tools, and grounds visual observation.

2 Related Work

Coding-Agent Harness Optimization. Recent work has begun to automate the optimization of coding-agent harnesses using execution traces and evaluation outcomes to revise prompts and skills. Meta-Harness (Lee et al., 2026) searches over executable harness code using the source code, execution traces, and scores of previously evaluated candidates, while Self-Harness (Zhang et al., 2026) identifies model-specific weaknesses and proposes targeted modifications that are retained through regression-based validation. Related approaches improve observability over editable agent components (Lin et al., 2026) or combine failure diagnosis and validation to produce persistent harness updates (Park et al., 2026). These methods have primarily been studied in coding and command-line environments, where failures often provide relatively explicit diagnostic signals such as failed tests, exceptions, or localized execution errors. Embodied environments, in contrast, may provide only sparse action-level feedback during interaction and a delayed task-level success signal.

Self-Improvement in Embodied Agents. Embodied-agent research has similarly explored improving behavior from interaction experience while keeping the underlying model fixed. Existing approaches concentrate self-improvement on a specific behavioral component, such as reusable skills, task-specific policies, or recovery mechanisms. EmbodiSkill (Ju et al., 2026) revises skills through trajectory-based reflection, VASO (Yang et al., 2026) uses verification and counterexamples to guide skill evolution, ASPIRE (Lu et al., 2026) improves robotic behavior by iteratively evolving task-specific code and accumulating successful repairs as reusable skills, and Zetta (Ding et al., 2026) focuses on failure detection and recovery by evolving runtime critics together with corresponding

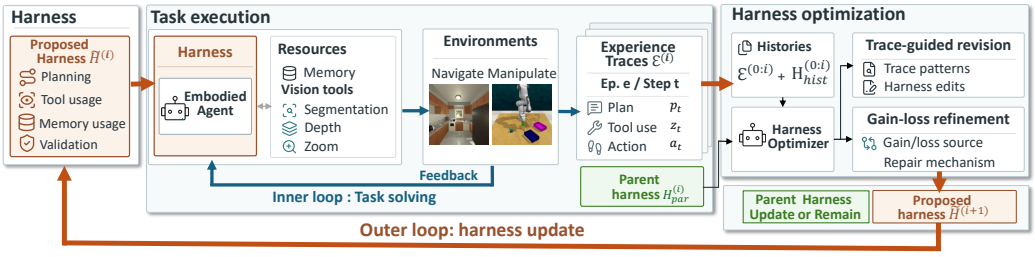


Figure 1: Overview of EMHO and EMHO-Merge. During harness optimization, EMHO uses trace-guided revision, while EMHO-Merge adds gain–loss refinement.

recovery behaviors. Our work extends this direction by treating the broader harness as the object of optimization, rather than focusing on a particular skill or recovery component. We further study how a single shared harness can be optimized across subtasks with differing behavioral demands.

3 Embodied Agent Harness Optimization (EMHO)

We optimize the harness H surrounding a frozen vision-language model M. The harness consists of the agent-side instructions and runtime logic that structure interaction with the environment, including observation and context construction, memory management, planning, tool use, action generation, output validation, and failure recovery. Let $\mathcal{H}$ denote the space of possible harnesses. Our objective is to find a harness that maximizes expected task success:

$$H^\star = \arg\max_{H \in \mathcal{H}} \mathbb{E}_{e \sim \mathcal{D}} \left[y_e(M, H) \right], \tag{1}$$

where $y_e(M, H) \in \{0, 1\}$ denotes the final outcome of episode e. Throughout optimization, only the harness is modified while the embodied model and environment remain fixed. EMHO alternates between two stages: an inner loop that executes the current harness on embodied tasks and records experience traces, and an outer loop that analyzes these traces alongside the history of previous optimization attempts to propose a targeted harness revision.

3.1 EMHO

Inner Loop: Experience Trace Collection At optimization iteration i, the agent executes the current harness $H^{(i)}$ on each episode $e \in \mathcal{D}_{\text{search}}$. The resulting execution trace is recorded as

$$\mathcal{T}_e^{(i)} = \left(x_e, \{(o_t, p_t, z_t, a_t, f_t)\}_{t=1}^{T_e}, y_e \right), \quad \mathcal{E}^{(i)} = \left\{ \mathcal{T}_e^{(i)} \mid e \in \mathcal{D}_{\text{search}} \right\}. \tag{2}$$

Here, x_e specifies the task and y_e denotes the final episode outcome. At interaction step t, o_t, p_t, z_t, a_t, and f_t denote the observation, generated plan, optional tool interaction, executed action, and environmental feedback, respectively. Embodied feedback is often sparse and weakly diagnostic, as a failed action or unsuccessful episode does not directly reveal the underlying cause.

Outer Loop: History-Aware Harness Refinement The outer loop maintains a current parent harness $H^{(i)}_{\text{par}}$, which serves as the basis for the next revision. The optimizer also receives the complete experience and harness histories accumulated through iteration i. It proposes a candidate harness and updates the parent according to

$$\widetilde{H}^{(i+1)} = \mathcal{O}\left(H^{(i)}_{\text{par}}, \mathcal{E}^{(0:i)}, \mathbf{H}^{(0:i)}_{\text{hist}} \right), \quad H^{(i+1)}_{\text{par}} = \begin{cases} \widetilde{H}^{(i+1)}, & \text{if } s\left(\widetilde{H}^{(i+1)}\right) > s\left(H^{(i)}_{\text{par}}\right), \\ H^{(i)}_{\text{par}}, & \text{otherwise.} \end{cases} \tag{3}$$

Here, $s(H) = \frac{1}{|\mathcal{D}_{\text{search}}|} \sum_{e \in \mathcal{D}_{\text{search}}} y_e(M, H)$ denotes the success rate of harness H on the search set. $\mathcal{E}^{(0:i)}$ denotes all experience traces collected from the initial evaluation through iteration i, including traces from both accepted and rejected candidates. Similarly, $\mathbf{H}^{(0:i)}_{\text{hist}}$ denotes the complete harness

history, including the previously evaluated harnesses, the modifications applied to them, and their evaluation outcomes. Using these histories, the optimizer identifies recurring behavioral weaknesses and proposes a targeted modification of the current parent harness. The candidate is evaluated and replaces the parent only if it achieves a higher search-set success rate. An example of the harness optimizer prompt is provided in Appendix A.

3.2 EMHO-MERGE

Gain–Loss-Aware Refinement. To move toward a more general-purpose embodied agent, we optimize a single shared harness over the combined search set of multiple subtasks. Since a revision can improve some episodes while degrading others, EMHO-Merge uses these mixed outcomes to guide refinement of when and how the revised behavior is applied. EMHO-Merge extends the outer loop of EMHO by using episode-level outcome changes from the most recently evaluated candidate to guide the next proposal. The candidate $\widetilde{H}^{(i)}$ is compared with its parent harness $H_{\mathrm{par}}^{(i-1)}$. Episodes that change from failure to success are gains $\mathcal{G}^{(i)}$, while those that change from success to failure are losses $\mathcal{L}^{(i)}$. When both occur, the optimizer may repair the previous harness revision by adjusting how or when the behavior is applied, rather than introducing a different behavior. It examines paired gain and loss traces to identify observable conditions under which the behavior is expected to improve task success, aiming to preserve the gains while reducing the losses. Each original revision is allowed at most one evaluated repair, which is proposed and evaluated as the candidate in the next optimization iteration. The optimizer determines whether repair is available and proposes the next candidate as

$$r^{(i)} = \mathbb{I}\Big[|\mathcal{G}^{(i)}| > 0 \wedge |\mathcal{L}^{(i)}| > 0 \wedge \text{repair unused}\Big], \quad \widetilde{H}^{(i+1)} = \mathcal{O}\Big(H_{\mathrm{par}}^{(i)}, \mathcal{E}^{(0:i)}, \mathcal{G}^{(i)}, \mathcal{L}^{(i)}; r^{(i)}\Big). \quad (4)$$

Here, $r^{(i)} = 1$ indicates that the revision evaluated in candidate $\widetilde{H}^{(i)}$ is available for repair in the next iteration. If repair is unavailable or unsupported by the collected experience, the optimizer proposes a different behavioral change. The next candidate $\widetilde{H}^{(i+1)}$ is constructed from the current parent $H_{\mathrm{par}}^{(i)}$, allowing experience from a rejected candidate to guide the next revision.

4 QUANTITATIVE RESULTS AND DISCUSSION

We conduct experiments on EmbodiedBench (Yang et al., 2025), covering both navigation and manipulation tasks. We use the Qwen3.5-9B model (Qwen Team, 2026a) and the Qwen3.8-27B-FP8 model (Qwen Team, 2026b) as the underlying embodied agents. We evaluate both models in the main results and use the 27B model for all other experiments unless otherwise stated. For visual perception, the agent is equipped with Depth Anything 3 (Lin et al., 2025) for depth estimation, SAM 3 (Carion et al., 2026) for segmentation, and zoom tools. We initialize the agent harness based on the original EmbodiedBench implementation and perform 10 iterations of harness optimization for each task. For each task, we divide the benchmark evenly into disjoint search and held-out sets. We use the search set to collect experience traces and optimize the harness. The harnesses produced during optimization are also evaluated on the held-out set. Details of the harness structure are provided in Appendix A.

4.1 MAIN RESULTS

As shown in Table 1, EMHO improves performance on the search set for navigation and manipulation, and the evolved harnesses also achieve higher success than the initial harness on held-out episodes. The improvements also extend beyond the relatively simple Base tasks, appearing across diverse subtasks that require longer-horizon reasoning, visual grounding, and state-aware manipulation. Notably, the gains are observed for both the 9B and 27B models, the evolved 9B navigation harness reaches performance competitive with larger published models on search set, while the 27B model also benefits considerably from harness evolution, particularly in manipulation. To further analyze these gains, we ablate the contributions of experience traces and harness-revision history. As reported in Appendix C, both sources of context improve EMHO's performance.

4.2 COMPARISON WITH FRONTIER-MODEL ONE-SHOT REVISION

We use one-shot harness revisions generated by Claude Opus-5 (Anthropic, 2026) and GPT-5.6 Sol (OpenAI, 2026) as baselines to assess the gains achieved by EMHO. For each domain, these

Table 1: Success rates on the navigation and manipulation tasks for each subtask. Initial harness and EMHO results are reported separately for the search and held-out splits. EMHO entries report the best score attained on each split over the optimization iterations. Bold entries denote the highest score. More details on the baseline and subtask-level results are provided in Appendix B.

