EMBODISKILL: SKILL-AWARE REFLECTION FOR SELF-EVOLVING EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reusable skills are essential for embodied agents because they provide procedural guidance for acting under changing environments. However, embodied skills often express high-level guidance that must be translated into context-dependent actions, creating a gap between skill guidance and action realization. A failed trajectory may therefore arise from either a skill defect or an execution lapse in which valid guidance is present but overlooked. Existing skill self-evolution methods often do not distinguish these causes and may unnecessarily revise guidance that is already correct. We introduce **EMBODISKILL**, a training-free framework that uses skill-aware reflection to separate skill-changing evidence from execution lapses. It revises the skill body when guidance needs to be added, refined, or corrected, while updating an appendix to emphasize valid but overlooked guidance. Experiments on ALFWorld and EmbodiedBench show consistent improvements over direct-execution, memory-based, and skill-based baselines. On ALFWorld, under a fully local and model-matched setting, EmbodiSkill improves a frozen Qwen3.5-27B executor from 61.19% to 86.57% task success. Using GPT-5.2 only for offline skill evolution further increases task success to 93.28%.

1 INTRODUCTION

Skills are critical for agents because they provide reusable procedural knowledge that helps agents solve tasks more reliably across repeated executions (Wang et al., 2024; Fang et al., 2026; Cao et al., 2026; Ni et al., 2026; Zhang et al., 2026a; Mi et al., 2026). For embodied agents, such knowledge is especially important because high-level instructions must be carried out by the procedural guidance under changing perceptions, object states, and action feasibility in physical environments (Shridhar et al., 2020; 2021; Yang et al., 2025). Failing to follow the skills can easily move the embodied agent into a stuck state not covered by training or prior experience and affect many downstream decisions.

To accumulate experience and avoid relying on fixed skills, recent work studies *skill self-evolution*: explicit skills are updated offline from the agent's own interaction trajectories and reused in subsequent executions without retraining the executor (Wang et al., 2024; Fang et al., 2026; Cao et al., 2026; Ni et al., 2026; Zhang et al., 2026a; Mi et al., 2026). These methods commonly treat trajectory outcomes as evidence for modifying skill content. In particular, a failed trajectory is often assumed to indicate that the current skill is missing, incorrect, or insufficient, and the skill is then revised accordingly.

However, directly applying this skill self-evolution paradigm to embodied tasks can be problematic. By examining failed trajectories in ALFWorld, we observe a characteristic failure mode: even when the current skill contains correct task-relevant guidance, the agent may still fail because it overlooks or fails to apply that guidance during execution. We refer to such failures as execution lapses, as illustrated in Figure 1 (left).

This overlooking stems from the larger gap between skill guidance and action realization in embodied environments. In digital environments, standardized interfaces often make the procedure for the same task relatively stable across instances, allowing skills to specify tool calls, API invocations, or interface-level action sequences directly. In embodied environments, however, the required action sequence varies with scene layout, object state, visibility, and agent position. To remain reusable, embodied skills must therefore express higher-level guidance, such as prerequisites, subgoal ordering, and recovery strategies. Realizing this guidance requires perception, state reasoning,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

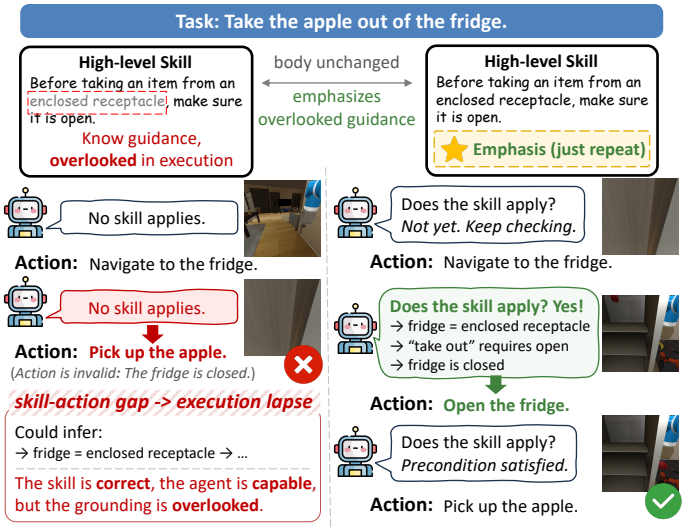


Figure 1: An example of EmbodiSkill recovering an execution lapse by emphasizing overlooked guidance. Without emphasis, the executor overlooks valid guidance and attempts to pick the apple from a closed fridge. When only the previously overlooked guidance is explicitly emphasized, the executor recognizes its applicability and follows the correct sequence, while the skill content remains unchanged.

and context-dependent action generation, creating more opportunities for valid skill content to be overlooked, misapplied, or only partially followed.

Interestingly, we observe that such failures can sometimes be resolved by explicitly emphasizing the overlooked guidance while keeping the underlying skill content unchanged, as illustrated in Figure 1 (right). This suggests that embodied skill self-evolution should not only ask how a trajectory should change a skill, but also determine whether the skill should be changed at all. We instantiate this view with **EMBODISKILL**, a training-free framework built on two key designs. First, *skill-aware reflection* distinguishes evidence that warrants skill revision from execution lapses where valid guidance is overlooked. Second, EMBODISKILL separates the skill into a *body* and an *appendix*. The body is revised when the skill itself needs improvement. The appendix instead emphasizes valid but overlooked guidance without changing the skill body. Together, these designs enable reliable skill self-evolution by revising deficient guidance while preserving and emphasizing valid guidance that is overlooked during execution.

We evaluate EmbodiSkill on embodied benchmarks covering visual object interaction, navigation, and household task completion (Shridhar et al., 2021; Yang et al., 2025). On ALFWorld, under fully local and model-matched settings, EmbodiSkill enables a frozen open-weight Qwen3.5-27B executor to achieve 86.57% task success, outperforming all compared baselines. Using GPT-5.2 as the offline evolution model further improves performance, with task success increasing from 72.39% after the first skill update to 93.28% over subsequent evolution stages. This final result represents relative improvements of 31.58% over GPT-5.2 and 13.63% over Gemini-3-flash when they are used directly as task executors. Together, these results show that distinguishing skill revision from execution emphasis enables reliable skill self-evolution and more effective refinement of procedural guidance over successive evolution stages.

2 RELATED WORK

2.1 REUSABLE EMBODIED SKILLS

Embodied agents commonly rely on reusable skills (Li et al., 2026), action primitives, or programmatic policies to translate high-level instructions into executable behaviors (Song et al., 2023; Puig et al., 2018). SayCan selects pretrained robotic skills by jointly considering their semantic relevance

and physical feasibility (Ichter et al., 2023), while Code as Policies and ProgPrompt compose perception and control primitives into executable programs for situated task execution (Liang et al., 2023; Singh et al., 2023). More recent systems such as CaP-X further improve flexibility by synthesizing skills and programs from execution feedback (Fu et al., 2026). These approaches effectively connect task-level reasoning with embodied action (Yao et al., 2023), but their effectiveness depends heavily on the coverage of available primitives and the reliability of generated programs. Predefined skills may not capture novel object states or execution conditions, whereas programs synthesized for individual interactions can be sensitive to scene-specific feedback and produce local fixes that do not generalize across subsequent executions. These limitations motivate mechanisms that consolidate interaction experience into persistent and reusable procedural guidance.

