EmbodiSkill: Skill-Aware Reflection for Self-Evolving Embodied Agents

Ruofei Ju[1*†] **Xinrui Wang**[2*†] **Xin Ding**[3] **Yifan Yang**[4] **Hao Wu**[1] **Shiqi Jiang**[4] **Qianxi Zhang**[4]
Hao Wen[5] **Xiangyu Li**[5] **Weijun Wang**[5] **Kun Li**[5] **Yunxin Liu**[5] **Haipeng Dai**[1] **Wei Wang**[1] **Ting Cao**[5‡]

[1] Nanjing University [2] Huazhong University of Science and Technology [3] University of Science and Technology of China
[4] Microsoft Research [5] Institute for AI Industry Research (AIR), Tsinghua University

Embodied agents can benefit from skills that guide object search, action execution, and state changes across diverse environments. Since embodied environments vary across layouts, object states, and other execution factors, these skills must self-evolve from trajectories generated during task execution. However, existing skill self-evolution methods are mainly developed in digital environments and often convert trajectories into coarse skill updates. Directly applying this paradigm to embodied settings is problematic, because a failed task execution may reflect not only incorrect skill content, but also an execution lapse in which the agent fails to follow valid guidance. We propose EmbodiSkill, a training-free framework for embodied skill self-evolution through skill-aware reflection and targeted revision. EmbodiSkill interprets each trajectory with respect to the current skill, uses skill-changing evidence to update the skill body, and uses execution-lapse evidence to preserve and emphasize valid guidance. Experiments on ALFWorld and EmbodiedBench show that EmbodiSkill consistently improves embodied task success. On ALFWorld, EmbodiSkill enables a frozen Qwen3.5-27B executor to reach 93.28% task success, outperforming GPT-5.2 used as a direct agent without skills by 31.58%. These results show that skill-aware self-evolution helps embodied agents accumulate reusable procedural knowledge from their own trajectories.

1. Introduction

Embodied agents are expected to complete household-like tasks in physical or physically grounded 3D environments, where they must observe the scene, navigate through space, interact with objects, satisfy action preconditions, and handle failed actions [1, 2, 3]. In such tasks, the trajectories generated by different task executions often exhibit recurring procedural structures: an agent may need to locate an object before manipulating it, check whether a container is open before placing it, or adjust its viewpoint when the target is not visible. Following prior work that treats skills, action primitives, or programmatic policies as reusable procedural units for grounding high-level instructions into executable behavior [4, 5, 6, 7], we define an *embodied skill* as a persistent and revisable procedural specification that guides an embodied agent across task executions. In embodied tasks, a skill specifies reusable guidance such as prerequisites, subgoal ordering, object affordances, visual-search strategies, action preconditions, and recovery strategies. Since embodied environments vary in layouts, object states, visibility conditions, and feasible action sequences, an initial skill cannot cover all situations the agent may encounter. Therefore, the central challenge is not only how an embodied agent uses a skill, but how the skill can self-evolve from trajectories into a more complete, accurate, and executable form.

Recent skill self-evolution methods show that agents can improve by extracting, revising, and reusing skills from trajectories [7, 8, 9, 10, 11, 12]. However, many existing methods update skills at a coarse granularity: a trajectory is summarized into feedback, and the resulting feedback is used to coarsely

[*]Ruofei Ju and Xinrui Wang contributed equally to this work.
[†]This work was done while Ruofei Ju and Xinrui Wang were interns at the Institute for AI Industry Research (AIR), Tsinghua University.
[‡]Corresponding author tingcao@mail.tsinghua.edu.cn.