Method / Model	Navigation					
	Avg	**Base**	**Common**	**Complex**	**Visual**	**Long**
GPT-5	56.8	62.0	62.0	68.0	50.0	42.0
Claude-4.6-Sonnet	62.2	68.0	63.0	**77.0**	55.0	48.0
Gemini-2.0-flash	48.7	63.3	65.0	50.0	51.7	13.3
InternVL3-78B	53.7	66.7	63.3	61.7	45.0	31.7
gemma-3-27b-it	45.3	53.3	45.0	61.7	50.0	16.7
One-Shot Harness w/ Opus (Qwen3.5-9B)	53.3	56.7	60.0	63.3	36.7	50.0
One-Shot Harness w/ Sol (Qwen3.5-9B)	49.3	50.0	53.3	63.3	40.0	40.0
One-Shot Harness w/ Opus (Qwen3.8-27B)	67.7	78.3	70.0	75.0	58.3	56.7
One-Shot Harness w/ Sol (Qwen3.8-27B)	65.7	73.3	70.0	61.7	63.3	60.0
Initial Harness (Search, Qwen3.5-9B)	50.0	58.3	46.7	65.0	40.0	40.0
Initial Harness (Held-out, Qwen3.5-9B)	48.7	58.3	53.3	58.3	33.3	40.0
EMHO (Search, Qwen3.5-9B)	63.0	71.7	70.0	70.0	50.0	53.3
EMHO (Held-out, Qwen3.5-9B)	56.7	66.7	60.0	65.0	40.0	51.7
Initial Harness (Search, Qwen3.8-27B)	54.7	60.0	60.0	50.0	50.0	53.3
Initial Harness (Held-out, Qwen3.8-27B)	66.0	73.3	63.3	66.7	66.7	60.0
EMHO (Search, Qwen3.8-27B)	67.3	76.7	**76.7**	63.3	63.3	56.7
EMHO (Held-out, Qwen3.8-27B)	**75.3**	**86.7**	**76.7**	73.3	**76.7**	**63.3**

Method / Model	Manipulation					
	Avg	**Base**	**Common**	**Complex**	**Visual**	**Spatial**
GPT-5	26.8	21.0	25.0	31.0	22.0	35.0
Claude-4.6-Sonnet	37.6	40.0	35.0	**40.0**	36.0	37.0
Gemini-2.0-flash	16.5	14.6	8.3	14.6	13.9	31.3
InternVL3-78B	26.3	29.2	22.9	22.9	25.0	31.3
gemma-3-27b-it	17.5	25.0	16.7	16.7	8.3	20.8
One-Shot Harness w/ Opus (Qwen3.5-9B)	13.2	10.4	16.7	12.5	13.9	12.5
One-Shot Harness w/ Sol (Qwen3.5-9B)	11.8	12.5	6.3	10.4	11.1	18.8
One-Shot Harness w/ Opus (Qwen3.8-27B)	27.8	22.9	20.8	29.2	38.9	27.1
One-Shot Harness w/ Sol (Qwen3.8-27B)	25.3	25.0	25.0	27.1	22.2	27.1
Initial Harness (Search, Qwen3.5-9B)	9.7	8.3	8.3	6.3	11.1	14.6
Initial Harness (Held-out, Qwen3.5-9B)	10.7	6.3	4.2	8.3	22.2	12.5
EMHO (Search, Qwen3.5-9B)	18.3	14.6	12.5	16.7	25.0	22.9
EMHO (Held-out, Qwen3.5-9B)	21.0	14.6	20.8	18.8	36.1	14.6
Initial Harness (Search, Qwen3.8-27B)	23.9	16.7	12.5	29.2	44.4	16.7
Initial Harness (Held-out, Qwen3.8-27B)	26.4	25.0	25.0	29.2	27.8	25.0
EMHO (Search, Qwen3.8-27B)	**44.2**	**50.0**	**41.7**	37.5	**50.0**	**41.7**
EMHO (Held-out, Qwen3.8-27B)	39.4	41.7	**41.7**	33.3	38.9	**41.7**

models receive the initial harness provided in EmbodiedBench, representative task examples, and are asked to produce a single improved harness. Although one-shot revision improves the initial harness, EMHO performs better across both domains. One-shot revisions tend to provide broad guidance, such as changing strategy when a loop occurs, without specifying how the agent should recognize and respond to the failure. EMHO instead converts recurring patterns into concrete operating conditions. For example, when a movement is blocked, the evolved harness terminates the remaining actions in the current plan, avoids retrying the same direction, and replans from the updated observation. This example illustrates how interaction experience helps EMHO develop specific execution rules beyond the broad guidance of one-shot revisions.

4.3 Full-Harness versus Skill-Only Optimization

Skill-only optimization primarily improves behavior by accumulating and refining textual recovery guidance. Across iterations, the learned skills become increasingly specific about how to respond to recurring failures, such as blocked navigation or repeated ineffective actions. Full-harness optimization discovers similar recovery principles, but is not limited to expressing them as instructions: it can also modify how the interaction history is represented, add mechanisms that explicitly detect

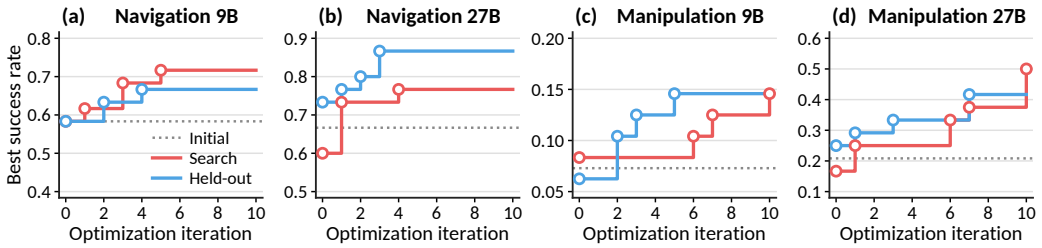


Figure 2: Quantitative results on the navigation and manipulation base subtask across optimization iterations. Full results are reported in Appendix B.

Table 2: Success-rate gains in percentage points over the initial harness on the search set, comparing skill-only updates with full-harness optimization using EMHO.

(a) Navigation

Method	Base	Common	Complex	Visual	Long
Skill-only	+10.0	+10.0	0.0	+10.0	+3.3
EMHO	+16.7	+16.7	+13.3	+13.3	+3.3

(b) Manipulation

Method	Base	Common	Complex	Visual	Spatial
Skill-only	+16.7	+8.3	+8.3	+11.1	+8.3
EMHO	+33.3	+29.2	+8.3	+5.6	+25.0

repeated or failed behavior, and ensure that the resulting state is exposed to subsequent planning. Moreover, it can intervene beyond recovery by changing progress monitoring, tool orchestration, and grounding behavior. This broader optimization space is reflected in the substantially larger average performance gains of full-harness optimization in Table 2. The comparison suggests that the benefit of harness evolution comes not only from discovering better behavioral rules, but also from changing the surrounding system so that those rules can be detected, triggered, and enforced during interaction.

4.4 Shared Harness Optimization with EMHO-Merge

When optimizing a shared harness across subtasks, a revision that helps one subtask may interfere with another. Separately optimized navigation harnesses often evolve different responses to blocked movement: Base favors an immediate action change, visual-appearance prefers directions supported by visible open space, and long-horizon first reassesses whether a direction change is necessary. Simply pooling experience across these subtasks can therefore introduce conflicting behaviors. For example, a rule that moves laterally before proceeding forward after a blocked or repeated approach helped Base navigation, but repeatedly triggered in a long-horizon trajectory, redirecting the agent into nearby obstacles. We evaluate EMHO-Merge on three subtasks per domain against EMHO (Separate) and EMHO (Combined), as shown in Table 3. EMHO-Merge uses gain–loss trajectories to refine when a behavior should activate; for instance, obstacle avoidance is restricted to cases with evidence of blockage. It achieves higher average success than EMHO (Combined) in both navigation and manipulation, suggesting that gain–loss-aware refinement better manages cross-subtask trade-offs.

5 Qualitative Analysis of Harness Evolution

Aggregate success rates show whether harness optimization improves performance, but not how the resulting behavior changes. We therefore analyze the edits discovered by EMHO and their effects on held-out trajectories, focusing primarily on results from the 27B model. We categorize the resulting edits into **recovery** and **non-recovery** edits. Recovery edits determine how the agent responds when execution fails or ceases to make progress, whereas non-recovery edits change how the agent monitors progress, selects external vision tools, and grounds decisions during ongoing interaction. For each behavior, we examine the corresponding harness change together with its effect across held-out trajectories. Due to space constraints, we illustrate each modification type with an example from either navigation or manipulation. The full set of examples and analyses is provided in Appendix D.

Table 3: Subtask-wise comparison of different harness optimization settings on the search and held-out sets. EMHO (Separate) refers to independent optimization for each subtask, whereas EMHO (Combined) optimizes a single shared harness using experience traces pooled across subtasks.

(a) Navigation

Split	Harness	Avg.	Base	Visual	Long
Search	Initial	54.4	60.0	50.0	53.3
	EMHO (Separate)	65.6	76.7	63.3	56.7
	EMHO (Combined)	61.1	73.3	63.3	46.7
	EMHO-Merge	67.8	80.0	60.0	63.3
Held-out	Initial	66.7	73.3	66.7	60.0
	EMHO (Separate)	75.6	86.7	76.7	63.3
	EMHO-Merge	75.6	93.3	70.0	63.3

(b) Manipulation

Split	Harness	Avg.	Base	Visual	Spatial
Search	Initial	25.9	16.7	44.4	16.7
	EMHO (Separate)	47.2	50.0	50.0	41.7
	EMHO (Combined)	34.7	41.7	33.3	29.2
	EMHO-Merge	39.8	37.5	44.4	37.5
Held-out	Initial	25.9	25.0	27.8	25.0
	EMHO (Separate)	40.7	41.7	38.9	41.7
	EMHO-Merge	33.8	29.2	38.9	33.3

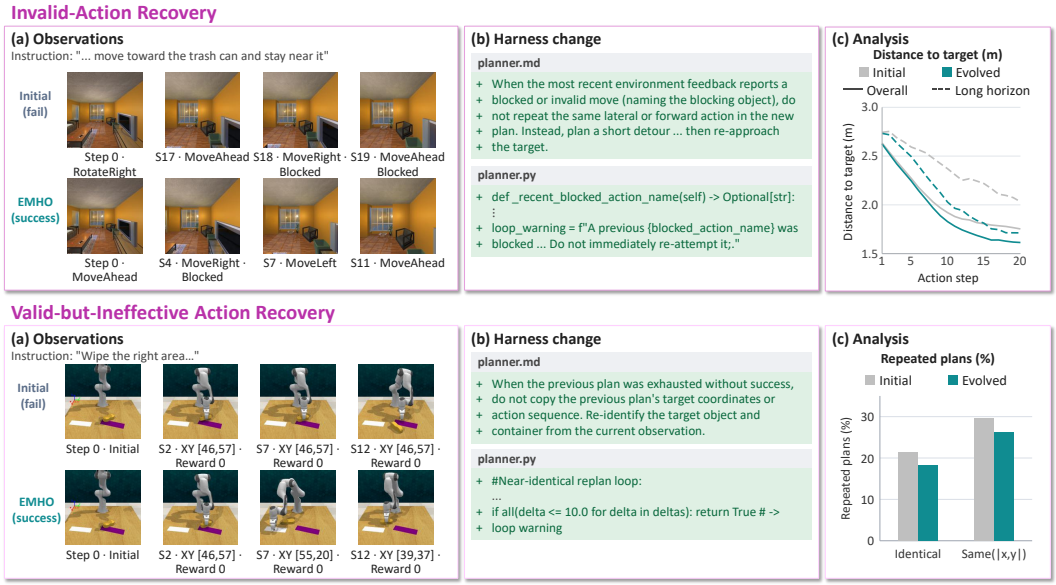


Figure 3: Qualitative analysis of recovery edits during harness optimization. (a) Representative observation frames from the initial harness and evolved EMHO trajectories. (b) Examples of the corresponding harness changes. (c) Analysis of their effects.