2.2 Skill Self-Evolution

Skill self-evolution enables agents to construct and refine persistent procedural guidance from interaction trajectories (Zhang et al., 2026b; Yang et al., 2026; Xia et al., 2026; Zhao et al., 2024). Voyager generates, verifies, and accumulates executable skills through exploration in Minecraft (Wang et al., 2024); Skill-Pro iteratively optimizes reusable procedural skills from interaction experience (Mi et al., 2026); and XSkill extracts task-level skills together with action-level experience from multimodal trajectories (Jiang et al., 2026). Trace2Skill improves skill evolution by consolidating trajectory-local lessons into transferable skills (Ni et al., 2026), while CoEvoSkills co-evolves skill generators and verifiers to construct reliable skill packages (Zhang et al., 2026a). In embodied settings, ASPIRE diagnoses execution failures, repairs control programs, and distills validated solutions into a reusable skill library (Lu et al., 2026). These methods primarily investigate how trajectory feedback should be used to create, refine, consolidate, verify, or repair skills. AutoManual (Chen et al., 2024) further distinguishes failures caused by imperfect rules from those caused by an imperfect agent, and modifies the corresponding rules based on this distinction. However, these approaches still incorporate trajectory feedback by modifying the skill or rule itself, which can be problematic when a failure arises from valid guidance being overlooked during execution: such modification may introduce redundant or conflicting content, distort guidance that is already correct, or encode an incidental execution error as a general procedural rule. EmbodiSkill instead separates skill revision from execution emphasis, preserving the skill body and updating a separate appendix in such cases.

3 Method

We present EmbodiSkill, a training-free framework for evolving embodied skills from interaction trajectories without updating model parameters. An overview of the framework is shown in Figure 2. Its core component, *skill-aware reflection*, compares each trajectory with the current skill to determine whether it warrants skill revision or instead indicates an execution lapse. Accordingly, we organize the skill into a *body* that stores consolidated procedural guidance and an *appendix* that emphasizes valid but overlooked guidance. Skill-changing evidence revises the body, whereas execution-lapse evidence preserves the body and updates the appendix.

3.1 Problem Formulation

We consider an embodied agent executing a natural-language task instruction I in a physical or physically grounded environment. The agent observes the environment and produces actions over a finite horizon, yielding a trajectory

$$\tau = (I, o_1, a_1, \ldots, o_T, a_T, r), \tag{1}$$

where o_t and a_t denote the observation and action at step t, respectively, and $r \in \{0, 1\}$ is the final task-success signal.

At evolution stage n, the agent is guided by the current skill

$$S^{(n)} = \left(S^{(n)}_{\text{body}}, S^{(n)}_{\text{app}}\right). \tag{2}$$

The skill body stores consolidated procedural guidance, while the appendix selectively emphasizes valid guidance already contained in the body and introduces no new skill rules.

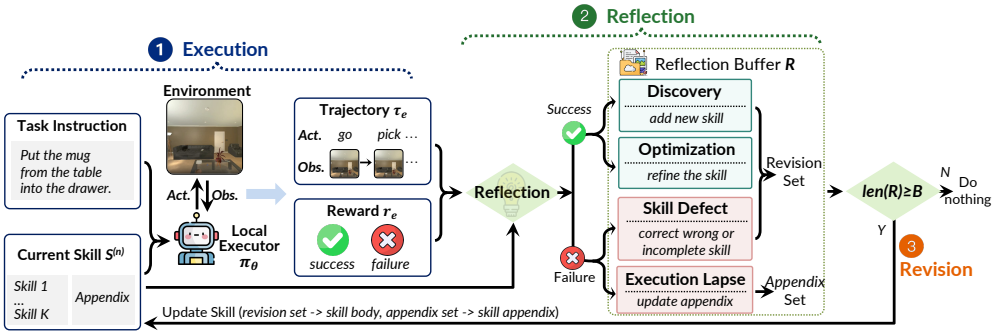


Figure 2: Overview of EmbodiSkill's closed-loop skill evolution. A frozen executor collects trajectories under the current body-and-appendix skill, and skill-aware reflection classifies the resulting evidence as Discovery, Optimization, Skill Defect, or Execution Lapse. Once sufficient reflection evidence has been accumulated, the first three outcomes revise the skill body, whereas Execution Lapse updates only the appendix to emphasize overlooked guidance for subsequent execution.

Given the interaction history $h_t = (o_1, a_1, \ldots, o_t)$, the executor π_θ generates actions according to

$$a_t \sim \pi_\theta\Big(\cdot \mid I, S^{(n)}, h_t\Big). \tag{3}$$

The executor parameters θ remain fixed throughout skill evolution.

After trajectories are collected under the current skill $S^{(n)}$, a skill evolution model F analyzes them offline and produces the next skill version $S^{(n+1)}$. Repeating this process yields an evolving skill sequence $S^{(0)} \to S^{(1)} \to \cdots \to S^{(N)}$.

3.2 Skill-Aware Reflection

Skill-aware reflection determines whether and how each trajectory should influence skill evolution. Rather than directly converting trajectory feedback into a skill update, it jointly examines the trajectory τ and the current skill $S^{(n)}$ to produce a set of valid reflection records:

$$\mathcal{R}_\tau = \mathrm{Reflect}\Big(F; \tau, S^{(n)}, K\Big), \qquad 0 \le |\mathcal{R}_\tau| \le K, \tag{4}$$

where K is the maximum number of reflection records produced from one trajectory.

Each reflection record is represented as

$$\rho_i = (c_i, q_i, t_i, u_i), \tag{5}$$

where c_i denotes the reflection outcome, q_i provides the attribution rationale by comparing the trajectory with the current skill, t_i identifies the target skill content, and u_i specifies the corresponding skill-evolution directive.

For each reflection record, the admissible outcome c_i depends on the trajectory result: a successful trajectory admits Discovery or Optimization, whereas a failed trajectory admits Skill Defect or Execution Lapse.

Discovery. The trajectory reveals reusable procedural knowledge absent from the current skill body and proposes adding it. Therefore, no existing target content is required and $t_i = \varnothing$.

Optimization. The trajectory demonstrates a more effective procedure for valid existing guidance and proposes refining the corresponding target content.

Skill Defect. The trajectory reveals that the target guidance is incorrect, incomplete, or underspecified and proposes a targeted correction.

Execution Lapse. The trajectory reveals that valid target guidance is already present but not followed, and proposes emphasizing it through the skill appendix without modifying the skill body.

Distinguishing Skill Defect from Execution Lapse is the central attribution step. If the existing guidance is incorrect or would remain insufficient even when followed, the trajectory indicates a Skill Defect. If the guidance is valid and the failure could have been avoided by following it, the trajectory indicates an Execution Lapse. Consequently, Discovery, Optimization, and Skill Defect provide *skill-changing evidence*, whereas Execution Lapse provides *execution-lapse evidence*. These two forms of evidence are accumulated separately and lead to different skill evolution operations.

3.3 Skill Revision and Emphasis

EmbodiSkill updates the skill after every B trajectories collected under the current version $S^{(n)}$, where B denotes the update frequency. The reflection records produced from these trajectories are accumulated as

$$\mathcal{R}^{(n)} = \bigcup_{j=1}^{B} \mathcal{R}_{\tau_j}. \tag{6}$$

The accumulated reflection records are first split by outcome:

$$\left(\mathcal{R}^{(n)}_{\text{disc}}, \mathcal{R}^{(n)}_{\text{opt}}, \mathcal{R}^{(n)}_{\text{def}}, \mathcal{R}^{(n)}_{\text{lap}}\right) = \text{Split}\left(\mathcal{R}^{(n)}\right). \tag{7}$$

Discovery, Optimization, and Skill Defect records provide skill-changing evidence, whereas Execution Lapse records provide execution-lapse evidence and are handled separately through the skill appendix. Before revising the skill body, EmbodiSkill removes redundant skill-changing records and resolves conflicting directives:

$$\mathcal{R}^{(n)}_{\text{chg}} = \text{Reconcile}\left(F; \mathcal{R}^{(n)}_{\text{disc}}, \mathcal{R}^{(n)}_{\text{opt}}, \mathcal{R}^{(n)}_{\text{def}}\right). \tag{8}$$

The resulting $\mathcal{R}^{(n)}_{\text{chg}}$ contains non-redundant and mutually consistent reflection records for targeted skill revision.