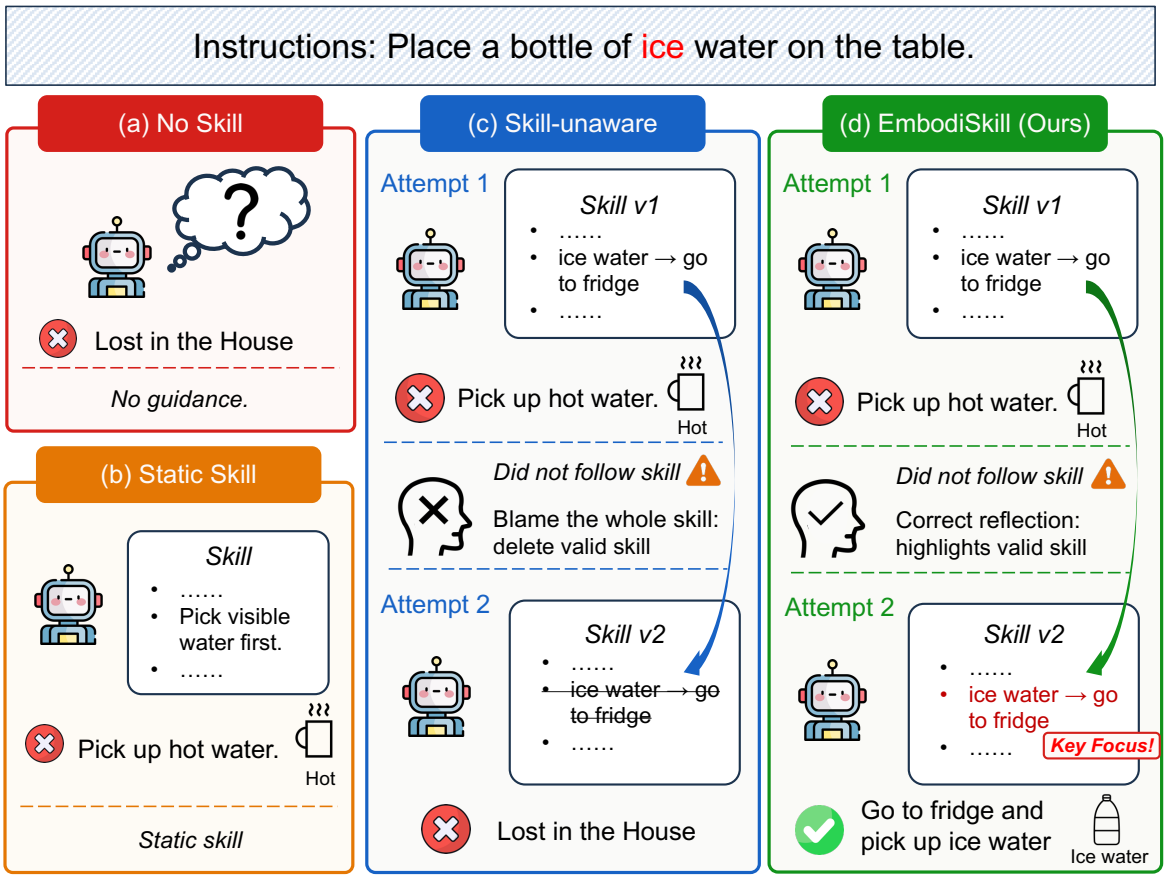

Figure 1 | Motivating example of EmbodiSkill. For the same task instruction, (a) no skill leaves the agent without procedural guidance and causes inefficient exploration, (b) a static skill provides incomplete guidance and cannot adapt from trajectories, and (c) skill-unaware evolution may misinterpret an execution lapse as a skill defect and wrongly revise valid skill content. In contrast, (d) EmbodiSkill reflects on the trajectory against the current skill, preserves valid guidance, and updates the skill to improve later task executions.

update the whole skill, without explicitly identifying which skill content is implicated, why it should be changed, and how it should be changed. This update strategy becomes problematic in embodied environments, where a trajectory outcome is entangled with perception, spatial grounding, object states, action preconditions, and execution reliability [1, 2, 3, 4, 13]. A successful trajectory may reveal useful new skill content or a better way to perform an existing skill, but a failed trajectory does not necessarily mean that the current skill is wrong. It may instead result from the agent failing to follow valid skill content [4, 13, 2]. If such trajectories are converted into coarse skill updates, the evolving skill may accumulate redundant content, overwrite valid guidance, or preserve incorrect prescriptions. Figure 1 illustrates this problem with a motivating example, where skill-unaware evolution misinterprets a failed task execution and incorrectly revises valid skill content.