5.1 Recovery Edits

Invalid-Action Recovery. Embodied environments provide sparse execution feedback, such as blocked navigation movements or unsuccessful manipulation actions. The initial harness provides limited guidance on how to respond to these failures, so the agent may repeat unsuccessful actions. EMHO adds explicit recovery instructions and modifies the harness to retain failure feedback and carry it into the context for subsequent planning, as illustrated in Figure 3. In navigation, the evolved harness instructs the agent to avoid retrying a blocked movement and instead reorient or take a short detour. Consistent with these changes, evolved trajectories are more likely to continue reducing their distance to the target after encountering a blockage. Note that the distance to the target is computed post hoc for analysis. In manipulation, invalid actions are reflected in execution feedback, such as a path-generation failure. EMHO explicitly exposes this feedback during replanning and requires the next plan to differ from the failed attempt. Rather than retrying the same action, the planner can revise the target location or approach direction before attempting the manipulation again.

Valid-but-Ineffective Recovery. A different failure mode occurs when individual actions are valid but collectively make little progress. In navigation, the agent may repeatedly rotate or reverse direction while every action executes successfully, and in manipulation, it may reissue nearly identical plans or repeatedly attempt grasps at the same coordinates. Because no explicit invalid-action signal is available, the harness must infer from the recent trajectory that the current behavior is ineffective. EMHO introduces mechanisms that detect these patterns and require the next plan to change strategy.

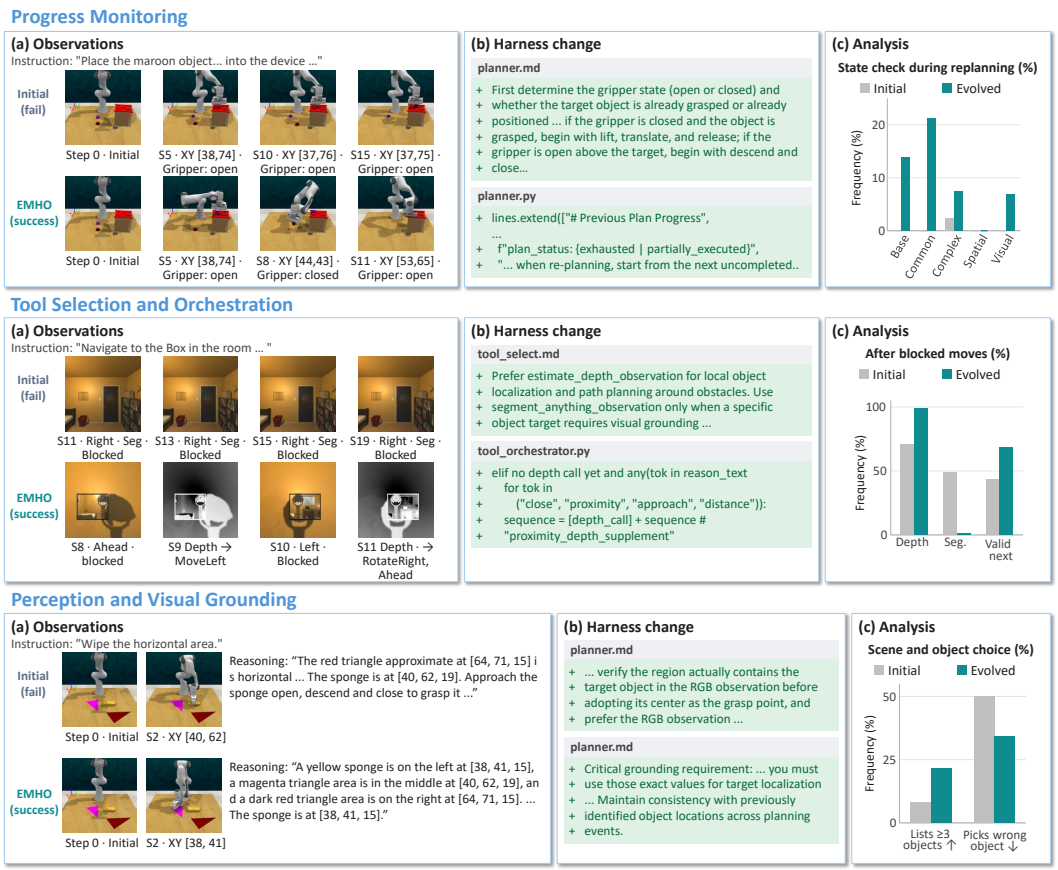


Figure 4: Qualitative analysis of non-recovery edits during harness optimization. (a) Representative observation frames from the initial harness and evolved EMHO trajectories. (b) Examples of the corresponding harness changes. (c) Analysis of their effects.

In navigation, sequences of at least five consecutive rotations become less frequent. In manipulation, identical replans decrease from approximately 21% to 18%, while repeated grasps at the same coordinates decrease from approximately 30% to 26%. These results suggest that EMHO helps the agent recognize ineffective behavior even when the individual actions are valid.

5.2 Non-Recovery Edits

Progress Monitoring. Beyond responding to failure, EMHO changes how the harness determines whether an ongoing trajectory is making meaningful progress. In navigation, the evolved harness compares successive observations and uses changes in the target's apparent position and scale, as well as changes in scene appearance, to assess whether recent motion is bringing the agent closer to the target. In manipulation, as shown in Figure 4, the evolved harness tracks execution phases, including approach, grasp, lift, translate, and release. During replanning, it uses the current gripper and object state to infer which phases have already been completed and which action should follow, rather than restarting the entire sequence. Consistent with this change, replanning traces more frequently identify the current execution phase, reaching approximately 21% in some subtasks.

Tool Selection and Orchestration. EMHO also changes how and when external vision tools are selected. The evolved harness selects tools based on the type of uncertainty relevant to the next decision. In navigation, depth estimation supports obstacle and proximity reasoning, while segmentation helps identify specific targets, and zoom provides finer visual details once the relevant region is identified. After blocked moves, the use of depth estimation rises from approximately 70% to nearly 100%, alongside an increase in the fraction of valid subsequent actions from approximately 44% to 68%. In manipulation, the harness learns to call depth estimation proactively for height-

sensitive interactions, such as stacking and container placement, where relative height and clearance are important for subsequent actions.

Perception and Visual Grounding. Finally, EMHO strengthens the connection between perceptual evidence and the actions that follow from it. In navigation, the evolved harness compares candidate targets using instruction-relevant attributes and uses the selected visual evidence to guide the subsequent motion decision. When the agent explicitly identifies a target in its reasoning, the arrival rate increases, and cases in which the agent passes the target and moves farther away decrease. In manipulation, the harness more explicitly distinguishes object identity from executable grasp coordinates and re-evaluates both from the current observation during planning. The fraction of scene descriptions listing at least three objects increases from approximately 8% to 22%, while incorrect object selection in initial plans decreases from 50% to 34% in spatial-relation tasks. These results indicate more detailed scene descriptions and fewer object-selection errors during planning.

5.3 Evolution Across Optimization Iterations

Recovery edits dominate early, while non-recovery edits emerge over time. Across the harness optimization process, the earliest proposals predominantly introduce recovery edits. Non-recovery edits are present from the start but tend to gain share only after the harness has acquired an explicit response to failure. This temporal pattern suggests that observable failure signals provide the most direct initial optimization target. In contrast, non-recovery edits require a more developed diagnosis of the ongoing trajectory, such as whether the embodied agent is making sufficient progress or whether its current plan should be adjusted before an explicit failure occurs. A detailed analysis of how the relative proportions of recovery and non-recovery edits change across optimization iterations is provided in Appendix D.3.

Rules become more detailed. Later generations do not simply replace the first recovery rule; instead, they refine it by introducing increasingly detailed conditions. A representative navigation chain progresses from "do not repeat the blocked action," to "rotate after several blocked directions," and eventually to selecting a response based on the blocked direction, target heading, and the distance required to clear the obstacle. Across runs, recovery rules become more detailed over successive generations, with additional conditions and clauses that specify when and how each response should be applied. This suggests that the optimizer is not merely accumulating advice, but gradually transforming generic reactions into more detailed and context-dependent policies.

Later iterations can exhibit overfitting. As optimization proceeds, the harness that performs best on the search split is not always the one that performs best on held-out tasks. In several runs, later iterations continue to improve or maintain search performance while held-out performance plateaus or declines. This divergence suggests an emerging overfitting pattern, where the harness becomes increasingly specialized to the trajectories observed during optimization. These results motivate the need to consider generalization when selecting among harness iterations.

6 Conclusion

We introduced EMHO, a framework for improving embodied agents by evolving the harness around a frozen underlying model from experience traces and sparse environmental feedback. Rather than restricting adaptation to reusable skills or recovery prompts, EMHO optimizes the broader interaction procedure, including planning, context construction, and tool use. Across navigation and manipulation, EMHO improves search-set performance, and the resulting harnesses also outperform the initial harness on held-out tasks, suggesting that harness evolution can help translate existing model capabilities into more successful embodied behavior. Our analysis shows that the evolved harness changes not only how agents recover from failures, but also how they monitor progress, gather perceptual evidence, and translate observations into subsequent actions. We further introduced EMHO-Merge, which uses episode-level gains and losses to refine a shared harness across subtasks and mitigate trade-offs during joint optimization. Together, these results show that harness optimization provides a promising direction for improving embodied-agent behavior without updating the underlying model.