The reconciled skill-changing records are then used to update the skill body:

$$\left(S^{(n+1)}_{\text{body}}, \mathcal{T}^{(n)}_{\text{body}}\right) = \text{ReviseBody}\left(F; S^{(n)}_{\text{body}}, \mathcal{R}^{(n)}_{\text{chg}}\right). \tag{9}$$

Discovery records add previously absent guidance, while Optimization and Skill Defect records refine or correct their identified target content. Skill content not implicated by $\mathcal{R}^{(n)}_{\text{chg}}$ is preserved. Along with the revised body, $\mathcal{T}^{(n)}_{\text{body}}$ records how content in $S^{(n)}_{\text{body}}$ corresponds to $S^{(n+1)}_{\text{body}}$, indicating whether each relevant target is preserved, revised, or deleted.

Execution-lapse evidence follows a separate update path through the skill appendix:

$$S^{(n+1)}_{\text{app}} = \text{UpdateApp}\left(F; \mathcal{T}^{(n)}_{\text{body}}, S^{(n)}_{\text{app}}, \mathcal{R}^{(n)}_{\text{lap}}\right). \tag{10}$$

Based on $\mathcal{T}^{(n)}_{\text{body}}$, this operation preserves emphasis whose target remains unchanged, aligns emphasis with revised target content, and removes emphasis whose target has been deleted. It then incorporates new emphasis from $\mathcal{R}^{(n)}_{\text{lap}}$ and merges repeated items. The resulting appendix therefore emphasizes only valid guidance contained in $S^{(n+1)}_{\text{body}}$ and introduces no new procedural rules.

The resulting skill is

$$S^{(n+1)} = \left(S^{(n+1)}_{\text{body}}, S^{(n+1)}_{\text{app}}\right). \tag{11}$$

The complete EmbodiSkill procedure is summarized in Algorithm 1 in Appendix A.

By separating these two update paths, EmbodiSkill revises procedural guidance when skill-changing evidence warrants a content change, while preserving valid guidance and adjusting only its emphasis when the failure is attributed to an execution lapse.

Repeated across evolution stages, the revised skill guides subsequent executions, whose trajectories in turn provide new evidence for further revision and emphasis. This iterative closed loop constitutes skill self-evolution, supporting continual improvement in the skill's completeness, correctness, and executability.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks and Metrics. We evaluate EmbodiSkill on ALFWorld (Shridhar et al., 2021) and two tracks of EmbodiedBench (Yang et al., 2025): EmbodiedBench-Habitat (EB-Habitat) and EmbodiedBench-Navigation (EB-Navigation). ALFWorld evaluates household task completion involving object search, navigation, container interaction, and object-state changes. EB-Habitat evaluates visual object interaction in 3D environments (Savva et al., 2019), while EB-Navigation evaluates visual navigation. For ALFWorld, we use 500 tasks sampled from the official training split for skill evolution and evaluate on 134 held-out test tasks. As EmbodiedBench provides only evaluation tasks, we construct 500 additional evolution tasks for each track using the benchmark's official data-generation utilities and evaluate on the 300 official test tasks. Details of the task construction are provided in Appendix B. We report task success rate on all benchmarks. For ALFWorld, we report the overall score and six subtask categories; for EmbodiedBench, we report the overall score in the main paper and provide detailed subset-wise results in Appendix C.

Baselines. We compare EmbodiSkill with direct-execution, memory-based, and skill-based baselines. Direct-execution baselines include GPT-5.2 (OpenAI, 2025), Gemini-3-flash (Google DeepMind, 2025), and the corresponding local Qwen executor, all operating without external memory or skills. The closed-source models serve as strong capability references, while local direct execution provides an executor-matched baseline. Memory-based methods include Mem0 (Chhikara et al., 2025), G-Memory (Zhang et al., 2025), and LangMem (LangChain AI, 2025), which store and retrieve experience from interaction trajectories (Zhong et al., 2024; Xu et al., 2025). Skill-based methods include Skill-Pro (Mi et al., 2026), XSkill (Jiang et al., 2026), and Trace2Skill (Ni et al., 2026), which construct or refine reusable skills from trajectories. Unless otherwise specified, each memory-based and skill-based baseline uses the same local Qwen model for both task execution and memory construction or skill evolution. Across controlled comparisons, all methods are matched in the frozen executor, the memory or skill evolution model, and the evolution budget.

Implementation Details. EmbodiSkill collects trajectories on the evolution set and updates the external skill offline while keeping all model parameters fixed. We initialize the external skill as empty. Unless otherwise specified, we set the maximum number of reflection records per trajectory to $K = 1$, update the skill every $B = 50$ trajectories, and perform $N = 10$ evolution stages, yielding a total budget of 500 trajectories. We further study sensitivity to the update frequency B and reflection limit K under a fixed evolution budget in Appendix D. Evolution tasks are sampled in randomized order. During evaluation, the final evolved skill is fixed, and no test trajectory is used for further evolution. For ALFWorld, we use Qwen2.5-14B-Instruct (Yang et al., 2024) and Qwen3.5-27B (Qwen Team, 2026) as executors. For EB-Habitat and EB-Navigation, we use Qwen3-VL-8B-Instruct and Qwen3-VL-32B-Instruct (Bai et al., 2025). For EmbodiSkill, the skill evolution model F is instantiated with the corresponding local Qwen model, GPT-5.2, or Gemini-3-flash, as specified in each experiment.

4.2 MAIN RESULTS

Tables 1 and 2 report the main results on ALFWorld and EmbodiedBench, respectively.

Results on ALFWorld. EmbodiSkill consistently improves task success across executor scales. Under the fully local and model-matched setting, it improves Qwen2.5-14B from 46.27% without memory or skills to 79.85%, and Qwen3.5-27B from 61.19% to 86.57%, yielding gains of 33.58 and 25.38 percentage points, respectively. EmbodiSkill also outperforms existing experience-based methods under the same model configuration. With Qwen2.5-14B, it exceeds G-Memory, the strongest model-matched baseline, by 12.69 percentage points. With Qwen3.5-27B, it outperforms Skill-Pro, the strongest model-matched baseline, by 5.23 points. Under fully local and model-matched settings, EmbodiSkill outperforms all competing baselines across both executor scales.

Using a stronger evolution model while keeping the executor frozen further improves performance. Replacing the local evolution model with GPT-5.2 raises success from 79.85% to 86.57% for Qwen2.5-14B and from 86.57% to 93.28% for Qwen3.5-27B. The best configuration also surpasses

Table 1: Task success rate (%) on ALFWorld for direct-execution, memory-based, and skill-based methods. Results are reported overall and across six ALFWorld subtask categories under different frozen executors. Bold and underlined values indicate the best and second-best results, respectively.

Method	Overall	ALFWorld Subtasks					
		Put	Clean	Heat	Cool	Examine	PutTwo
Closed-source direct agents							
GPT-5.2	70.90	87.50	67.74	56.52	76.19	83.33	52.94
Gemini-3-flash	82.09	<u>91.67</u>	83.87	65.22	85.71	83.33	82.35
Executor: Qwen2.5-14B-Instruct							
Memory-based baselines							
No memory	46.27	33.33	51.61	26.09	85.71	72.22	5.88
Mem0	12.69	0.00	0.00	4.35	0.00	88.89	0.00
G-Memory	67.16	50.00	83.87	69.57	85.71	83.33	17.65
LangMem	36.57	12.50	32.26	26.09	71.43	83.33	0.00
Skill-based baselines							
Skill-Pro	64.18	87.50	64.52	56.52	52.38	<u>94.44</u>	23.53
XSkill	40.30	45.83	38.71	47.83	47.62	22.22	35.29
Trace2Skill	44.78	54.17	51.61	52.17	14.29	77.78	11.76
EmbodiSkill (Ours)							
EmbodiSkill w/ Qwen-14B	79.85	<u>91.67</u>	83.87	**82.61**	85.71	77.78	47.06
EmbodiSkill w/ GPT-5.2	86.57	<u>91.67</u>	<u>96.77</u>	<u>73.91</u>	**95.24**	83.33	70.59
EmbodiSkill w/ Gemini-3-flash	85.82	79.17	93.55	<u>73.91</u>	<u>90.48</u>	**100.00**	76.47
Executor: Qwen3.5-27B							
Memory-based baselines							
No memory	61.19	66.67	51.61	65.22	76.19	72.22	35.29
Mem0	64.18	62.50	61.29	56.52	85.71	72.22	47.06
G-Memory	74.63	62.50	77.42	**82.61**	85.71	83.33	52.94
LangMem	62.69	62.50	61.29	52.17	76.19	72.22	52.94
Skill-based baselines							
Skill-Pro	81.34	**95.83**	77.42	43.48	85.71	**100.00**	<u>94.12</u>
XSkill	55.97	58.33	51.61	47.83	57.14	55.56	70.59
Trace2Skill	64.18	79.17	77.42	0.00	76.19	66.67	88.24
EmbodiSkill (Ours)							
EmbodiSkill w/ Qwen-27B	86.57	<u>91.67</u>	**100.00**	**82.61**	85.71	72.22	76.47
EmbodiSkill w/ GPT-5.2	**93.28**	**95.83**	<u>96.77</u>	<u>73.91</u>	**95.24**	**100.00**	**100.00**
EmbodiSkill w/ Gemini-3-flash	<u>87.31</u>	**95.83**	93.55	69.57	**95.24**	83.33	82.35

