Harness Optimization for Embodied Agents via Experience Traces

Hyun Jung Lee
Korea University

Jungtaek Kim
University of Arkansas

Jongwon Jeong
University of Wisconsin–Madison

Tae-Eui Kam
Korea University

Donghyun Kim*
Korea University

Yong Jae Lee*
University of Wisconsin–Madison

Abstract

The performance of embodied agents is shaped by the capabilities of the underlying vision-language model and by the harness that structures interaction with the environment, including prompting, tool use, and failure recovery. Refining this harness can improve agent's behavior without model weight updates, yet manual harness design remains labor-intensive and nontrivial. We introduce *EMbodied Harness Optimization (EMHO)*, a framework that evolves the harness of a frozen embodied model from interaction experience under sparse environmental feedback. EMHO collects execution traces, identifies behavioral weaknesses, and evaluates harness revisions through sequential interaction. Across navigation and manipulation tasks, EMHO improves task performance and generalizes to held-out environments without updating the model. Qualitative analysis shows that evolved harnesses learn reusable recovery behaviors, including breaking navigation loops and verifying task states after failed actions.

1 Introduction

Recent progress in embodied agents has largely been driven by advances in the underlying language and vision-language models [3, 2, 7, 5]. However, an agent's behavior depends not only on the capabilities encoded in the model weights, but also on the agent harness that structures how the model interacts with the environment. The harness specifies how the agent plans, selects actions and tools, recovers from failures, and presents observations and interaction history to the model. Consequently, even with the same frozen model, different harness designs can lead to different behaviors and task performance. Yet manually refining a harness is labor-intensive and nontrivial, as it requires examining long interaction traces across many episodes to identify recurring behavioral weaknesses.

This suggests a method to improve agent's performance: *can an agent improve a harness through its own experience?* Recent work on self-evolving agents [14, 6] has demonstrated the effectiveness of this approach in domains such as coding and command line interfaces, where execution traces and evaluation outcomes can be used to iteratively revise agent-side logic and improve performance. Extending this idea to embodied agents, however, is more challenging. Coding environments and command line interfaces often provide highly diagnostic feedback, including exceptions, failed tests, and localized execution errors, whereas embodied environments may offer only action-validity signals during interaction and a binary success signal at the end of an episode.

Despite this limited feedback, recent embodied-agent methods have explored improvement from interaction experience by evolving reusable skills, procedural knowledge, or recovery mechanisms [4,

*Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026). Workshop: Self-Evolving Diversity-Driven Search for Robust AI Systems.

13, 8]. Our work builds on this direction by optimizing the broader agent harness, which governs planning, context and memory management, tool use, action generation, and failure recovery.

We therefore ask: *can an embodied agent improve its own harness from interaction experience when environmental feedback is sparse and weakly diagnostic?* To study this question, we introduce *EMbodied Harness Optimization (EMHO)*, a framework that iteratively improves the harness surrounding a frozen embodied model. In the inner loop, the agent solves embodied tasks and collects execution traces containing observations, plans, actions, tool usage, feedback, and outcomes. In the outer loop, a harness optimizer analyzes these traces together with the history of previous harness modifications and their performance, identifies recurring behavioral weaknesses, and edits the harness. Candidate harnesses are then evaluated through actual embodied interaction, and better-performing variants are carried forward. Across manipulation and navigation tasks, EMHO improves task performance without any model updates. Importantly, the improvements extend beyond the search tasks used during optimization to held-out unseen tasks, suggesting that the resulting harness changes capture generalizable behavioral improvements rather than simply overfitting to observed failures.