AI USE STATEMENT

We used generative AI tools to aid and polish the writing and code development of this paper. We have not used generative AI tools for research ideation or execution, and generating synthetic datasets or proving mathematical claims are not applicable to this work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies methods for improving the harness of embodied agents through interaction experience. Our experiments do not involve human subjects or the collection of personally identifiable information. We evaluate our method in controlled benchmark environments using existing models, tasks, and evaluation protocols.

REPRODUCIBILITY STATEMENT

We provided the main methodological and experimental details relevant to our study in the main paper and appendices. These include descriptions of the proposed approach, benchmark and task setup, evaluation metrics, baselines, and additional experimental analyses.

REFERENCES

Anthropic. Introducing Claude Opus 5. `https://www.anthropic.com/news/claude-opus-5`, July 2026.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. SAM 3: Segment anything with concepts. In *International Conference on Learning Representations (ICLR)*, 2026.

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. *arXiv preprint arXiv:2608.16590*, 2026.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, 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*, 2023.

Ruofei Ju, Xinrui Wang, Xin Ding, Yifan Yang, Hao Wu, Shiqi Jiang, Qianxi Zhang, Hao Wen, Xiangyu Li, Weijun Wang, Kun Li, Yunxin Liu, Haipeng Dai, Wei Wang, and Ting Cao. EmbodiSkill: Skill-aware reflection for self-evolving embodied agents. *arXiv preprint arXiv:2605.10332*, 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. In *Proceedings of the 8th Conference on Robot Learning*, Proceedings of Machine Learning Research, 2025.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. In *Conference on Language Modeling (COLM)*, 2026. arXiv:2603.28052.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, et al. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026.

Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. OK-Robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi Fan, and Guanzhi Wang. ASPIRE: Agentic skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

OpenAI. GPT-5.6 Sol. `https://developers.openai.com/api/docs/models/gpt-5.6-sol`, July 2026.

Sungho Park, Wonjoong Kim, Rongyuan Tan, Jue Zhang, Wook-Shin Han, Pengfei Gao, Chanyoung Park, Yongqiang Yao, Rao Fu, Elsie Nallipogu, et al. AutoSaddler: Automatic harness optimization with durable updates from agent execution traces. *arXiv preprint arXiv:2608.23041*, 2026.

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. In *Proceedings of the 9th Conference on Robot Learning*, Proceedings of Machine Learning Research, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, February 2026a.

Qwen Team. Qwen3.8-27B-FP8 model card. `https://huggingface.co/Qwen/Qwen3.8-27B-FP8`, August 2026b.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70576–70631, 2025.

Yunhao Yang, Neel P. Bhatt, Kevin Wang, Samuel Tetteh, Zhangyang Wang, and Ufuk Topcu. VASO: Formally verifiable self-evolving skills for physical AI agents. *arXiv preprint arXiv:2606.05395*, 2026.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026.

Xueyang Zhou, Zijia Wang, Qianjiang Li, Yibo Hu, Guiyao Tie, Li Wan, Yidan Liu, Pan Zhou, Lichao Sun, and Yongchao Chen. Enabling extensible embodied capabilities with tools. *arXiv preprint arXiv:2605.26637*, 2026.

APPENDIX

A DETAILS OF HARNESS STRUCTURE

Table 4: Overview of the four main functional components of the manipulation harness.

Component	Main files	Role in the harness

Planning	`SKILL.md`
	`system.md`
	`final_planner.md`
	`agent.py`
	`planner.py`
	`model_client.py`
		Defines the manipulation-planner contract and implements the main decision loop, including model-input construction, plan generation, action execution, and replanning from environment feedback.

Vision and tool use	`tool_orchestrator.py`
	`vision_policy.py`
	`tool_select.md`
		Controls when and how visual evidence tools, such as zoom, segmentation, and depth, are invoked and determines how their outputs are incorporated into subsequent planning.

Memory and context	`module_selector.py`
	`memory.py`
	`memory_render.md`
	`safe_context.md`
		Maintains bounded episode history and selects the model-facing context, including previous actions, observations, feedback, and other safe contextual information used during replanning.

Validation	`parser.py`
	`repair.py`
	`repair.md`
		Validates generated plans and actions, corrects malformed or non-executable outputs, and ensures that the resulting plan is executable.

The harness consists of the instructions and runtime components that control planning, context construction, tool use, action generation, validation, and recovery. During optimization, EMHO modifies these components based on recurring failure patterns observed in experience traces.

EMHO Harness Optimizer Instruction Example This example illustrates the instruction provided to the harness optimizer for the Navigation Base subtask. It shows the general structure of the optimizer instruction, including the available evidence, editable harness components, regression-aware constraints, and required workflow. File and path names are shown in simplified placeholder form for presentation.

EMHO Harness optimizer instruction example

```
Runtime workspace paths:

- read-only proposer_view root:  {{PROPOSER_VIEW_ABSOLUTE_PATH}}
- editable candidate root:  {{CANDIDATE_ABSOLUTE_PATH}}

You are optimizing the EmbodiedBench navigation base subtask harness to improve task success.

Use the visible evidence in {{file_path}}, and the current harness snapshot, to find
harness changes that are likely to improve navigation performance.

Make focused, evidence-supported changes to the allowed candidate harness files.
The changes should improve how the agent selects episode memory or vision
support, formats outputs, repairs mistakes, or plans actions.

You must make at least one substantive change to an allowed candidate harness
file.  An unchanged copy of the parent harness is rejected as a failed proposal.
Keep the patch practical and coherent.  Edit the files needed for the improvement.
The goal is better task success, not producing the smallest possible diff.

Changes must be general and non-episode-specific.  Do not use hidden simulator
state, shortest paths, heldout data, or hard-coded episode solutions.  Do not force
narrow tool/action recipes; improve the harness behavior so the model can make better
decisions.  Do not stop after describing a plan or saying that you will read files.
Make the edits, then write {{file_path}} with cited evidence
```

```
and changed files.

Read only:

- {{PROPOSER_VIEW_ABSOLUTE_PATH}}

Edit only these candidate workspace harness files:  {{file_paths}}

Make a coherent set of changes to the allowed harness files when needed.
Prefer the coordinated update that best supports task success.

Regression-aware editing:

- Review candidate scores, parent candidates, changed harness paths, score deltas,
  and harness_diff_from_parent.  If a lower-scoring child added or strengthened a rule,
  tool-selection policy, prompt section, or parser behavior, do not repeat that
  structure unless the evidence clearly explains why your version avoids the regression.
- Before adding a new rule, ask whether an existing rule should be removed,
  shortened, weakened, or reverted.  A valid candidate may improve by deleting or
  simplifying text.

Rules:

- Do not use heldout data, hidden simulator state, coordinates, rewards, shortest
  paths, or episode-specific hacks.
- Do not edit evaluator, dataset splits, datasets, runner scripts, or action semantics.
- Preserve CandidateHarness API.
- Cite search evidence from search_evidence.jsonl in rationale using event_id,
  episode_ref, env_step, or record_type where available.

Workflow:

1.  Read {{file_path}} and any relevant current prompts.
2.  If {{file_path}} exists, compare it only as optional context because
  it differs from source_harness.
3.  Check frontier.json for parent-success episodes lost by later candidates and for
  harness changes correlated with lower scores.
4.  Review the visible search evidence and regression evidence for harness changes
  likely to improve task success.
5.  Edit the allowed files needed for a coherent improvement.
6.  Write {{file_path}} with cited evidence, changed files, and why
  the patch should improve task success.
7.  Run {{file_path}} only for syntax check.
```

EMHO-Merge Harness Optimizer Instruction Example This example illustrates the instruction provided to the EMHO-Merge harness optimizer for the Navigation Base subtask. File and path names are shown in simplified placeholder form for presentation.

EMHO-Merge Harness optimizer instruction example

```
Runtime workspace paths:

- read-only proposer_view root:  {{PROPOSER_VIEW_ABSOLUTE_PATH}}
- editable candidate root:  {{CANDIDATE_ABSOLUTE_PATH}}

You are optimizing the EmbodiedBench navigation base subtask harness to improve
task success.

Use the visible evidence in {{file_path}}, and the current harness snapshot, to find
harness changes that are likely to improve navigation performance.

Make focused, evidence-supported changes to the allowed candidate harness files.
The changes should improve how the agent selects episode memory or vision
support, formats outputs, repairs mistakes, or plans actions.

Follow the proposal mode supplied by the optimizer.  If the mode is `new_mechanism`,
propose a genuinely new behavior.  If the mode is `repair_or_new`, investigate the
previous change using the required paired trajectories before deciding between
`repair` and `new`.  Record the selected mode and its supporting evidence in
`experiment.json`.

- `new_mechanism`:  test a genuinely different behavior, not a reworded version of
```

```
any mechanism in `forbidden_new_mechanism_ids`.

- `repair_or_new`: the latest evaluated change produced both gained and lost
episodes. This allows one repair investigation but does not establish causality.
Inspect both parent and candidate trajectories for at least one gained and one
lost episode from any active subset. Verify that the changed behavior appears and
identify a visible condition that separates benefit from harm. Choose `repair`
only when the evidence supports that condition; otherwise choose `new` and
explain `why_not_repair`.

- A repair must reuse the original `mechanism_id` and specify `repair_of`. Each
mechanism receives only one evaluated repair attempt. Mechanical validation
retries do not count.

- The writable candidate starts from the current best parent, which may differ from
the previous candidate. Interpret `previous_change.diff` relative to its actual
parent. Rejected changes are not installed. A repair may reconstruct only the
useful behavior with a narrower trigger.

- Treat small differences as uncertain. Do not invent causal explanations or assume
additional evaluation or parallel proposal budgets.

- For both `new` and `repair`, inspect and cite the parent and candidate event IDs
for every category in `required_previous_pair_categories`. Use
`comparison --category gained` and `comparison --category lost` to find eligible
IDs, then query each trajectory before writing the contract. Baseline-only
examples do not satisfy this requirement.

- Novelty is checked against behavior descriptions and patch content, including
`discouraged_behavior_families` and `previous_patch_signatures`. Renamed rotation,
blocked-move, detour, lateral-step, or reposition warnings still belong to the
same recovery family. They cannot be submitted as new mechanisms, although one
evidence-supported repair is allowed.

- Before adding an intervention, inspect a successful trajectory where the proposed
trigger also occurs. Compare the visible feedback and subsequent progress. If
harmful and useful activations cannot be distinguished, narrow or abandon the
intervention. Raw action counts alone do not prove the agent is stuck. Record the
counterexample and remaining uncertainty in `regression_risk`.

For a repair also supply `repair_of` and `separating_evidence` (including the observed
activation and the visible distinction between gained/lost situations). Both proposal
modes must cite BOTH parent and candidate event IDs for each required gained/lost
category; the subtask's net score sign is irrelevant. For a new mechanism when a
repair was offered, supply `why_not_repair`. Use a stable `mechanism_id` across
repairs; renaming the same warning is not a new mechanism. This metadata is
optimizer-only and must not enter runtime prompts.

Workflow:

1. Read {{file_path}}. Read the relevant hunks of `previous_change.diff` and the
hypothesis/change sections of {{file_path}} when available.

2. Briefly retain the observed failure/success differences across subsets as you go.

3. Now select one mechanism supported by that review. The completed review queue
includes required previous gained/lost pairs. Use `comparison` to identify their
category and cite BOTH event IDs per required category.

4. Explain both prospective gains and regressions. State which subsets the mechanism
affects, which evidence contradicts it, and what the previous edit taught you. A
lower score alone does not prove an idea failed: check implementation activation
and run variability. Do not repeat a prior unsuccessful change without a specific
correction or new evidence. Keep overall search success as the objective; do not
require every subset to improve in every sample.

5. Read then write the draft `experiment.json` before further source research. The
draft is editable until the first harness edit; writing it does not waive novelty
or final validation. Once you choose the hypothesis, read only the details needed
to implement it. Use {{file_path}} to locate exact source ranges.

6. Make one coherent edit in the existing allowlisted files.

7. Record what was implemented, checks performed, and unresolved assumptions; do
not claim an unrun experiment succeeded.
```