GPT-5.2 and Gemini-3-flash used directly as task executors by 22.38 and 11.19 percentage points, respectively. Unlike direct closed-source agents, EmbodiSkill uses the closed-source model only for offline skill evolution; invocation costs are detailed in Appendix E. These results show that EmbodiSkill is effective when both the executor and evolution model are local, and can further benefit from a stronger offline evolution model without changing the executor. The subtask results further show that the gains span different household task types: the best configuration reaches 100% on Examine and PutTwo and exceeds 95% on Put, Clean, and Cool.

EmbodiedBench. EmbodiSkill also performs effectively on vision-driven embodied tasks. Under fully local and model-matched settings, it improves Qwen3-VL-8B from 24.00% to 42.67% on EB-Habitat and from 45.00% to 49.00% on EB-Navigation. With Qwen3-VL-32B, it improves direct execution from 34.67% to 51.33% and from 46.33% to 56.00% on the two tracks, respectively. EmbodiSkill outperforms all model-matched memory- and skill-based baselines across both executor scales, exceeding the strongest baseline by 13.67 and 2.00 percentage points with Qwen3-VL-8B, and by 4.33 and 4.00 points with Qwen3-VL-32B, on EB-Habitat and EB-Navigation, respectively. Using stronger evolution models provides additional gains in several settings, raising the best results to 45.33% and 50.33% with Qwen3-VL-8B, and to 52.33% and 61.33% with Qwen3-VL-32B. The best Qwen3-VL-32B configurations further surpass the strongest closed-source direct agents by 6.33 points on EB-Habitat and 4.00 points on EB-Navigation. Together, these results show that Em-

Table 2: Task success rate (%) on EB-Habitat and EB-Navigation.

Exec.	Method	EB-Hab.	EB-Nav.
GPT-5.2	Direct	40.00	57.33
Gemini-3-flash	Direct	46.00	56.00
Qwen3-VL-8B	Direct	24.00	45.00
	Mem0	20.33	46.33
	G-Memory	25.67	37.33
	LangMem	19.67	47.00
	Skill-Pro	15.33	43.33
	XSkill	28.67	43.00
	Trace2Skill	29.00	35.00
	Ours-Qwen	42.67	49.00
	Ours-GPT	45.33	50.33
	Ours-Gemini	41.00	49.00
Qwen3-VL-32B	Direct	34.67	46.33
	Mem0	38.33	52.00
	G-Memory	45.00	50.33
	LangMem	38.33	52.00
	Skill-Pro	26.67	45.33
	XSkill	34.33	39.00
	Trace2Skill	47.00	28.33
	Ours-Qwen	51.33	56.00
	Ours-GPT	50.33	**61.33**
	Ours-Gemini	**52.33**	**61.33**

bodiSkill remains effective across visual object interaction and navigation, and can further benefit from stronger offline evolution models while keeping the executor fixed.

4.3 Ablation Study

We evaluate the two central designs of EmbodiSkill on ALFWorld using Qwen2.5-14B as the frozen executor. Both ablations perform offline skill evolution and evaluate the resulting final skill with the frozen executor. *-w/o Skill-Aware Reflection* modifies the evolution procedure by directly evolving the skill from trajectories without reflection or failure attribution, while retaining the same evolution budget and settings. *-w/o Appendix* keeps the standard EmbodiSkill evolution procedure and removes only the appendix from the final evolved skill at evaluation time.

Table 3: Ablation study on ALFWorld with Qwen2.5-14B as the frozen executor and GPT-5.2 or Gemini-3-flash as the evolution model. *-w/o Skill-Aware Reflection* removes reflection-based failure attribution during evolution, while *-w/o Appendix* removes the appendix during evaluation.

Configuration	Evolution Model	
	GPT-5.2	Gemini-3-flash
EmbodiSkill	**86.57**	**85.82**
-w/o Skill-Aware Reflection	70.90	67.91
-w/o Appendix	76.87	78.36

As shown in Table 3, removing skill-aware reflection reduces task success from 86.57% to 70.90% with GPT-5.2 and from 85.82% to 67.91% with Gemini-3-flash. These consistent drops show that directly converting trajectories into skill updates is less effective than first identifying whether the observed evidence requires skill revision or emphasis. Removing only the appendix at evaluation time also decreases performance to 76.87% and 78.36%, respectively. Since this ablation uses the same evolved skill body and differs only in whether the appendix is provided during evaluation, these results demonstrate that explicitly emphasizing valid but previously overlooked guidance substantially improves its realization during execution.

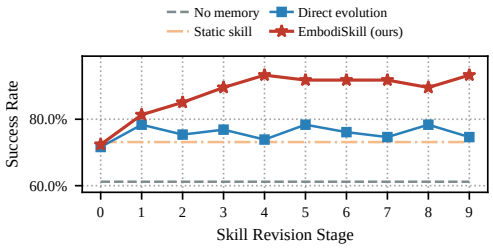


Figure 3: Test success across skill-evolution stages on ALFWorld, comparing no memory, a static skill, direct evolution, and EmbodiSkill.

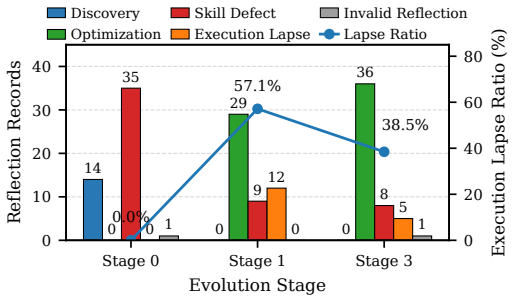


Figure 4: Reflection outcomes across representative evolution stages on ALFWorld with Qwen2.5-14B as both the executor and evolution model.

4.4 Analysis

Performance across Evolution Stages. Figure 3 shows how test performance changes throughout skill evolution. The no-memory and static-skill configurations remain fixed, serving as references for execution without external guidance and one-shot skill construction, respectively. Direct evolution obtains an initial improvement but subsequently fluctuates within a relatively low range. In contrast, EmbodiSkill achieves 72.39% task success after the first skill update (stage 0) and further improves to 93.28% over subsequent evolution stages, yielding an absolute gain of 20.89 percentage points. These results show that EmbodiSkill benefits not only from introducing an external skill, but also from continually refining and emphasizing it using newly accumulated interaction evidence.

Evolution of Reflection Outcomes. Figure 4 reports reflection outcomes on the ALFWorld evolution set with Qwen2.5-14B serving as both the executor and evolution model. An Invalid Reflection denotes an empty, malformed, or otherwise invalid reflection output that fails to satisfy the reflection requirements and is excluded from subsequent skill updates. Reflections from successful trajectories shift from 14 Discovery records at stage 0 to 29 and 36 Optimization records at stages 1 and 3, showing that evolution moves from acquiring new guidance to refining accumulated knowledge. Meanwhile, Skill Defect records from failed trajectories decrease from 35 to 9 and 8. Execution Lapse emerges after the skill is established, but subsequently decreases from 12 to 5 records, with its share among failed-trajectory reflections falling from 57.1% to 38.5%. These trends suggest that skill revision progressively reduces content defects, while emphasizing overlooked guidance is associated with fewer execution lapses in later stages. Together with Figure 3, these results provide a complementary view of the evolution process. Early stages primarily address missing or defective guidance, while later stages increasingly refine accumulated knowledge.