2 EMbodied Harness Optimization (EMHO)

Overview. We optimize the harness H of a frozen embodied agent. The harness comprises the agent-side instructions and logic that structure interaction with the environment, including context and memory management, planning and action generation, tool use, and failure recovery. Let M denote the frozen embodied model and $\mathcal{H}$ the space of possible harnesses. Our objective is to find an optimized harness H^* that maximizes expected task success:

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

where $y_e \in \{0, 1\}$ denotes the final outcome of episode e. Throughout optimization, the model, environment, action space, and tool implementation remain fixed. EMHO consists of two nested loops. The inner loop executes the current harness and collects interaction experience. The outer loop analyzes this experience together with the previous harness revisions and their evaluation results, proposes a new candidate, and evaluates it through embodied interaction.

Inner Loop: Experience Trace Collection. In the inner loop, at optimization iteration i, the embodied agent interacts with the environment under the current harness $H^{(i)}$, producing an execution trace $\mathcal{T}_e^{(i)}$ for each episode e:

$$\mathcal{T}_e^{(i)} = \left(\{(o_t, p_t, z_t, a_t, f_t)\}_{t=1}^{T_e}, y_e \right), \tag{2}$$

where o_t is the current observation, p_t is the generated plan, z_t is a tool interaction when applicable, a_t is the executed action, f_t denotes feedback from the environment, and T_e denotes the number of interaction steps in episode e. We define the collected trace set at iteration i as $\mathcal{E}^{(i)} = \{\mathcal{T}_e^{(i)} \mid e \in \mathcal{D}_{\text{search}}\}$ and additionally compute aggregate statistics such as task success, invalid actions, output errors, tool-use patterns, and recovery behavior. These traces are important because embodied feedback is often weakly diagnostic. A failed action or episode indicates that the behavior was unsuccessful, but does not directly identify whether the error arose from planning, context handling, tool use, or recovery. The traces therefore provide evidence for identifying recurring behavioral weaknesses across episodes.

Outer Loop: Harness Refinement. At optimization iteration i, the harness optimizer receives the current harness $H^{(i)}$, its collected experience traces $\mathcal{E}^{(i)}$, and an optimization log $\mathcal{L}^{(i-1)}$ containing the previous harness revisions and their evaluation results:

$$\left(q^{(i)}, H_{\text{evol}}^{(i+1)} \right) = \mathcal{O}\left(H^{(i)}, \mathcal{E}^{(i)}, \mathcal{L}^{(i-1)} \right). \tag{3}$$

Here, $q^{(i)}$ denotes a hypothesis about a recurring behavioral weakness, and $H_{\text{evol}}^{(i+1)}$ denotes the evolved harness produced by the optimizer for the next iteration. The optimizer searches for recurring patterns across execution traces rather than attributing individual failures to a specific cause. Proposed edits may modify any editable harness component, including planning instructions, context construction, tool routing, memory, output validation, replanning, and recovery logic. Each candidate is evaluated through actual environment interaction, and better-performing candidates are retained for subsequent optimization iterations.

Table 1: Success rates on EmbodiedBench. The initial harness reports results averaged across the search and held-out splits, whereas evolved harness results are reported separately. Bold entries denote the highest score in each subtask. The upper block of each benchmark is from Yang et al. [12].

Model	EB-Navigation					
	Avg	**Base**	**Common**	**Complex**	**Visual**	**Long**
GPT-4o	57.7	55.0	60.0	58.3	60.0	55.0
GPT-4o-mini	32.8	31.7	33.3	35.0	28.3	33.3
Claude-3.7-Sonnet	45.0	50.0	61.7	50.0	36.7	26.7
Gemini-1.5-Pro	24.3	23.3	25.0	25.0	28.3	20.0
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
InternVL3-38B	50.7	55.0	61.7	55.0	56.7	25.0
Qwen2.5-VL-72B-Ins	40.0	46.7	46.7	46.7	26.7	33.3
gemma-3-27b-it	45.4	53.3	45.0	61.7	50.0	16.7
gemma-3-12b-it	34.0	38.3	36.7	48.3	40.0	6.7
Initial Harness (Qwen3.5-9B)	49.3	58.3	50.0	61.7	36.7	40.0
Modified Harness – Opus (Qwen3.5-9B)	53.3	56.7	60.0	63.3	36.7	50.0
Modified Harness – Sol (Qwen3.5-9B)	49.3	50.0	53.3	63.3	40.0	40.0
EMHO (Search, Qwen3.5-9B)	62.3	71.7	70.0	70.0	50.0	50.0
EMHO (Held-out, Qwen3.5-9B)	55.3	60.0	60.0	65.0	40.0	51.7
Initial Harness (Qwen3.8-27B)	60.3	66.7	61.7	58.3	58.3	56.7
One-Shot Harness – Opus (Qwen3.8-27B)	67.7	78.7	70.0	**75.0**	58.3	56.7
One-Shot Harness – Sol (Qwen3.8-27B)	65.7	73.3	70.0	61.7	63.3	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**