B Quantitative Results

This appendix presents full quantitative results for all navigation and manipulation subtasks. For each subtask, the plots show the best success rate attained separately on the search and held-out splits up to each optimization iteration. In each EMHO run, the same model serves as both the embodied agent and the harness optimizer. Baseline results for other underlying models are taken from Embodied Tool Protocol (Zhou et al., 2026) and EmbodiedBench (Yang et al., 2025). Throughout optimization, candidate generation and parent updates are guided by search-set feedback and harness history; held-out trajectories and scores are not provided to the optimizer. The split-wise maxima characterize the best observed performance among the resulting candidates, complementing our analysis of how experience-guided revisions change the harness and agent behavior. Since the search and held-out maxima may occur at different iterations, we interpret the held-out results as retrospective summaries of candidate performance, rather than estimates of the generalization performance of a single search-selected harness.

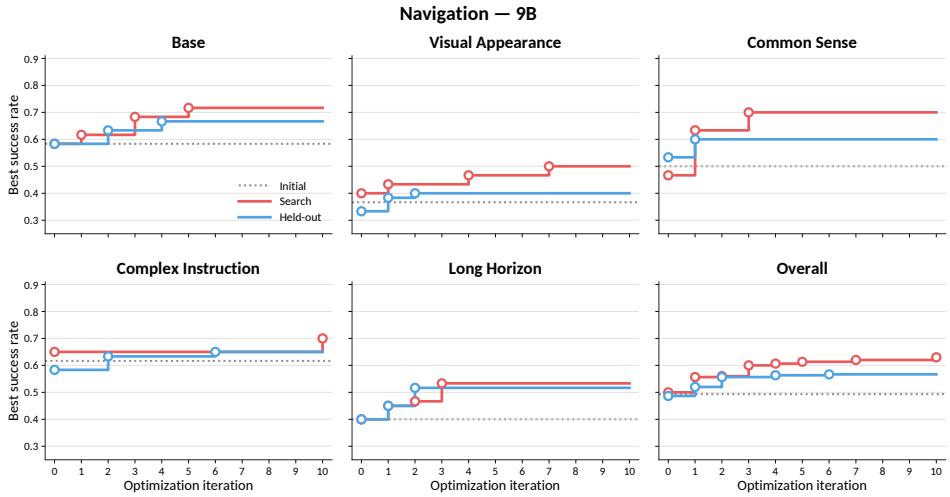


Figure 5: Quantitative results on the navigation task with the 9B model across optimization iterations.

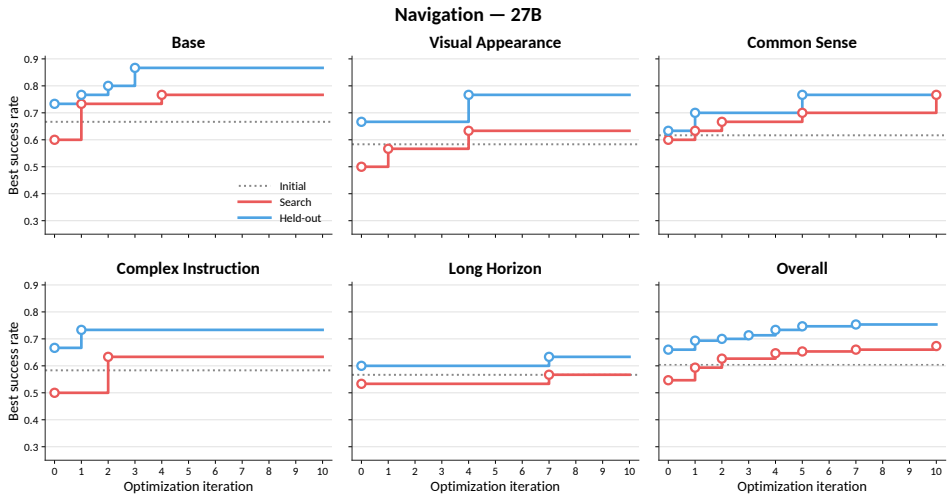


Figure 6: Quantitative results on the navigation task with the 27B model across optimization iterations.

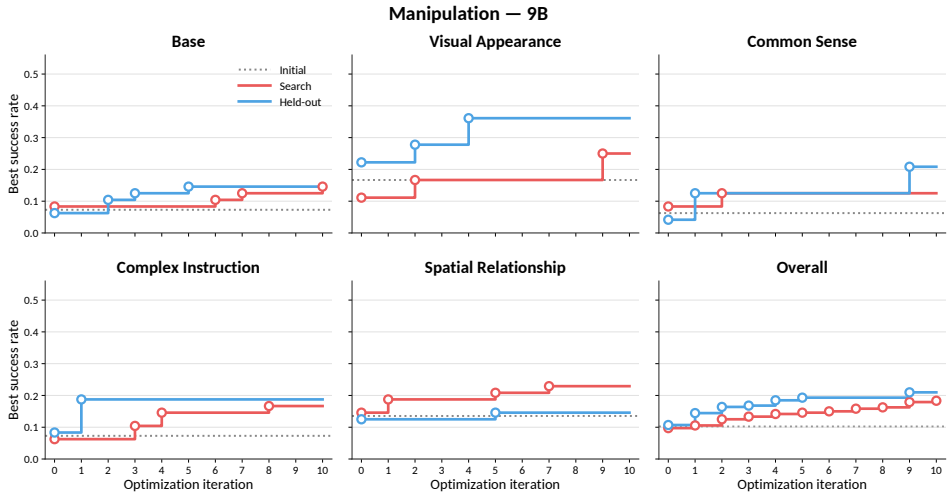


Figure 7: Quantitative results on the manipulation task with the 9B model across optimization iterations.

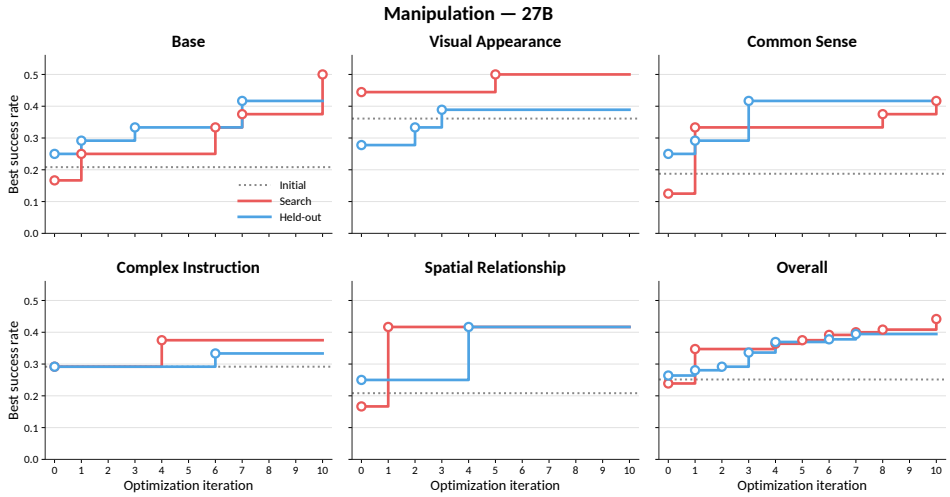


Figure 8: Quantitative results on the manipulation task with the 27B model across optimization iterations.

C Ablation Study

EMHO provides the harness optimizer with two forms of historical context: episode-level experience traces, $\mathcal{E}^{(0:i)}$, and the history of previous harness revisions and their evaluation outcomes, $\mathbf{H}_{\text{hist}}^{(0:i)}$. We ablate these two sources of information on the Navigation Base subtask to examine their respective contributions. As shown in Table 5, Full EMHO receives both experience traces and harness history. The 'without Experience Traces' variant removes episode-level trajectories while retaining previous harness revisions and their scores; the 'without Harness History' variant retains the trajectories but hides prior harness modifications and candidate history. Finally, 'Score Only' removes both forms of historical context and provides only the current parent harness and its evaluation score. The comparison isolates the complementary roles of the two information sources. Experience traces provide behavioral evidence about *why* an evaluated harness succeeds or fails, while harness history informs the optimizer about *what* modifications have already been attempted and how they affected performance. Comparing the full setting against the two single-source ablations therefore measures whether either source alone is sufficient, while the score-only setting tests whether scalar evaluation feedback can guide harness optimization without trajectory-level or revision-level context.

Table 5: Ablation of the optimization context on the Navigation Base subtask search set. All variants use the same model and evaluation procedure, while differing only in whether experience traces and harness-revision history are provided. Performance is reported on the search set.

Method	Traces	History	Search
Full EMHO	✓	✓	**76.7**
w/o Experience Traces	–	✓	70.0
w/o Harness History	✓	–	66.7
Score Only	–	–	63.3

Full EMHO achieves the strongest performance on the search set, while removing either source of context degrades performance. This suggests that the two histories provide complementary information: traces help diagnose recurring behavioral failures, whereas revision history helps the optimizer avoid revisiting ineffective modifications and refine previously useful ones.