Qualitative Attribution Behavior. Appendix F provides representative cases of Discovery, Skill Defect, and Execution Lapse. These examples illustrate that the appropriate update depends on the relationship between the trajectory and the current skill, rather than on the trajectory outcome alone. In particular, the same failure outcome may require different updates depending on whether the current skill contains defective guidance or valid guidance that was overlooked during execution.

5 Conclusion

We introduced EmbodiSkill, a training-free framework for embodied skill self-evolution. It distinguishes skill-changing evidence from execution lapses, revising the skill body only when necessary and emphasizing valid but overlooked guidance in an appendix. Experiments on ALFWorld and EmbodiedBench show consistent gains over direct-execution, memory-based, and skill-based baselines, while ablations validate both components. These results highlight that reliable skill self-evolution requires deciding not only how to update skills, but also whether an update is needed.

AI USE STATEMENT

Generative AI tools were used to assist with editing the manuscript for readability and clarity. All AI-assisted content was reviewed and verified by the authors, who take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the algorithm and experimental setup to support reproducibility. Section 3 describes the EmbodiSkill framework and evolution procedure, while Section 4.1 specifies the benchmarks, models, evaluation protocol, and evolution settings. Appendix B details the construction of the EmbodiedBench evolution tasks, Appendix E reports closed-source model invocation costs, and Appendix G provides the core prompt templates used for skill reflection and evolution. An anonymized implementation is provided in the supplementary material.

REFERENCES

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

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.829. URL https://aclanthology.org/2026.findings-acl.829/.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, volume 37. Curran Associates, Inc., 2024. doi: 10.52202/079017-0019. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0142921fad7ef9192bd87229cdafa9d4-Abstract-Conference.html.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.866. URL https://aclanthology.org/2026.findings-acl.866/.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation, 2026. URL https://arxiv.org/abs/2603.22435.

Google DeepMind. Gemini 3 Flash model card. Model card, December 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. XSkill: Continual learning from experience and skills in multimodal agents. In *International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2603.12056. URL https://arxiv.org/abs/2603.12056. Accepted to ICML 2026. arXiv:2603.12056.

LangChain AI. LangMem. Software repository, 2025. URL https://github.com/langchain-ai/langmem. Accessed: 2026-05-04.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han-chung Lee, and Dawn Song. SkillsBench: Benchmarking how well agent skills work across diverse tasks, 2026. URL https://arxiv.org/abs/2602.12670.

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

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, 2026. URL https://arxiv.org/abs/2607.00272.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-Pro: Learning reusable skills from experience via non-parametric PPO for LLM agents, 2026. URL https://arxiv.org/abs/2602.01869. Accepted at ICML 2026 (spotlight).

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills, 2026. URL https://arxiv.org/abs/2603.25158.

OpenAI. Update to GPT-5 system card: GPT-5.2. OpenAI system card update, December 2025. URL https://openai.com/index/gpt-5-system-card-update-gpt-5-2/.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. VirtualHome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, June 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Puig_VirtualHome_Simulating_Household_CVPR_2018_paper.html.

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

Nikolai Rozanov and Marek Rei. StateAct: Enhancing LLM base agents via self-prompting and state-tracking. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 367–385, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.realm-1.27. URL https://aclanthology.org/2025.realm-1.27/.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339–9347, October 2019. URL https://openaccess.thecvf.com/content_ICCV_2019/html/Savva_Habitat_A_Platform_for_Embodied_AI_Research_ICCV_2019_paper.html.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10740–10749, June 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Shridhar_ALFRED_A_Benchmark_for_Interpreting_Grounded_Instructions_for_Everyday_Tasks_CVPR_2020_paper.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: program generation for situated robot task planning using large language models. *Autonomous Robots*, 47:999–1012, 2023. doi: 10.1007/s10514-023-10135-3. URL https://doi.org/10.1007/s10514-023-10135-3.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. LLM-Planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2998–3009, October 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Song_LLM-Planner_Few-Shot_Grounded_Planning_for_Embodied_Agents_with_Large_Language_ICCV_2023_paper.html.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang,

Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

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. PMLR, 2025. URL https://proceedings.mlr.press/v267/yang25f.html.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. AutoSkill: Experience-driven lifelong learning via skill self-evolution, 2026. URL https://arxiv.org/abs/2603.01145.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-Memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, pp. 12988–13018. Curran Associates, Inc., 2025. doi: 10.52202/085713-0439. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/136a45cd9b841bf785625709a19c6508-Abstract-Conference.html.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification, 2026a. URL https://arxiv.org/abs/2604.01687. Accepted at COLM 2026.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. MemSkill: Learning and evolving memory skills for self-evolving agents, 2026b. URL https://arxiv.org/abs/2602.02474.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

A ALGORITHM

Algorithm 1 summarizes the complete EmbodiSkill evolution procedure. At each evolution stage, the frozen executor collects B trajectories under the current skill, and the evolution model performs skill-aware reflection on each trajectory. The resulting reflection records are then separated by outcome: skill-changing evidence is reconciled and used to revise the skill body, while Execution Lapse evidence updates only the appendix. The updated skill is subsequently used to guide the next stage of interaction.

Algorithm 1 EMBODISKILL

Require: Initial skill $S^{(0)} = (S^{(0)}_{\text{body}}, S^{(0)}_{\text{app}})$; executor π_θ; evolution model F; task stream $\mathcal{I}$; stages N; update frequency B; reflection limit K
Ensure: Evolved skill $S^{(N)}$

for $n = 0, \dots, N-1$ **do**
  Collect and reflect
  $\mathcal{R}^{(n)} \leftarrow \varnothing$ ▷ Initialize stage records
  for $j = 1, \dots, B$ **do**
    $I_j \leftarrow \text{Next}(\mathcal{I})$
    $\tau_j \leftarrow \text{Execute}(\pi_\theta, I_j, S^{(n)})$
    $\mathcal{R}_{\tau_j} \leftarrow \text{Reflect}(F; \tau_j, S^{(n)}, K)$
    $\mathcal{R}^{(n)} \leftarrow \mathcal{R}^{(n)} \cup \mathcal{R}_{\tau_j}$
  end for
  Attribute and reconcile
  $(\mathcal{R}^{(n)}_{\text{disc}}, \mathcal{R}^{(n)}_{\text{opt}}, \mathcal{R}^{(n)}_{\text{def}}, \mathcal{R}^{(n)}_{\text{lap}}) \leftarrow \text{Split}(\mathcal{R}^{(n)})$ ▷ Attribute reflection outcomes
  $\mathcal{R}^{(n)}_{\text{chg}} \leftarrow \text{Reconcile}(F; \mathcal{R}^{(n)}_{\text{disc}}, \mathcal{R}^{(n)}_{\text{opt}}, \mathcal{R}^{(n)}_{\text{def}})$ ▷ Reconcile skill-changing evidence
  Update skill
  $(S^{(n+1)}_{\text{body}}, \mathcal{T}^{(n)}_{\text{body}}) \leftarrow \text{ReviseBody}(F; S^{(n)}_{\text{body}}, \mathcal{R}^{(n)}_{\text{chg}})$ ▷ Revise skill body
  $S^{(n+1)}_{\text{app}} \leftarrow \text{UpdateApp}(F; \mathcal{T}^{(n)}_{\text{body}}, S^{(n)}_{\text{app}}, \mathcal{R}^{(n)}_{\text{lap}})$ ▷ Emphasize overlooked guidance
  $S^{(n+1)} \leftarrow (S^{(n+1)}_{\text{body}}, S^{(n+1)}_{\text{app}})$
end for
return $S^{(N)}$

B EMBODIEDBENCH EVOLUTION-TASK CONSTRUCTION

We construct the evolution sets using the official task-generation pipelines released with EmbodiedBench.