Model	EB-Manipulation					
	Avg	**Base**	**Common**	**Complex**	**Visual**	**Spatial**
GPT-4o	28.9	39.6	29.2	29.2	19.4	25.0
GPT-4o-mini	4.8	4.2	6.3	2.1	0.0	10.4
Claude-3.7-Sonnet	28.5	31.3	20.8	**43.8**	25.0	20.8
Gemini-1.5-Pro	21.1	14.6	14.6	22.9	16.7	35.4
Gemini-2.0-flash	16.7	14.6	8.3	14.6	13.9	31.3
InternVL3-78B	26.3	29.2	22.9	22.9	25.0	31.3
InternVL3-38B	22.6	20.8	14.6	20.8	19.4	37.5
Qwen2.5-VL-72B-Ins	16.2	12.5	12.5	16.7	22.2	18.8
gemma-3-27b-it	17.5	25.0	16.7	16.7	8.3	20.8
gemma-3-12b-it	20.6	20.8	22.9	20.8	19.4	18.8
Initial Harness (Qwen3.5-9B)	9.9	7.3	6.3	7.3	16.7	13.5
One-Shot Harness – Opus (Qwen3.5-9B)	13.2	10.4	16.7	12.5	13.9	12.5
One-Shot Harness – Sol (Qwen3.5-9B)	11.8	12.5	6.3	10.4	11.1	18.8
EMHO (Search, Qwen3.5-9B)	18.0	14.6	12.5	16.7	25.0	22.9
EMHO (Held-out, Qwen3.5-9B)	20.2	14.6	20.8	18.8	36.1	14.6
Initial Harness (Qwen3.8-27B)	24.6	20.8	18.8	29.2	36.1	20.8
One-Shot Harness – Opus (Qwen3.8-27B)	27.2	22.9	20.8	29.2	38.9	27.1
One-Shot Harness – Sol (Qwen3.8-27B)	25.4	25.0	25.0	27.1	22.2	27.1
EMHO (Search, Qwen3.8-27B)	**43.0**	**50.0**	**41.7**	33.3	**50.0**	**41.7**
EMHO (Held-out, Qwen3.8-27B)	37.7	41.7	33.3	33.3	38.9	**41.7**

3 Experimental Results

Setup. We conduct experiments on EmbodiedBench [12], covering both navigation and manipulation tasks. We use the Qwen3.5-9B model [10] and the Qwen3.8-27B-FP8 model [11] as the underlying embodied agent and keep their weights, action space, and tools fixed throughout optimization. For each domain, we divide the benchmark evenly into search and held-out splits with no overlapping environments between them. The search split is used to collect interaction traces and optimize the harness, while the resulting harness is evaluated without further modification on the held-out split. Example harness structure is presented in Appendix A.