D Qualitative Results

Aggregate task success reveals whether harness optimization works, but not *what the optimizer changes*. We therefore compare the initial and evolved harnesses on held-out episodes. We organize the resulting edits into two groups. Recovery edits govern how the agent responds once a failure is observed or inferred, whereas non-recovery edits operate independently of failure signals, shaping how the agent processes observations, reasons about the current state, and selects actions during ongoing interaction. For each category, we first describe the harness change, then examine a representative episode, and finally measure whether the intended behavior appears across held-out trajectories.

D.1 Recovery Edits

EmbodiedBench provides only lightweight feedback about the outcome of interaction. In navigation, the environment reports action-level outcomes such as valid, invalid, or blocked. In manipulation, it provides action-validity feedback, a scalar reward in $-1, 0, 1$, and whether the agent is currently holding an object. These signals expose limited information about execution outcomes, but generally do not identify the underlying cause of failure. We divide recovery edits into **invalid-action recovery**, which responds to actions reported as invalid or blocked, and **valid-but-ineffective action recovery**, which responds when actions execute validly but repeated attempts or unchanged observations show that the task is not progressing.

Invalid-Action Recovery.

Navigation: turning blocked-action feedback into a detour. When a navigation action is blocked, the initial harness often remains trapped around the same obstacle, proposing alternative actions without actually escaping the local constraint. EMHO evolves the harness to retain and act on this failure information during subsequent replanning. The resulting edits introduce explicit recovery rules that discourage movement along recently blocked directions and instead favor reorientation, backtracking, or lateral detours before re-approaching the target. They also add mechanisms that identify which action and direction were blocked, track multiple recent failures, and summarize these constraints for the planner, ensuring that this information remains available across replanning steps rather than being lost after the immediate feedback. In some cases, the blocked-state information is further carried through memory and tool-derived evidence, allowing later decisions to remain conditioned on what has already failed. Together, these changes turn blocked feedback from a transient execution message into a persistent constraint on subsequent planning. We assess the effect of this recovery behavior by tracking the agent's distance to the target over the course of each episode. This metric is used only for post-hoc analysis and is not provided to the optimizer during harness optimization. With the initial harness, trajectories often make limited progress toward the target and remain at a similar distance for multiple steps after encountering blockage. In contrast, evolved harnesses tend to continue reducing the distance to the target, with the clearest difference observed in long-horizon episodes.

Manipulation: conditioning replanning on reward and state. Manipulation provides limited feedback after plan execution: a reward of -1 indicates an invalid action, 0 indicates valid execution without

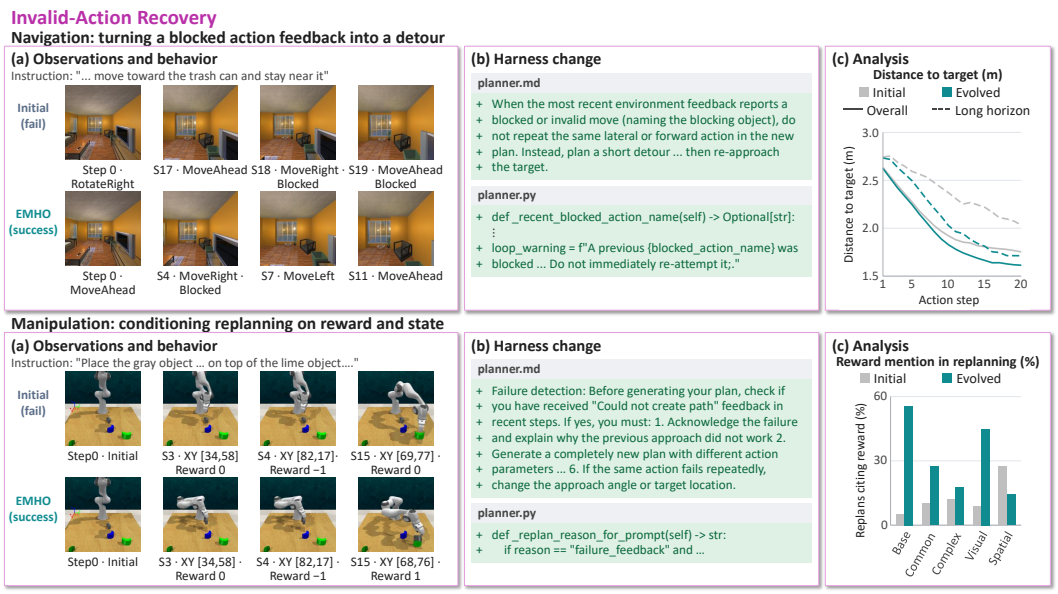


Figure 9: Qualitative analysis for invalid recovery in navigation and manipulation.

task completion, and 1 indicates task success, together with a state signal indicating whether the object is currently held. EMHO evolves the harness to explicitly incorporate this feedback into replanning. In particular, when an action receives invalid feedback, the planner is instructed not to simply repeat the same attempt, but to revise the next plan by changing relevant action parameters, such as the target location or approach direction. Related edits also expose the most recent reward and execution outcome to the planner, so that the revised plan is conditioned on what failed rather than generated independently of the previous attempt. To quantify this behavior, we measure how often replanning traces explicitly refer to the observed reward feedback. This fraction increases most clearly for the base, common-sense, and visual task families, rising from approximately 5% to 55%, 10% to 28%, and 10% to 45%, respectively.

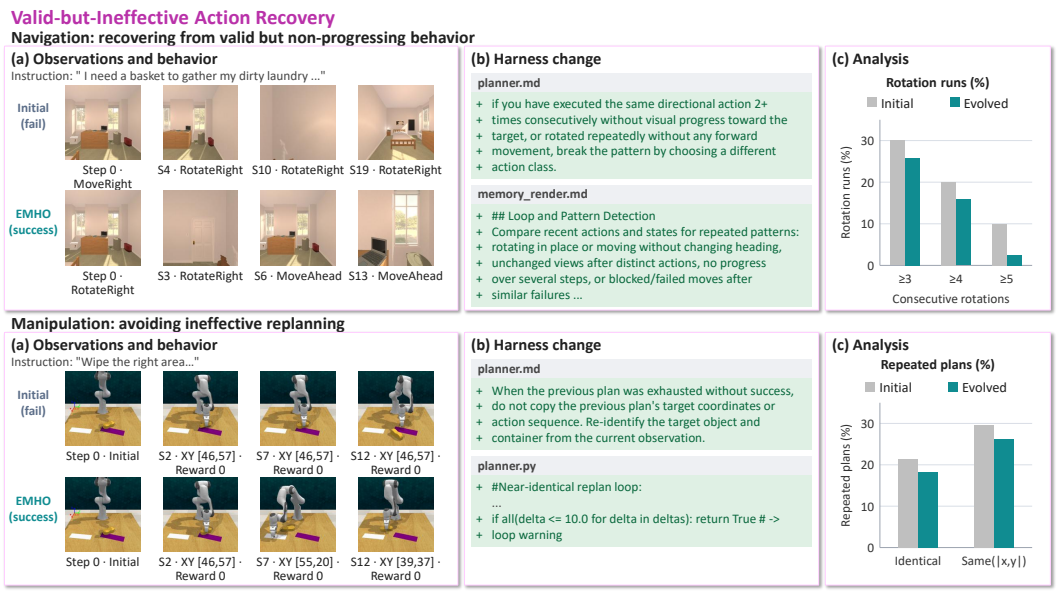


Figure 10: Qualitative analysis for valid but ineffective action recovery in navigation and manipulation.

Valid-but-Ineffective Action Recovery.

Navigation: recovering from valid but non-progressing behavior. In navigation, the agent can continue issuing valid actions while repeatedly rotating, reversing direction, or otherwise failing to make progress toward the target. EMHO introduces mechanisms that recognize such behavior from the recent interaction history and make it visible during replanning. The evolved harness summarizes repeated or reversing actions, compares recent observations to determine whether the agent is making meaningful progress, and encourages a change in action strategy when valid actions repeatedly leave the agent in a similar state. These changes help the planner distinguish successful execution from actual task progress. We count runs of at least three, four, and five consecutive rotations in held-out trajectories. The evolved harness reduces repetition at all three thresholds, with runs of at least five rotations decreasing from approximately 10% to 3%.

Manipulation: avoiding ineffective replanning. A similar failure mode arises in manipulation when an executable plan completes without solving the task, yet replanning produces essentially the same plan or returns to the same grasp location. EMHO modifies the harness to retain what was attempted and require the subsequent plan to make a meaningful change, such as revising the grasp point, approach height, descent depth, or target estimate. Additional mechanisms compare consecutive plans for near-duplicate behavior and prompt the agent to re-localize the relevant object when replanning would otherwise repeat an ineffective attempt. We compare consecutive replans for identical action sequences and reuse of the same grasp coordinates. Both forms of repetition become less frequent with the evolved harness.

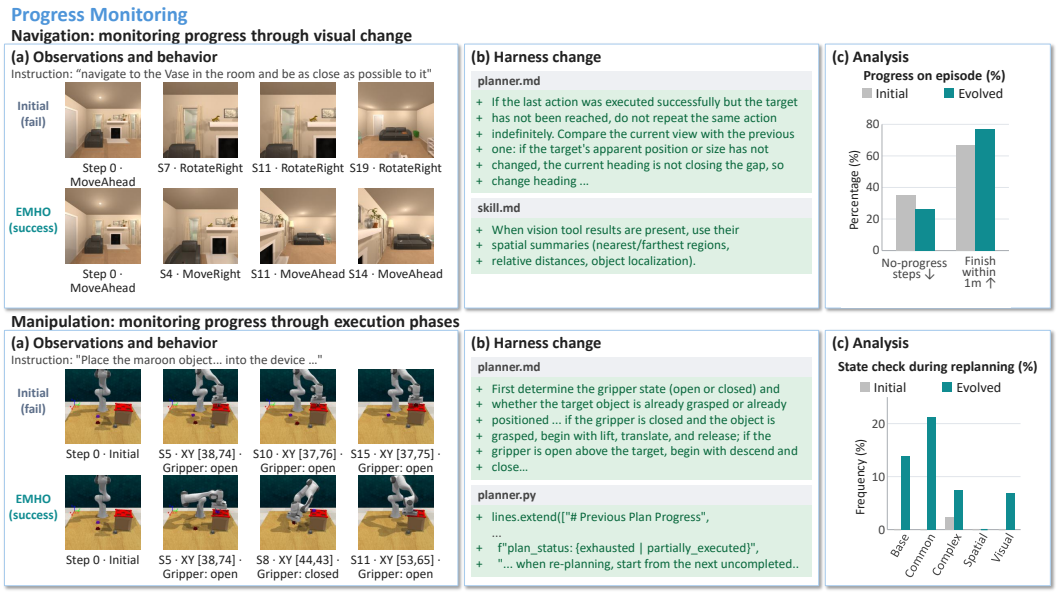


Figure 11: Qualitative analysis for progress monitoring in navigation and manipulation.