To address these problems, we propose EmbodiSkill, a framework for embodied skill self-evolution through skill-aware reflection and a skill-aware evolution spiral. Given a trajectory and the current skill, EmbodiSkill does not convert the trajectory into general feedback for rewriting the whole skill. Instead,

skill-aware reflection uses the trajectory to examine the current skill and determine which skill content should be added, optimized, corrected, or preserved. A trajectory can lead to DISCOVERY when it reveals missing skill content, OPTIMIZATION when it suggests a better way to perform an existing skill, SKILL DEFECT when it exposes incorrect, incomplete, or underspecified skill content, or EXECUTION LAPSE when the skill is valid but the agent fails to execute it. The resulting reflections serve as targeted update signals for skill revision. In the skill-aware evolution spiral, justified reflections are integrated into the next skill version, and the revised skill is then used to guide subsequent task executions that generate new trajectories. This creates a closed loop in which trajectories refine the skill, and the refined skill guides later task executions that produce further trajectories for skill evolution. By separating these cases, EmbodiSkill revises only the skill content that should change while preserving valid skill content from unnecessary updates. This selective revision process allows the skill to evolve progressively: each trajectory contributes targeted evidence, and the skill becomes increasingly complete, accurate, and executable over time.

We evaluate EmbodiSkill on embodied and interactive task benchmarks covering visual object interaction, navigation, and household task completion [2, 3]. On ALFWorld, EmbodiSkill enables a local open-weight Qwen3.5-27B executor to achieve 93.28% task success, outperforming GPT-5.2 used as a direct agent by 31.58%. It also exceeds the representative memory-based baseline G-Memory [14] by 25.01%. More importantly, compared with the skill-unaware variant using the same skill evolution model, EmbodiSkill brings a 19.04% relative improvement, demonstrating the importance of skill-aware reflection. These results show that selectively updating the implicated skill content is more effective than coarsely revising the skill, allowing the skill to become progressively more complete, accurate, and executable.

2. Related Work

2.1. Memory-Based Methods for Embodied Agents

In embodied environments, agents often face recurring objects, spatial layouts, action constraints, and failure patterns across task executions, which are reflected in the resulting trajectories. A common way to exploit such recurrence is to adapt memory-based agent methods, which store past trajectories, summaries, or structured notes, and retrieve relevant memory to guide later decision making. For example, Reflexion stores verbal feedback from previous trials as episodic memory [15], while ExpeL extracts reusable lessons from successful and failed trajectories [16]. Recent memory systems such as Mem0, G-Memory, A-MEM, and LangMem further organize agent memory for long-term reuse [17, 14, 18, 19]. However, these methods mainly support trajectory-level reuse rather than skill-level guidance. Retrieved memory records what happened before, but it may remain fragmented or trajectory-specific, rather than forming a persistent skill that specifies how the agent should act in future task executions. This motivates moving beyond retrieved memory toward skills that provide higher-level, persistent, and procedural guidance for future embodied task execution.

2.2. Skills and Skill Self-Evolution for Agents

This higher-level procedural guidance is commonly instantiated as skills. In robotics and physically grounded task settings, prior work often relies on reusable skills, action primitives, or programmatic policies to connect high-level instructions with executable actions. SayCan selects feasible robotic skills by

combining language-model priors with affordance estimates [4]; Code as Policies generates executable policy code by composing perception outputs and control primitives [5]; and ProgPrompt represents household task plans as program-like structures for situated execution [6]. These methods demonstrate the value of skills for embodied task execution, but they mainly focus on selecting, composing, or executing existing skills rather than revising the skill itself from trajectories.