Main Results. As shown in Table 1, EMHO improves overall performance on both navigation and manipulation, and these gains largely carry over to the held-out sets. Although the 9B model initially achieves lower navigation performance than the substantially larger InternVL3-38B and InternVL3-78B, the evolved harness enables it to exceed the published full-benchmark scores of both models on both the search and held-out splits. The improvements are not limited to relatively simple Base tasks, but also appear in the categories that require longer-horizon progress monitoring or more

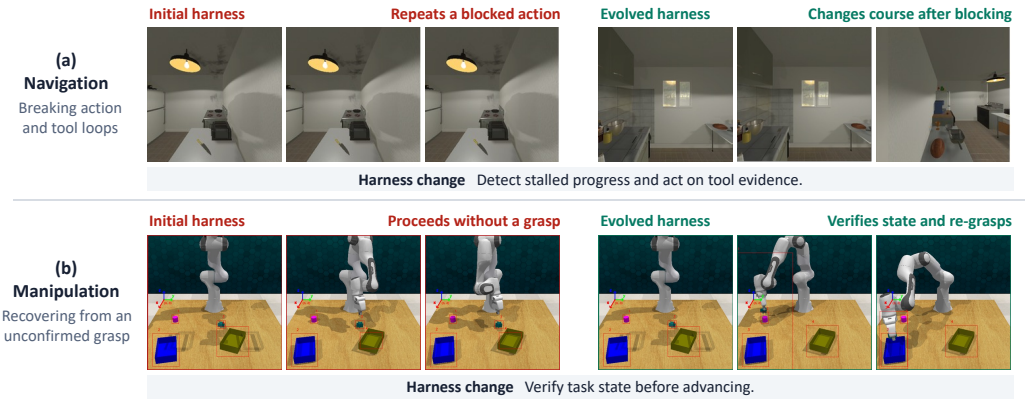


Figure 1: Representative recovery behaviors before and after harness evolution.

reliable state estimation, including Complex and Long navigation tasks and Visual manipulation tasks. EMHO also improves the 27B model across subtasks in both domains. The gains are particularly pronounced in manipulation, where an incorrect estimate of the task state can propagate through subsequent actions unless the harness explicitly supports verification and recovery. This suggests that the method is not limited to compensating for the weaknesses of a smaller model, but can also help a stronger model use its existing capabilities more effectively during embodied interaction.

One-Shot Harness Revision. We compare EMHO with one-shot harness revisions generated by Claude Opus-5 [1] and GPT-5.6 Sol [9]. For each domain, these models receive the initial harness, a description of EmbodiedBench, and 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. EMHO benefits from grounding each revision in the agent's observed behavior. 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, prevents the agent from retrying the same direction, and replans from the updated observation. Each candidate is then evaluated through embodied interaction, allowing ineffective changes to be discarded and useful rules to be retained or refined.

Qualitative Analysis. Figure 1 shows that the accepted harness edits primarily improve progress monitoring and recovery. In navigation, the initial harness repeatedly attempts blocked movements and continues invoking tools without converting their outputs into a change in behavior. The evolved harness detects stalled progress, selects tools according to the current failure context, and requires the resulting evidence to inform the next action. In manipulation, the initial harness proceeds to placement even when the preceding pickup has failed, since this failure does not necessarily produce an invalid-action signal. The evolved harness verifies whether the intended state change has occurred before advancing to the next subgoal. When a grasp remains unconfirmed, it re-estimates the relevant coordinates and restarts the pickup sequence. These examples illustrate how interaction traces allow the optimizer to identify implicit failures and translate them into reusable recovery rules despite limited environmental feedback. Additional examples of the concrete harness modifications underlying these behaviors are provided in Appendix B.

4 Limitations and Future Work

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.

Acknowledgments

This work was supported in part by NSF (IIS2404180), IBM, and grants from the Institute of Information & communications Technology Planning & Evaluation (IITP) funded by the Korea government (MSIT) (No. 2022-0-00871, Development of AI Autonomy and Knowledge Enhancement for AI Agent Collaboration; No. RS-2025-2543949, Environment-Aware and Domain-Adaptive Multimodal Embodied AI for Real-World Interaction; No. RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University); and No. RS-2025-25439490). Additional support was provided by the Electronics and Telecommunications Research Institute (ETRI) grant (26CB1200, Development and Application of Science-Specialized Multimodal Foundation Models), the KOCCA grant (RS-2024-00345025), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-25302986). This work also used resources provided through CIS251382 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, supported by NSF awards OAC-2138259, OAC-2138286, OAC-2138307, OAC-2137603, and OAC-2138296. This research was also supported by a grant from the Korean ARPA-H Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health and Welfare, Republic of Korea (RS-2025-25455839).