D.2 Non-Recovery Edits

Progress Monitoring.

Navigation: monitoring progress through visual change. EMHO evolves the navigation harness to explicitly monitor whether recent actions are producing observable progress toward the task. The harness compares consecutive observations after successful actions and uses changes in the target's apparent position, size, or surrounding scene to determine whether the current motion is advancing the agent or leaving it effectively stationary. When little visual progress is observed, the planner is encouraged to change its strategy—for example, by rotating, translating laterally, or moving to a new viewpoint—rather than continuing along the same heading. Related edits shorten cached multi-action plans so that the agent can re-observe the scene and reassess progress before committing to additional motion. Across held-out visual-appearance episodes, no-progress steps decrease from 35% to 26%, while the fraction of episodes ending within one meter of the target increases from 67% to 77%.

These changes indicate that the evolved harness more consistently uses observable state changes to determine whether its current behavior is making meaningful progress.

Manipulation: monitoring progress through execution phases. EMHO evolves the manipulation harness to organize execution into an explicit five-stage process: approach with the gripper open, descend and grasp, lift, translate, and release. The harness then introduces stage-specific behavioral rules that determine which phase has been completed, whether the expected state transition occurred, and what action should follow. Before replanning, the agent examines the current observation together with the gripper and object state to determine whether the object has been grasped, lifted, moved, or already positioned, and either continues from the next unfinished phase or returns to an earlier phase when the expected transition has not occurred. In this way, manipulation progress is represented explicitly as movement through a structured sequence of execution stages rather than as a collection of independent plans. We count replanning traces that explicitly name the current execution phase. The evolved share reaches approximately 14% in base, 21% in common-sense, 8% in complex, and 7% in visual tasks, compared with near-zero initial rates in those families.

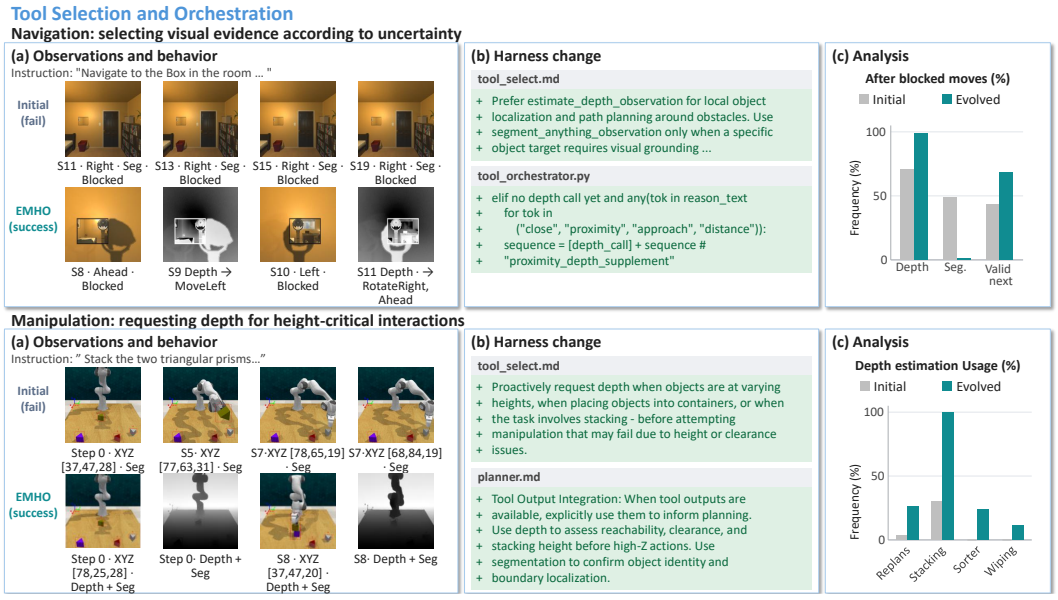


Figure 12: Qualitative analysis for tool usage in navigation and manipulation.

Tool Selection and Orchestration.

Navigation: selecting visual evidence according to uncertainty. EMHO evolves the navigation harness to make tool use conditional on the information required for the next decision rather than invoking the same visual modules at every planning step. The resulting edits distinguish among appearance uncertainty, spatial or proximity uncertainty, and cases in which the current observation already provides sufficient evidence. The harness introduces explicit gating rules that favor no additional tool after a successful action when the scene and target direction remain clear, while retaining episode memory when recent blocked or failed actions make previous experience relevant. Tool-specific rules further route depth estimation toward obstacle and local path reasoning, segmentation toward identifying a particular target, and zoom toward resolving fine visual details once the relevant region is known. Complementary orchestration rules also invoke depth estimation when the planner expresses uncertainty about proximity or layout. We analyze tool calls immediately after blocked moves. Depth-estimation use rises from approximately 70% to nearly 100%, while segmentation use falls from approximately 50% to near zero. The share of valid next actions rises from approximately 44% to 68%.

Manipulation: requesting depth for height-critical interactions. EMHO introduces an instruction to request depth proactively when object heights differ, the task involves stacking, or placement into a container requires clearance reasoning. In the illustrated stacking episode, the evolved harness requests depth together with segmentation both at the initial perception step and again

during replanning, whereas the initial harness relies on segmentation alone. We further observe depth-estimation usage across task types and observe the largest increase for stacking tasks, where estimating the relative positions and heights of the object and the robot arm is particularly important for successful placement.

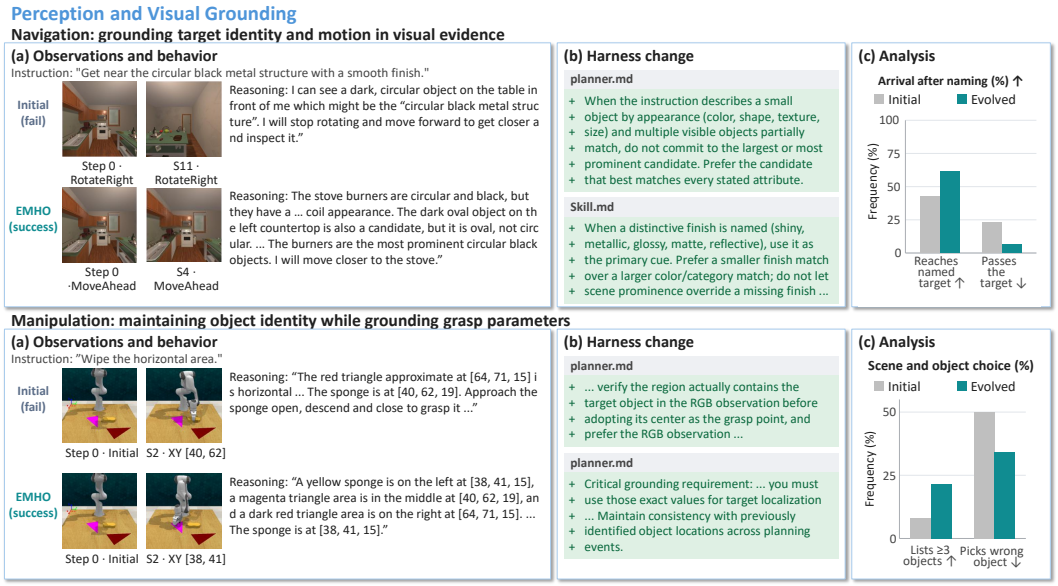
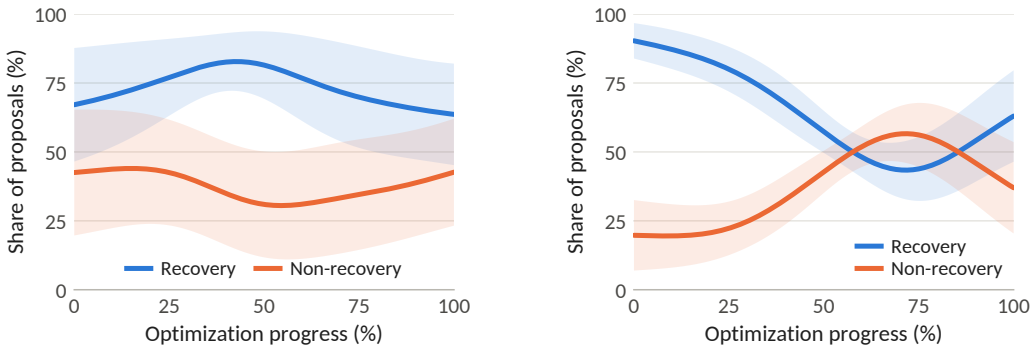


Figure 13: Qualitative analysis for perception and grounding in navigation and manipulation.

Perception and Grounding.

Navigation: grounding target identity and motion in visual evidence. EMHO evolves the navigation harness to connect perceptual evidence more explicitly to both target selection and the subsequent motion decision. When several visible objects partially match an instruction, the planner is encouraged to compare the attributes specified by the task—such as color, material, shape, or finish—rather than choosing the most visually salient candidate. Other edits propagate the target phrase from the instruction directly into segmentation, allowing the visual query itself to reflect the object description, while the planner is instructed to treat segmentation, zoom, and the original RGB image as complementary sources of evidence. This also allows the harness to recover from imperfect tool outputs. When segmentation produces an empty or unreliable mask, the planner can retain a visually supported target hypothesis from the RGB observation rather than interpreting detector failure as evidence that the object is absent. The resulting trajectories exhibit clearer evidence-to-decision chains: visual tools help resolve target attributes, the planner compares candidate objects using those attributes, and the selected evidence is then carried forward into the navigation decision. To quantify this behavior, we measure the conditional arrival rate after the agent has explicitly identified a target in its reasoning. This rate increases from 43% to 62%. At the same time, the fraction of such trajectories in which the agent passes the target and subsequently moves farther away decreases from 23% to 7%. These results indicate improved navigation after a target hypothesis has been formed, while not establishing a general improvement in target-identification accuracy.

Manipulation: maintaining object identity while grounding grasp parameters. EMHO evolves manipulation grounding around the distinction between identifying the intended object and choosing an executable grasp location. The harness requires relative descriptions such as "the right object" to be resolved against the current image, treats segmentation boxes and cropped regions as approximate perceptual cues rather than directly executable coordinates, and retains the RGB observation as the primary evidence for determining the final grasp pose. When replanning after an unsuccessful attempt, the agent can combine the current object list, newly requested visual evidence, and the history of previous grasp locations to re-estimate where the intended object is and modify the corresponding action parameters. In this way, grounding is treated as an ongoing process that can be updated from new observations rather than fixing the target location from a single initial detection. We inspect the planner's scene description and first object choice. Descriptions naming at least three objects with



(a) Navigation 27B

(b) Manipulation 27B

Figure 14: Evolution of recovery and non-recovery edits across the harness optimization process. The relative composition of proposed edits changes differently across navigation and manipulation.

coordinates rise from approximately 8% to 22%, and first plans choosing the wrong object fall from 50% to 34% across evolved harnesses. These measurements support clearer scene representation and fewer relational object-choice errors, but do not establish fine-grained grasp-coordinate accuracy.