Recent work further studies skill self-evolution and procedural-memory evolution, where agents extract, refine, and reuse skills or reusable procedural knowledge from trajectories and interaction experience. Voyager builds an executable skill library through exploration in an interactive digital world [7]. Trace2Skill distills trajectory-local lessons into transferable agent skills through large-scale trajectory analysis and hierarchical consolidation [10], while EvoSkills studies self-evolving multi-file skill packages through co-evolutionary verification [11]. Other recent work explores related directions such as recursive skill construction, experience-derived executable skills, memory-skill evolution, procedural-memory refinement, and multimodal skill accumulation [20, 12, 21, 8, 9, 22]. These works show that self-evolving skills are a promising paradigm for long-term agent improvement.

However, most existing skill self-evolution methods are not designed for physically grounded embodied environments, where trajectory outcomes are affected by perception, spatial grounding, object states, action preconditions, and execution reliability. As a result, coarse whole-skill update strategies may add redundant content, overwrite valid skill content, or preserve incorrect skill content. EmbodiSkill addresses this gap by making skill self-evolution skill-aware: each trajectory is used to produce targeted revision signals, and justified revisions are integrated through a skill-aware evolution spiral rather than rewriting the skill as a whole.

3. Method

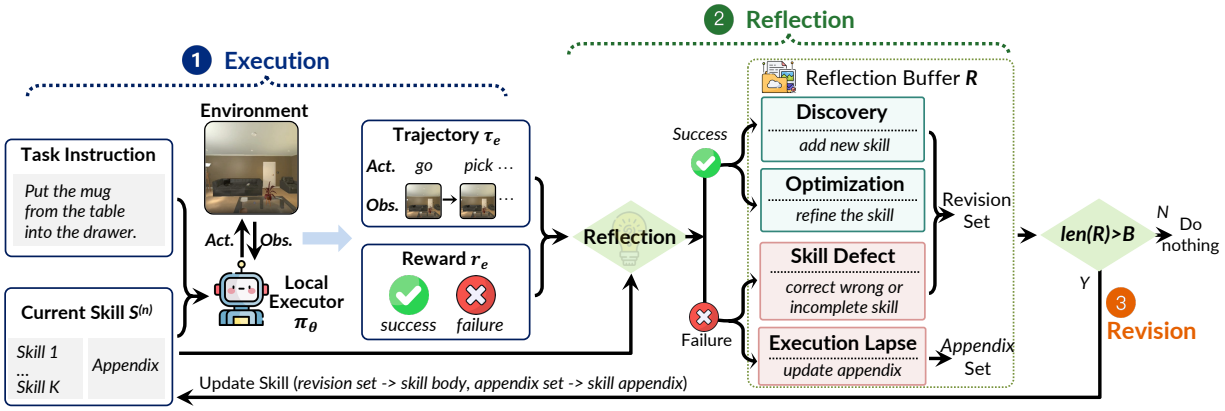

Figure 2 | Overview of EmbodiSkill. The executor uses the current skill to perform embodied tasks and generate trajectories. Skill-aware reflection uses each trajectory to produce targeted reflection records. Accumulated reflections are consolidated into body-level revisions and skill-appendix updates, forming the next skill version. The revised skill then guides subsequent task execution, creating a Skill-Aware Evolution Spiral.

3.1. Problem Formulation

Embodied trajectory. We consider an embodied agent interacting with a physical or physically grounded environment under natural-language task instructions. For a task instruction I, the agent observes the environment and produces actions over a finite horizon. We denote the resulting trajectory as

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

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

Skill-guided execution. At evolution step n, the agent is guided by a skill $S^{(n)}$, a persistent and revisable procedural specification for embodied task execution. We represent the skill as

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

where $S^{(n)}_{\text{body}}$ contains the main prescriptive skill content for future task execution, and $S^{(n)}_{\text{app}}$ is a skill appendix that emphasizes valid content in $S^{(n)}_{\text{body}}$. The appendix does not introduce new skill rules; it only highlights existing skill content that should be followed carefully during execution.