References

[1] Anthropic. Introducing Claude Opus 5. `https://www.anthropic.com/news/claude-opus-5`, 2026. Accessed: October 29, 2026.

[2] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of the Conference on Robot Learning*, 2025.

[3] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence. PaLM-E: An embodied multimodal language model. In *Proceedings of the International Conference on Machine Learning*, 2023.

[4] R. Ju, X. Wang, X. Ding, Y. Yang, H. Wu, S. Jiang, Q. Zhang, H. Wen, X. Li, W. Wang, K. Li, Y. Liu, H. Dai, W. Wang, and T. Cao. EmbodiSkill: Skill-aware reflection for self-evolving embodied agents. *arXiv preprint arXiv:2605.10332*, 2026.

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the Conference on Robot Learning*, 2024.

[6] Y. Lee, R. Nair, Q. Zhang, K. Lee, O. Khattab, and C. Finn. Meta-Harness: End-to-end optimization of model harnesses. In *Proceedings of the Conference on Language Modeling*, 2026.

[7] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. M. M. Shafiullah, and L. Pinto. Demonstrating OK-Robot: What really matters in integrating open-knowledge models for robotics. In *Proceedings of Robotics: Science and Systems*, 2024.

[8] R. Lu, Y. Wu, E. Kou, H. Lin, L. Fu, W. Xiao, A. Mandlekar, Y. Xu, G. Shi, K. Goldberg, A. Chen, M. Chowdhury, J. Kautz, Y. Zhu, L. Fan, and G. Wang. ASPIRE: Agentic /Skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

[9] OpenAI. GPT-5.6 Sol. `https://developers.openai.com/api/docs/models/gpt-5.6-sol`, 2026. Accessed: October 29, 2026.

[10] Qwen Team. Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, 2026. Accessed: October 29, 2026.

[11] Qwen Team. Qwen3.8-27B-FP8 model card. `https://huggingface.co/Qwen/Qwen3.8-27B-FP8`, 2026. Accessed: October 29, 2026.

[12] R. Yang, H. Chen, J. Zhang, M. Zhao, C. Qian, K. Wang, Q. Wang, T. V. Koripella, M. Movahedi, M. Li, H. Ji, H. Zhang, and T. Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the International Conference on Machine Learning*, 2025.

[13] Y. Yang, N. P. Bhatt, K. Wang, S. Tetteh, Z. Wang, and U. Topcu. VASO: Formally verifiable self-evolving skills for physical AI agents. *arXiv preprint arXiv:2606.05395*, 2026.

[14] H. Zhang, S. Zhang, K. Li, C. Zhang, Y. Chen, Y. Zhang, L. Bai, and S. Hu. Self-Harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026.

Appendices

A Details of Harness Structure

Table 2: Overview of the main components of the navigation and manipulation harness.

Component	Main files	Role in the harness

Prompt and task contract	`SKILL.md`
	`system.md`
	`final_planner.md`
		Defines the planner role, admissible evidence, action-space constraints, output format, and final-plan generation requirements.

Runtime and planning	`agent.py`
	`planner.py`
	`model_client.py`
		Implements the main solver loop by constructing model inputs, invoking the planner, executing generated actions, and triggering replanning from environment feedback.

Vision and tool orchestration	`tool_orchestrator.py`
	`vision_policy.py`
	`tool_select.md`
		Controls the use and ordering of visual evidence tools such as zoom, segmentation, and depth, and determines how their outputs are supplied to the planner.

Context, memory, and modules	`module_selector.py`
	`memory.py`
	`memory_render.md`
	`safe_context.md`
		Selects model-facing context modules and maintains bounded episode history, previous actions, feedback, and 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 model-facing 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 execution traces.