EB-Habitat. For EB-Habitat, we use the benchmark-provided training-side task templates, language templates, object categories, and scene split, and instantiate tasks by sampling concrete objects and receptacles. The generated task configurations are then validated in Habitat-Sim, and only physically valid episodes are retained, resulting in 500 evolution tasks.

EB-Navigation. For EB-Navigation, we follow the benchmark-provided navigation-generation procedure. For each scene, a target object and its valid interaction location are identified, and a feasible initial agent pose is sampled at an appropriate distance from the target. The resulting scene, target object, initial pose, and instruction define a base navigation task. We further construct commonsense, complex, visual, and long-horizon variants following the provided instruction-generation procedures, resulting in 500 evolution tasks.

The generated tasks form the evolution set, while the official EmbodiedBench test set is used only for evaluation.

C COMPLETE RESULTS ON EMBODIEDBENCH

Table 4: Task success rates (%) on EmbodiedBench, reported overall and across capability-oriented subsets of EB-Habitat and EB-Navigation. Ours-Qwen uses the corresponding Qwen model for offline skill evolution, whereas Ours-GPT and Ours-Gemini use GPT-5.2 and Gemini-3-flash, respectively. Bold and underline denote the best and second-best results in each column, respectively.

Method	Overall	EB-Habitat						Overall	EB-Navigation				
		Base	Com.	Comp.	Vis.	Spa.	Long		Base	Com.	Comp.	Vis.	Long
Closed-source direct agents													
GPT-5.2	40.00	80	40	26	44	32	18	<u>57.33</u>	63.33	**71.67**	68.33	65.00	18.33
Gemini-3-flash	46.00	78	42	38	56	32	30	56.00	61.67	60.00	55.00	46.67	**56.67**
Executor: Qwen3-VL-8B-Instruct													
Memory-based baselines													
No memory	24.00	58	20	20	18	18	10	45.00	58.33	53.33	61.67	50.00	1.67
Mem0	20.33	60	16	8	16	14	8	46.33	58.33	56.67	61.67	55.00	0.00
G-Memory	25.67	58	14	12	34	30	6	37.33	61.67	43.33	43.33	38.33	0.00
LangMem	19.67	60	12	8	16	14	8	47.00	60.00	56.67	61.67	56.67	0.00
Skill-based baselines													
Skill-Pro	15.33	44	6	10	10	20	2	43.33	56.67	50.00	53.33	56.67	0.00
XSkill	28.67	74	18	26	22	22	10	43.00	50.00	56.67	53.33	55.00	0.00
Trace2Skill	29.00	74	14	28	30	14	14	35.00	45.00	41.67	43.33	43.33	1.67
EmbodiSkill (Ours)													
Ours-Qwen	42.67	82	42	34	48	<u>36</u>	14	49.00	**70.00**	61.67	58.33	55.00	0.00
Ours-GPT	45.33	76	**46**	36	52	**38**	24	50.33	<u>68.33</u>	65.00	61.67	56.67	0.00
Ours-Gemini	41.00	78	34	32	52	34	16	49.00	60.00	63.33	66.67	51.67	3.33
Executor: Qwen3-VL-32B-Instruct													
Memory-based baselines													
No memory	34.67	78	26	28	34	28	14	46.33	60.00	53.33	60.00	53.33	5.00
Mem0	38.33	84	24	26	42	34	20	52.00	61.67	60.00	65.00	<u>66.67</u>	6.67
G-Memory	45.00	<u>92</u>	24	36	50	32	<u>34</u>	50.33	51.67	53.33	58.33	65.00	23.33
LangMem	38.33	86	26	28	40	32	18	52.00	61.67	60.00	65.00	<u>66.67</u>	6.67
Skill-based baselines													
Skill-Pro	26.67	72	24	12	22	30	0	45.33	56.67	51.67	56.67	55.00	6.67
XSkill	34.33	78	26	26	34	32	10	39.00	45.00	43.33	48.33	41.67	16.67
Trace2Skill	47.00	88	32	44	50	26	**42**	28.33	40.00	28.33	40.00	31.67	1.67
EmbodiSkill (Ours)													
Ours-Qwen	<u>51.33</u>	**96**	34	48	<u>60</u>	<u>36</u>	<u>34</u>	56.00	65.00	58.33	63.33	63.33	30.00
Ours-GPT	50.33	<u>92</u>	**46**	<u>50</u>	54	34	26	**61.33**	65.00	68.33	**73.33**	<u>66.67</u>	<u>33.33</u>
Ours-Gemini	**52.33**	**96**	<u>44</u>	**52**	**62**	<u>36</u>	24	**61.33**	66.67	<u>70.00</u>	<u>70.00</u>	**68.33**	31.67

Capability-Oriented Subsets. EmbodiedBench evaluates complementary levels of embodied control. EB-Habitat requires high-level rearrangement planning over navigation and object-interaction skills, whereas EB-Navigation requires low-level translational and rotational control from egocentric visual observations. Within each environment, the benchmark further organizes tasks by the capability needed to interpret and execute the instruction. *Base* contains low- to medium-difficulty tasks expressed through direct instructions. *Common Sense* (**Com.**) refers to objects indirectly through everyday knowledge. *Complex Instruction* (**Comp.**) embeds the intended task in longer contexts that may include irrelevant information. *Visual Appearance* (**Vis.**) identifies objects through attributes such as color or shape, while *Spatial Awareness* (**Spa.**) refers to objects through their locations relative to other objects. Finally, *Long Horizon* (**Long**) requires extended action sequences and sustained state tracking. EB-Habitat contains 50 test instances in each of the six subsets. EB-Navigation contains 60 instances in each of five subsets and omits Spatial Awareness, yielding 300 test tasks for each environment.

Detailed Results on EB-Habitat. Table 4 shows that EmbodiSkill consistently improves both executor scales on high-level rearrangement tasks. When the corresponding Qwen model is also used for offline skill evolution, EmbodiSkill raises Qwen3-VL-8B from 24.00% without memory to 42.67%, a relative improvement of 77.8%. It also outperforms the strongest model-matched memory- and skill-based baselines by 66.2% and 47.1%, respectively. With Qwen3-VL-32B, EmbodiSkill achieves 51.33%, exceeding the no-memory executor by 48.1% and the strongest memory- and skill-based baselines by 14.1% and 9.2%. When GPT-5.2 and Gemini-3-flash are used only for offline skill evolution, EmbodiSkill with Qwen3-VL-32B achieves 50.33% and 52.33%, respectively, outperforming the settings in which GPT-5.2 and Gemini-3-flash directly serve as task executors by 25.8% and 13.8%.

The capability breakdown shows that the overall improvement is not driven solely by the Base subset. With Qwen3-VL-32B, the best EmbodiSkill configuration improves Common Sense from 26% to 44%, Complex Instruction from 28% to 52%, and Visual Appearance from 34% to 62%.

Detailed Results on EB-Navigation. Table 4 shows that EmbodiSkill brings modest gains with Qwen3-VL-8B and larger gains with Qwen3-VL-32B. In the fully local setting, it improves Qwen3-VL-8B from 45.00% without memory to 49.00%, while raising Qwen3-VL-32B from 46.33% to 56.00%, a relative improvement of 20.9%. With the 32B executor, it also exceeds the strongest model-matched memory- and skill-based baselines by 4.00 and 10.67 points, respectively. When GPT-5.2 and Gemini-3-flash are used only for offline skill evolution, the frozen Qwen3-VL-32B executor reaches 61.33% in both settings, outperforming the settings in which these models directly serve as task executors by 4.00 and 5.33 points, respectively.

The capability breakdown shows that the fully local 32B configuration improves all five subsets over the no-memory executor, with the largest gains appearing on Visual Appearance and Long Horizon at 10.00 and 25.00 points. Long Horizon nevertheless remains the main bottleneck. All EmbodiSkill configurations with Qwen3-VL-8B obtain only 0–3.33%, while Gemini-3-flash used directly reaches 56.67%, compared with 31.67% when it is used to evolve skills for Qwen3-VL-32B. These results show that skill evolution improves navigation across diverse capabilities, but cannot fully compensate for limitations in long-horizon perception and state tracking.