D.3 Evolution Across Optimization Iterations

Recovery edits dominate early, while non-recovery edits emerge over time. Figure 14 shows how the relative share of recovery and non-recovery edits changes over the course of optimization. In both domains, recovery edits are more prominent during the early iterations, suggesting that explicit or easily identifiable execution failures provide a natural initial target for harness refinement. However, the subsequent evolution differs across tasks. In navigation, recovery edits remain dominant throughout optimization; the share of non-recovery edits dips in the middle of optimization and recovers toward the end, narrowing the gap between the two categories. In manipulation, by contrast, the share of recovery edits decreases substantially as optimization proceeds, while non-recovery edits become increasingly prevalent during the middle and later iterations before the two categories converge.

E Detailed Results of Harness Modifications

The examples below illustrate representative changes discovered for navigation and manipulation. For readability, some examples are shortened by omitting less relevant details.

Navigation. Example of the evolved harness in the navigation task.

harness/prompts/final_planner.md

```
  Use the current visual observation, valid navigation actions, action history,
  environment feedback, selected context, and any tool outputs to choose a
- short executable plan.
+ short executable plan.  Incorporate lessons from recent feedback, especially when
+ earlier actions produced blocked or failed outcomes.
+
  Do not choose an action from target visibility alone.  If the target or relevant
  object is visible, consider its apparent direction, distance, alignment,
  obstacles, and recent feedback before deciding whether to move, rotate, tilt, or
- reposition.
+ reposition.
+
+
  When action history shows prior errors or blocked attempts with memory available, factor that
  into
+
  your plan to avoid known failures.  When action history shows prior successful steps, build on
  them
+ rather than replanning unnecessarily.
+
+ Minimize actions needed.  Prefer shorter, direct paths unless intermediate actions are required
+ to safely access the target.
+
+ When planning new actions after successful steps, prioritize current observation over memory.
```

```
+
+Return only a JSON object with:
+
 - "executable_plan":  a list of action objects, each with "action_id" (integer) and "action_name"
  (string)
+- Optional:  "visual_state_description", "reasoning_and_reflection", "language_plan"
+
+The planner will execute at most {max_plan_actions} actions.
```

harness/planner.py

```
      return [action_id], json.dumps(reasoning)

     def _invalid_action_response(self, error:  str) -> Tuple[Any, str]:
+     if self.task_type == "navigation":
+      valid_actions = ", ".join(self.actions) if self.actions else "none"
+      reasoning = json.dumps(
+       {
+        "visual_state_description":  "Qwen Code returned an invalid action.",
+        "reasoning_and_reflection":  (
+         f"{error} The previous action was not executed.  "
+         f"Choose a different valid navigation action from:  {valid_actions}."
+        ),
+
         "language_plan":  "No valid action was produced; select an alternative navigation
         action.",
+        "executable_plan":  [],
+       }
+      )
+      return -1, reasoning
       reasoning = json.dumps(
        {
         "visual_state_description":  "Qwen Code returned an invalid action.",
```

harness/prompts/repair.md

```
  The previous solver output did not match the required schema.  Preserve the
  current task context, valid action ids, and valid action names.  Make the smallest
  repair that restores the expected JSON shape.
+
+Common failure patterns to avoid:
+1.  Wrapping output in markdown code blocks - unwrap to plain JSON
+2.  Adding conversational text before/after JSON - output only the JSON object
+3.  Missing executable_plan key - ensure it exists as an array
+4.  Invalid action_id values - use only valid action ids from the available set
+5.  Plan length exceeding max_plan_actions - output at most the configured maximum
+
 6.  Non-dict items in executable_plan - each item must be an object with action_id and
  action_name
+7.  Missing action_id - every plan item must include an integer action_id
+
+Output valid JSON only.  Do not include markdown formatting or conversational text.
+
  Ensure executable_plan is a non-empty list of objects, each with valid action_id and action_name.
```

harness/prompts/tool_select.md

```
  Selection guidance:
  - Inspect the current observation image and the previous actions/environment feedback.
  - Decide which optional support would help the next task-solving step:  episode memory or vision
  evidence.
-- Use memory when current-episode history or feedback is needed to avoid repeating mistakes.
+
 - Use memory when current-episode history or feedback is needed to avoid repeating mistakes,
  especially when
+recent actions produced blocked, failed, or unexpected feedback.
  - Select the available evidence source with the greatest expected marginal value for the next
  action.
-
 - Consider no-tool as a valid choice.  Prefer new, decision-relevant information over repeated or
  redundant
-evidence.
+
 - Consider no-tool as a valid choice.  Prefer no-tool when current observation and feedback
  clearly indicate
+the next action-this avoids unnecessary context overhead when confidence is high.
+- Prefer new, decision-relevant information over repeated or redundant evidence.
+
 - Consult episode memory when past contexts suggest specific hazards or when action outcomes
  appeared ambiguous.
+
 - Only select vision tools when they provide new spatial or structural information not visible in
  current
+observation and not already represented in the context.

  Final planner will execute at most {max_plan_actions} actions.
+
+
 When tool_use == none (no context selected), the final planner has observe-only information and
  must
+complete navigation using observations alone.
```

harness/SKILL.md

```
  This is an embodied navigation task.
```

```
  Use only the active task instruction, current observation images, valid action
- list, visible action history and feedback, context materialized by the harness,
+ space, visible action history and feedback, context materialized by the harness,
  and permitted vision-tool results.  Never use hidden simulator state, benchmark
- source code, shortest paths, target coordinates, reward labels, or final success
- labels.
+ source code, maps, shortest paths, target coordinates, object poses, reward
+ labels, or final success labels.

  For a selector request, return only the requested module-selection JSON. Include
  vision tool requests only in the selector field required by its schema.
```

Manipulation. Example of the evolved harness in the manipulation task.

harness/prompts/final_planner.md

```
  `"[[X, Y, Z, Roll, Pitch, Yaw, Gripper]]"`.
  Use the original EmbodiedBench manipulation rhythm when useful:  approach open,
  descend and close, lift closed, move/place, release open.
+ When re-planning after a previous plan exhausted without success, do not simply
+ re-emit the same approach/descent/lift sequence.  First compare the current
+ visual state to the previous plan's intent:  if the gripper is already above or
+ near the target, or the target is already grasped, start the new plan from the
+ next phase (e.g., lift, translate, release) instead of repeating the phase
+ that already completed.
  Every X/Y/Z value must be in [0, 100], every rotation value must be in [0, 120],
  and Gripper must be exactly 0 or 1.
```

harness/planner.py

```
        return True
      if len(recent_actions) >= 4 and recent_actions[-4:-2] == recent_actions[-2:]:
        return True
+     # Near-identical replan loop:  the planner re-issues the same action
+     # structure with only minor coordinate tweaks (e.g.  grasp x 48 -> 41)
+     # after a failed attempt, so exact-repeat detection above misses it.
+     recent_vectors = [
+      self._numeric_action_vector(record.get("action"))
+      for record in self._executed_action_records[-6:]
+     ]
+     recent_vectors = [vector for vector in recent_vectors if vector is not None]
+     if len(recent_vectors) >= 3:
+      first, second, third = (
+       recent_vectors[-3],
+       recent_vectors[-2],
+       recent_vectors[-1],
+      )
+      if len(first) == len(second) == len(third):
+       deltas = [
+        max(
+         abs(a - b)
+         for a, b in zip(first, second)
+        ),
+        max(
+         abs(a - b)
+         for a, b in zip(second, third)
+        ),
+        max(
+         abs(a - b)
+         for a, b in zip(first, third)
+        ),
+       ]
+       if all(delta <= 10.0 for delta in deltas):
+        return True
      return False


                                                ...
       f"{action_text}"
      )
+   @staticmethod
+   def _numeric_action_vector(action:  Any) -> Optional[List[float]]:
+    if not isinstance(action, (list, tuple)) or len(action) < 3:
+     return None
+    values:  List[float] = []
+    for value in action:
+     if isinstance(value, bool) or not isinstance(value, (int, float)):
+      return None
+     values.append(float(value))
+    return values
+
     def _recent_repeated_loop(self) -> bool:
      recent_actions = [
      self._action_repeat_key(record.get("action_id"))
```

harness/tool_orchestrator.py

```
       or "Use complementary visual evidence before planning."
      )
      override_marker = self._path_failure_marker(observed_evidence)
+
+     # Redundancy guard:  if the same vision tool already ran successfully
+     # in a recent planning event of this episode and no fresh
+     # environment feedback (e.g.  a path-planning failure) has arrived
+     # since, skip re-issuing it.  The planner keeps its RGB frame and
+     # recent feedback, so a repeated identical segmentation call adds
+     # no new evidence while consuming the per-plan vision budget.
+     if not override_marker:
+      redundant_tool = self._redundant_vision_tool(
```

```
+        requested_tools,
+        episode_state,
+        history,
+       )
+       if redundant_tool:
+        return {
+         "vision_orchestration_diagnostics": {
+          "source": "expanded_manipulation_harness",
+          "tool": "",
+          "requested_tool": requested_tool,
+          "requested_tools": requested_tools,
+          "final tools": [],
+          "sequence_length": 0,
+          "max calls": self.max calls,
+          "routing_source": "redundancy_guard",
+          "selector_overridden": False,
+          "override_marker": "",
+          "selected_by_selector": selected,
+          "reason": (
+           f"Skipped {redundant_tool}: the same tool "
+           "already delivered a successful result in a "
+           "recent planning event and no fresh environment "
+           "feedback has arrived since."
+          )[:500],
+          "selector_reason": selector_reason[:500],
+         }
+        }
```

...

```
+
+         # Check that no fresh feedback arrived after this call.
+         # The entry's own env_feedback is the feedback that was
+         # present when the tool ran; if the latest history entry
+         # has different (non-empty) feedback, the scene may have
+         # changed and the call is not redundant.
+         entry_feedback = str(entry.get("env_feedback", "")).strip()
+         latest = history[-1] if history else {}
+         latest_feedback = (
+          str(latest.get("env_feedback", "")).strip()
+          if isinstance(latest, dict)
+          else ""
+         )
+         if latest_feedback and latest_feedback != entry_feedback:
+          continue
+         return tool_name
+       return ""
```