Given the current skill, the executor π_θ generates actions according to

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

where $h_t = (o_1, a_1, \ldots, o_t)$ denotes the within-trajectory history. The executor parameters θ are kept fixed; all improvement across task executions is externalized into the evolving skill.

Skill evolution model. The executor π_θ uses the current skill to perform embodied tasks and produce trajectories. In contrast, the skill evolution model F operates after trajectories are completed. We use F as a unified notation for the model calls used in skill-aware reflection, reflection consolidation, skill-body revision, and skill-appendix update; these calls share the same underlying model but use different prompts and output formats.

Skill self-evolution objective. The goal of embodied skill self-evolution is to construct a sequence of skills

$$S^{(0)} \to S^{(1)} \to \cdots \to S^{(N)} \tag{4}$$

such that later skills lead to higher task success when used by the same executor. Formally, we aim to improve

$$J(S; \pi_\theta) = \mathbb{E}_{I,\mathcal{E}}\left[r(\tau)\right], \quad \tau = \text{Execute}(\pi_\theta, I, \mathcal{E}, S), \tag{5}$$

where $\mathcal{E}$ denotes the embodied environment distribution.

Algorithm 1: EmbodiSkill

Input: Initial skill $S^{(0)} = (S^{(0)}_{\text{body}}, S^{(0)}_{\text{app}})$, executor π_θ, skill evolution model F, task stream $\mathcal{I}$, revision interval B, maximum reflections per trajectory K
Output: Evolved skill S

$(S_{\text{body}}, S_{\text{app}}) \leftarrow S^{(0)}$;
$S \leftarrow (S_{\text{body}}, S_{\text{app}})$;
$\mathcal{R} \leftarrow \emptyset$; // reflection buffer
foreach *task instruction* $I \in \mathcal{I}$ **do**
  $\tau \leftarrow \textsc{Execute}(\pi_\theta, I, S)$;
  $\mathcal{R}_\tau \leftarrow \textsc{SkillAwareReflect}(F, \tau, S, K)$;
  if $\mathcal{R}_\tau \neq \emptyset$ **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_\tau$;
  if $|\mathcal{R}| \geq B$ **then**
    $(\mathcal{R}_{\text{disc}}, \mathcal{R}_{\text{opt}}, \mathcal{R}_{\text{def}}, \mathcal{R}_{\text{lap}}) \leftarrow \textsc{PartitionByType}(\mathcal{R})$;
    $\widetilde{\mathcal{R}}_{\text{rev}} \leftarrow \textsc{ConsolidateRevisions}(F, S_{\text{body}}, \mathcal{R}_{\text{disc}}, \mathcal{R}_{\text{opt}}, \mathcal{R}_{\text{def}})$;
    $S_{\text{body}} \leftarrow \textsc{ReviseSkillBody}(F, S_{\text{body}}, \widetilde{\mathcal{R}}_{\text{rev}})$; // targeted body edits
    $S_{\text{app}} \leftarrow \textsc{UpdateSkillAppendix}(F, S_{\text{body}}, S_{\text{app}}, \mathcal{R}_{\text{lap}})$; // update appendix only
    $S \leftarrow (S_{\text{body}}, S_{\text{app}})$;
    $\mathcal{R} \leftarrow \emptyset$;
return S;

3.2. EmbodiSkill

EmbodiSkill implements embodied skill self-evolution as a skill-aware reflection and revision loop, as shown in Figure 2 and summarized in Algorithm 1. At evolution step n, the executor uses the current skill $S^{(n)}$ to perform embodied tasks, and each task execution produces a trajectory. After each trajectory, a skill evolution model uses the resulting trajectory to examine $S^{(n)}$ and produces up to K targeted reflections, or no reflection when the trajectory does not provide reliable evidence for skill revision. Each valid reflection specifies the trajectory evidence, the corresponding revision directive, and either the implicated existing skill content or newly proposed skill content.