B Detailed Results of Harness Modifications

The examples below illustrate representative changes discovered for navigation and manipulation.

Navigation. For navigation, harness optimization primarily focused on improving feedback-aware replanning and preventing repeated ineffective behavior. The initial harness could continue executing a plan after a movement was blocked, repeatedly issue similar actions with only minor coordinate changes, or invoke visual tools without translating their outputs into a different navigation decision. The optimized harness was therefore revised to incorporate recent environment feedback and action history explicitly when constructing the next plan. In particular, it encourages the agent to terminate the remainder of an ineffective plan after a blocked or failed action, avoid repeating recently unsuccessful movements, and replan from the latest observation. The planning instructions were also refined to consider the target's relative direction, distance, alignment, and nearby obstacles rather than relying on target visibility alone. In addition, tool-selection logic was made more selective: previously successful tool outputs can be reused when no new evidence has appeared, while redundant vision calls are skipped unless they are expected to provide new spatial information. Together, these changes aim to convert execution feedback into concrete changes in behavior rather than allowing the agent to remain in navigation loops.

`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. For manipulation, optimization focused on maintaining consistency between the intended subgoal and the actual scene state. A common failure mode of the initial harness was to continue to the next manipulation phase even when the preceding action had not achieved its intended effect. For example, the agent could proceed from grasping to placement without confirming that the object had actually been picked up. The optimized harness therefore introduces stronger state verification and recovery rules between manipulation phases. After a grasp, lift, or placement attempt, the planner is instructed to inspect the current observation and determine whether the expected state transition occurred before continuing. If the object is not securely grasped or the previous sequence has been exhausted without success, the agent replans from the current state instead of mechanically repeating the same approach–descend–grasp sequence. The revised planner also uses recent failed and successful actions to avoid near-duplicate retries, while allowing small coordinate adjustments when they are justified by newly observed evidence. Output validation and repair logic were additionally strengthened so that malformed or invalid plans are replaced with executable actions drawn from the valid action set. These changes are intended to make manipulation behavior more robust to implicit failures that may not produce an explicit invalid-action signal.

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 ""
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction describe harness optimization for frozen embodied agents, and Section 3 reports the navigation and manipulation results supporting these claims. The empirical findings concern the models and benchmarks evaluated in this paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4 discusses the divergence between search-split and held-out performance, the risk of overfitting during harness optimization, and the need to consider generalization when selecting harness iterations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper presents an empirical method and does not introduce theoretical results that require assumptions and proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 2 and 3 describe the harness optimization procedure and evaluation protocol. Appendices A and B provide additional details on the harness structure and representative code modifications.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The submission does not provide an executable release of the EMHO implementation or scripts and instructions for reproducing the experiments. The experiments use existing EmbodiedBench tasks, with the method and representative harness modifications described in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3 describes the evaluated models, benchmarks, search and held-out splits, and evaluation procedure. The inner and outer loops use the same model, and harness optimization is performed for 10 iterations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report empirical task success rates without error bars, confidence intervals, or statistical significance tests.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper does not report detailed computational resources or the execution times required for the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors confirm that the research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

Justification: The paper focuses on harness optimization and empirical evaluation in embodied benchmarks and does not include a discussion of the potential positive and negative societal impacts of the proposed method.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This submission does not release new pretrained model weights, scraped datasets, or other high-risk data or model assets requiring release safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The original sources of the benchmarks and models used in the experiments are cited in Section 3 and the references.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new standalone dataset, model, or software package is released as part of this submission.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research evaluates agents on existing embodied benchmarks and does not involve crowdsourcing or experiments with human participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The research does not involve human subjects; therefore, participant risk disclosures and human-subject research approvals are not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Sections 2 and 3 describe the use of language and vision-language models in the embodied agent, harness optimization, and one-shot revision baselines. The same model is used for the inner-loop embodied agent and the outer-loop harness optimizer.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.