Taken together, the results on EB-Habitat and EB-Navigation show that EmbodiSkill improves frozen executors across both high-level rearrangement and low-level navigation. The broad gains across capability-oriented subsets are consistent with our skill-aware design, which revises deficient guidance while emphasizing valid but overlooked guidance.

D Sensitivity to Evolution Settings

Table 5: Sensitivity to the number of trajectories per update B and reflection limit K. The total evolution budget is fixed at 500 trajectories for all settings, with $N = 500/B$ evolution stages.

K	**Trajectories per Update** B	
	50 ($N = 10$)	**100** ($N = 5$)
1†	**79.85**	73.88
2	76.12	76.87
4	<u>78.36</u>	71.64

Table 5 studies the number of trajectories per update B and reflection limit K on ALFWorld using Qwen2.5-14B as both the executor and evolution model. We keep the total evolution budget fixed at 500 trajectories across all settings, so changing B only changes how often the skill is updated: $B = 50$ yields $N = 10$ evolution stages, whereas $B = 100$ yields $N = 5$. Performance remains relatively stable across most tested configurations, although the combination $B = 100$ and $K = 4$ leads to a noticeable decrease. Increasing K does not consistently improve performance, suggesting that additional reflections may introduce redundant or less useful evidence rather than provide further gains. Similarly, less frequent updates do not offer a consistent advantage. Overall, the default

setting $B = 50$ and $K = 1$ achieves the highest success rate of 79.85%, while the results indicate reasonable robustness within the tested range.

E Closed-Source Model Invocation Cost

When GPT-5.2 or Gemini-3-flash is used for skill evolution, the closed-source model is invoked only during offline evolution. In each evolution stage, EmbodiSkill collects B trajectories and invokes the evolution model once to reflect on each trajectory, followed by three additional calls for reflection reconciliation, body revision, and appendix update. Therefore, the closed-source model invocation cost of one complete skill-evolution run is

$$C_{\text{evo}} = N(B + 3).$$

Under our default setting with $B = 50$ and $N = 10$, one complete skill-evolution run requires 530 closed-source model calls. After evolution, the skill is frozen and test-time execution is performed entirely by the local executor, requiring no additional closed-source model calls.

For comparison, direct closed-source agents invoke the model repeatedly throughout task execution. Prior work on ALFWorld reports average interaction lengths of 19.11–31.49 steps per task (Rozanov & Rei, 2025), corresponding to approximately 2,561–4,220 model invocations over the 134 test tasks. EmbodiedBench reports 4.2–7.4 average planner steps per task on EB-Habitat and 6.2–13.2 on EB-Navigation across evaluated models (Yang et al., 2025). Over the 300 test tasks in each track, these correspond to approximately 1,260–2,220 and 1,860–3,960 model invocations, respectively. Table 6 summarizes the comparison.

Based on these published step statistics, direct closed-source execution corresponds to approximately 4.8–$8.0\times$ more model invocations on ALFWorld, 2.4–$4.2\times$ more on EB-Habitat, and 3.5–$7.5\times$ more on EB-Navigation than EmbodiSkill's one-time offline evolution cost.

Table 6: Closed-source model invocation cost. Offline cost is reported for one complete evolution run under our default setting. Online costs for direct agents are reference estimates based on average interaction or planner steps reported in prior work.

	Offline	**Online execution**		
Method	**Evolution**	**ALFWorld**	**EB-Habitat**	**EB-Navigation**
Direct GPT/Gemini	0	2,561–4,220	1,260–2,220	1,860–3,960
EmbodiSkill-GPT/Gemini	530	0	0	0

F Case Studies

We use three representative trajectories—a discovery, a skill defect, and an execution lapse—to illustrate how skill-aware reflection maps interaction evidence to different update operations and determines whether the skill body or appendix should be updated. The examples show that an update depends on the relationship between a trajectory and the current skill, rather than on the trajectory outcome alone.

Discovery Figure 5 shows a successful trajectory in which the agent's forward motion is blocked near the target, and a single lateral step brings it sufficiently close to complete the task. Because the current skill lacks guidance for this near-goal situation, skill-aware reflection classifies the evidence as Discovery and adds the observed strategy to the skill body.

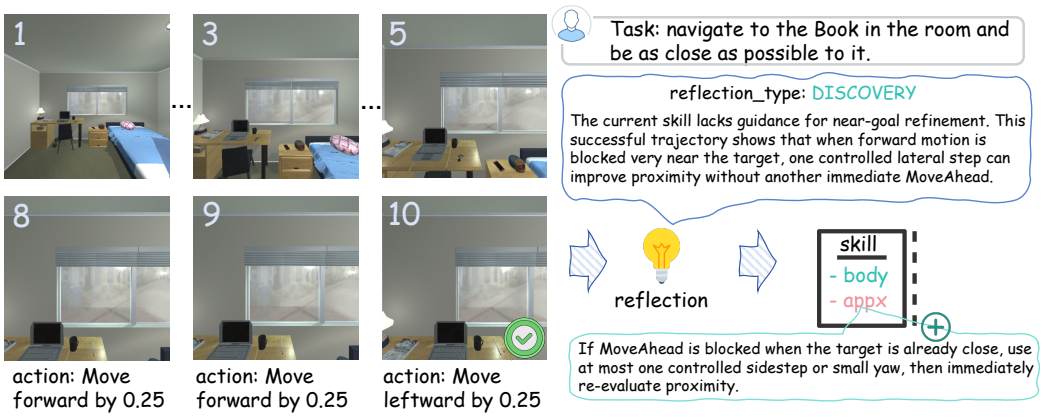


Figure 5: Case study of Discovery. A controlled lateral step resolves a near-goal blockage and completes the task, revealing missing guidance for near-goal refinement. EmbodiSkill generalizes this successful strategy and adds it to the skill body.

Skill Defect Figure 6 shows a failed trajectory in which the agent repeatedly executes `MoveAhead` while the same small obstacle blocks its path. The current skill incorrectly recommends making minor pose adjustments and continuing to retry the forward action, which keeps the agent in the same collision pattern. Skill-aware reflection therefore classifies the evidence as a Skill Defect and corrects the skill body to require a committed turn-and-bypass maneuver after repeated blockage, followed by target reacquisition before resuming the approach.

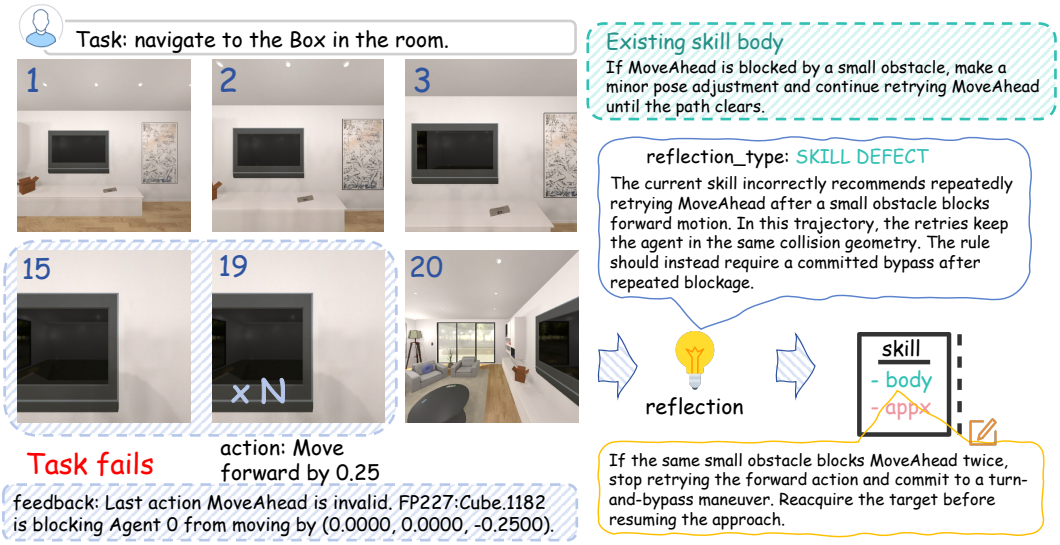


Figure 6: Case study of a Skill Defect. The existing skill repeatedly directs the agent to retry `MoveAhead` after minor pose adjustments, leaving it trapped against the same obstacle. EmbodiSkill corrects the skill body so that, after repeated blockage, the agent stops retrying the same action, switches to a more decisive bypass maneuver, and reacquires the target before continuing.