After a fixed number of valid reflections are accumulated, EmbodiSkill consolidates them before revising the skill. Reflections that propose body-level updates are consolidated to remove redundancy and resolve conflicts before updating S_{body}. Reflections attributed to execution lapses do not revise the skill body; instead, after S_{body} is updated, they are used together with the updated skill body and the previous skill appendix to produce a new S_{app}. The final revision step therefore yields the next skill version $S^{(n+1)} = (S^{(n+1)}_{\text{body}}, S^{(n+1)}_{\text{app}})$, where substantive changes to the skill body are grounded in skill-aware revision signals, while the skill appendix highlights valid skill content that should be followed carefully during later execution. The revised skill then guides subsequent task execution, forming a Skill-Aware Evolution Spiral.

3.2.1. Skill-Aware Reflection

Skill-aware reflection determines how a trajectory should be interpreted with respect to the current skill. Given a trajectory τ and the current skill $S^{(n)} = (S^{(n)}_{\text{body}}, S^{(n)}_{\text{app}})$, the skill evolution model produces a set of reflection records:

$$\mathcal{R}_\tau = F(\tau, S^{(n)}, K) = \{\rho_i\}_{i=1}^{m_\tau}, \quad 0 \le m_\tau \le K, \tag{6}$$

where K is the maximum number of reflections allowed for one trajectory. When the trajectory does not provide reliable evidence for skill revision, $m_\tau = 0$ and no reflection is produced.

Each reflection record ρ_i contains a reflection type c_i, trajectory evidence e_i, and an update directive d_i. For reflection types that modify or emphasize existing skill content, the record also contains a target skill content b_i that refers to a specific part of $S^{(n)}_{\text{body}}$. The reflection type is chosen according to the trajectory outcome. For a successful trajectory, the reflection type is chosen from

$$c_i \in \{\textsc{Discovery}, \textsc{Optimization}\}, \quad r = 1, \tag{7a}$$

whereas for a failed trajectory, the reflection type is chosen from

$$c_i \in \{\textsc{SkillDefect}, \textsc{ExecutionLapse}\}, \quad r = 0. \tag{7b}$$

DISCOVERY. A Discovery reflection indicates that the trajectory reveals useful skill content not covered by the current skill body. It does not require a target skill content, because it proposes new skill content rather than modifying existing content. Its directive d_i specifies the new skill content to be considered during revision.

OPTIMIZATION. An Optimization reflection indicates that an existing target skill content is valid, but the trajectory suggests a more effective way to perform it. It must identify a target skill content b_i and provide a revised version or modification directive for that target skill content.

SKILL DEFECT. A SkillDefect reflection indicates that an existing target skill content is incorrect, incomplete, or underspecified. It must identify the problematic target skill content b_i, provide trajectory evidence for the defect, and propose corrected skill content.

EXECUTION LAPSE. An ExecutionLapse reflection indicates that the target skill content is valid, but the executor fails to follow it in the trajectory. It must identify the valid target skill content b_i and describe the deviation between what the skill requires and what the executor actually did. Its directive d_i is not a body-level revision; instead, it produces appendix content for S_{app}, reminding the executor to follow the corresponding valid skill content carefully during later task execution.

Thus, skill-aware reflection differs from trajectory summarization. It does not convert a trajectory into general feedback for rewriting the whole skill. Instead, it uses the trajectory to determine whether the trajectory supports adding new skill content, optimizing or correcting existing skill content, or updating the skill appendix, and produces structured signals for the revision procedure described next.