Execution Lapse Figure 7 shows a failed trajectory in which the agent repeatedly issues invalid opening and picking actions instead of re-localizing and changing its search strategy. Because the existing skill body already instructs the agent not to repeat invalid actions and to switch to landmark-based navigation and structured search, skill-aware reflection classifies the failure as an Execution Lapse. EmbodiSkill therefore preserves the skill body and updates only the appendix to emphasize this applicable but overlooked guidance when attempting to open or pick an object.

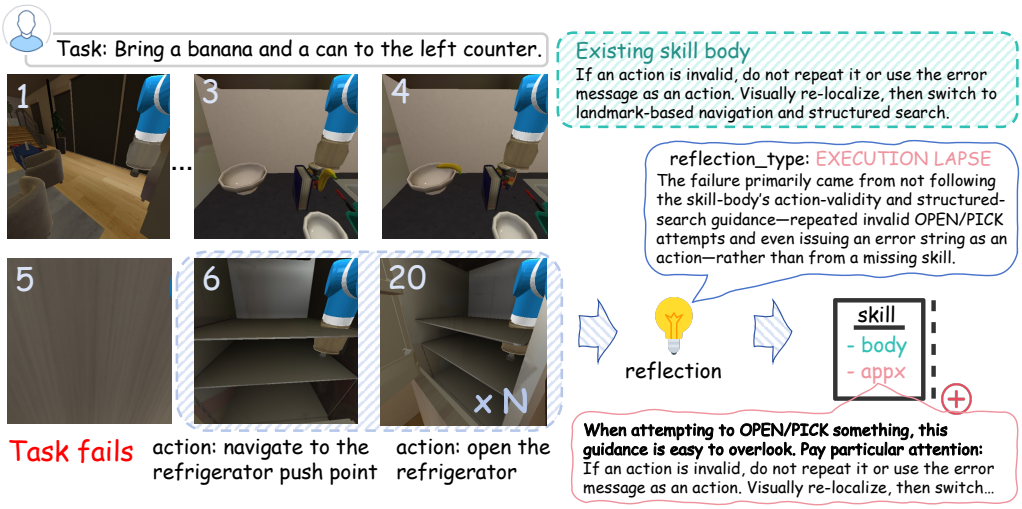


Figure 7: Case study of an Execution Lapse. Although the existing skill body already instructs the agent not to repeat invalid actions and to re-localize before switching search strategies, the executor repeatedly issues ineffective `OPEN` actions. EmbodiSkill preserves the skill body and emphasizes the overlooked guidance in the appendix.

Summary Across these cases, skill-aware reflection selects an update by comparing trajectory evidence with the current skill rather than by outcome alone. Discovery, Optimization, and Skill Defect indicate a need to add, refine, or correct the skill body, whereas an Execution Lapse preserves the body and updates only the appendix to emphasize overlooked guidance. This distinction is central to EmbodiSkill: reliable self-evolution requires deciding not only how to update a skill, but whether it should be changed at all.

G Core Prompt Templates

We provide the core prompt templates used in EmbodiSkill for skill-aware reflection, reflection reconciliation, skill-body revision, and skill-appendix update. These prompts implement the reflection and update operations described in Section 3.

Skill-Aware Reflection Prompt

You are given a task instruction, the current skill, and an interaction trajectory with its final outcome.
Your goal is to determine whether the trajectory provides useful evidence for skill evolution. It is acceptable for a trajectory to produce no reflection. When useful evidence exists, classify it into one of the following outcomes.
DISCOVERY. The trajectory succeeds and reveals reusable procedural guidance that is absent from the current skill. Identify the missing guidance and propose adding it to the skill body.
OPTIMIZATION. The trajectory succeeds and demonstrates a more effective procedure for guidance that is already valid in the current skill. Identify the corresponding existing guidance and propose a targeted refinement.
SKILL DEFECT. The trajectory fails because the relevant guidance in the current skill is missing, incorrect, incomplete, or underspecified. Identify the defective or missing guidance and propose a targeted correction.
EXECUTION LAPSE. The trajectory fails even though valid task-relevant guidance is already present in the current skill. Use this outcome only when the failure could have been avoided by following or correctly applying that existing guidance. Identify the exact existing guidance that was overlooked and propose an execution reminder rather than a revision to the skill body.
General rules:

- Reflections should capture high-level, reusable procedural knowledge rather than trajectory-specific details.
- Do not create a reflection when the trajectory provides no meaningful reusable evidence.
- Each reflection should target a distinct missing or existing piece of skill guidance.
- When referring to existing guidance, identify the exact corresponding skill content.
- Keep the proposed update concise and directly actionable.
- Return at most K distinct reflections, ordered by importance.

Reflection Reconciliation

You are given a set of skill-changing reflections classified as DISCOVERY, OPTIMIZATION, or SKILL DEFECT.
Reconcile these reflections before revising the skill body.
Rules:

- Group reflections that concern the same or closely related skill guidance.
- Remove duplicate and near-duplicate reflections.
- Merge compatible reflections that express the same reusable lesson into one concise update.
- When multiple reflections target the same existing guidance, resolve them jointly rather than applying them independently.
- Check proposed additions against existing guidance and avoid adding content that is already covered.
- If multiple reflections conflict, reconcile them when a consistent interpretation is clear.
- If conflicting reflections cannot be reconciled reliably, discard the conflicting updates rather than introducing inconsistent guidance.
- Preserve only non-redundant and mutually consistent skill-changing evidence for subsequent body revision.

Return the reconciled set of skill-changing reflections.

Skill Body Revision Prompt

You are given the current skill body and a set of reflections classified as DISCOVERY, OPTIMIZATION, or SKILL DEFECT.
Your goal is to produce the next version of the skill body.
Revision rules:

- Use DISCOVERY reflections to add reusable guidance that is missing from the current skill.
- Use OPTIMIZATION reflections to refine the targeted existing guidance while preserving its valid content.
- Use SKILL DEFECT reflections to correct incorrect, incomplete, or underspecified guidance, or to add guidance that is missing and caused a failure.
- Treat each reflection as a targeted edit. Preserve skill content that is unrelated to the reflected evidence.
- Before revision, remove redundant reflections and jointly resolve reflections that target the same or overlapping guidance.
- Merge compatible reflections into a single concise update.
- If reflections conflict and cannot be reconciled reliably, do not introduce the conflicting changes.
- Avoid introducing duplicate or near-duplicate rules.
- Keep the resulting skill concise, high-level, and broadly reusable.
- Do not incorporate EXECUTION LAPSE reflections into the skill body.

Return only the revised skill body.

Skill Appendix Update Prompt

You are given the revised skill body, the current skill appendix, and a set of EXECUTION LAPSE reflections.
Your goal is to update the appendix so that valid but previously overlooked guidance is more salient during future execution.
Update rules:

- Each appendix item must emphasize guidance that already exists in the current skill body.
- Every appendix item must be explicitly associated with the existing body guidance that it emphasizes.
- Do not introduce new procedural knowledge or new skill rules in the appendix.
- Do not modify the skill body.
- If the targeted body guidance was revised during the current update, align the reminder with its revised form.
- If the targeted guidance no longer exists and no clear corresponding guidance remains, discard the reminder.
- Maintain at most one reminder for the same body guidance.
- Merge redundant or overlapping reminders and resolve conflicts when possible.
- Preserve unrelated existing appendix items.
- Keep each reminder concise, direct, and actionable.

Return only the updated skill appendix.