3.2.2. Skill Revision

Skill revision converts accumulated skill-aware reflections into the next skill version. Once the reflection buffer reaches the revision interval B, EmbodiSkill partitions the buffered reflections by type:

$$(\mathcal{R}_{\text{disc}}, \mathcal{R}_{\text{opt}}, \mathcal{R}_{\text{def}}, \mathcal{R}_{\text{lap}}) = \textsc{PartitionByType}(\mathcal{R}), \tag{8}$$

where $\mathcal{R}_{\text{disc}}$, $\mathcal{R}_{\text{opt}}$, $\mathcal{R}_{\text{def}}$, and $\mathcal{R}_{\text{lap}}$ correspond to Discovery, Optimization, SkillDefect, and ExecutionLapse, respectively.

Consolidating body-level revision signals. The skill body is revised only from Discovery, Optimization, and SkillDefect reflections. Before editing the skill body, EmbodiSkill consolidates these reflections:

$$\widetilde{\mathcal{R}}_{\text{rev}} = F\left(S_{\text{body}}^{(n)}, \mathcal{R}_{\text{disc}}, \mathcal{R}_{\text{opt}}, \mathcal{R}_{\text{def}}\right). \tag{9}$$

This consolidation step turns multiple local reflections into a consistent set of revision signals. It removes redundant reflections, merges overlapping suggestions, groups target-specific modifications by their target skill contents, and resolves conflicts among signals when possible. If a signal is misplaced or conflicts cannot be reliably resolved, the model either reassigns it to a more appropriate revision type or discards it, rather than forcing uncertain changes into the skill.

Revising the skill body. The consolidated revision signals are then used to update the skill body:

$$S_{\text{body}}^{(n+1)} = F\left(S_{\text{body}}^{(n)}, \widetilde{\mathcal{R}}_{\text{rev}}\right). \tag{10}$$

This step uses the skill evolution model as a constrained editor rather than a free-form rewriter. The editor revises the skill body according to the consolidated revision set $\widetilde{\mathcal{R}}_{\text{rev}}$: Discovery adds new skill content, while Optimization and SkillDefect modify their identified target skill contents. Skill content not implicated by $\widetilde{\mathcal{R}}_{\text{rev}}$ is kept unchanged in substance. The editor may still perform limited consistency edits, such as removing redundancy, normalizing format, or resolving local conflicts among the affected skill content. Thus, the revision updates targeted skill content while avoiding coarse rewriting of the whole skill.

Updating the skill appendix. After the skill body has been revised, EmbodiSkill updates the appendix part using ExecutionLapse reflections:

$$S_{\text{app}}^{(n+1)} = F\left(S_{\text{body}}^{(n+1)}, S_{\text{app}}^{(n)}, \mathcal{R}_{\text{lap}}\right). \tag{11}$$

Unlike the body-level revision step, this update does not introduce, delete, or rewrite skill rules in S_{body}. Instead, it organizes execution-lapse evidence into appendix content anchored to the updated skill body. Each appendix item highlights a valid skill that the executor failed to follow, so that later executions pay closer attention to that skill. The appendix update can merge duplicate appendix items, remove obsolete appendix items that no longer correspond to the updated skill body, and incorporate new execution-lapse reflections.

The next skill version is therefore

$$S^{(n+1)} = \left(S_{\text{body}}^{(n+1)}, S_{\text{app}}^{(n+1)}\right). \tag{12}$$

This design separates body-level skill revision from skill-appendix update: DISCOVERY, OPTIMIZATION, and SKILLDEFECT reflections can change the skill body, while EXECUTIONLAPSE reflections only affect the appendix part. As a result, EmbodiSkill updates the skill selectively while reducing the risk of corrupting valid skill content.

3.2.3. Skill-Aware Evolution Spiral

The Skill-Aware Evolution Spiral describes how EmbodiSkill turns repeated skill-aware revisions into progressive skill improvement. At evolution step n, the executor uses the current skill $S^{(n)} = (S_{\text{body}}^{(n)}, S_{\text{app}}^{(n)})$ to perform embodied tasks. Each task execution produces a trajectory, which is reflected against the current skill to generate revision signals. These signals update the skill into $S^{(n+1)} = (S_{\text{body}}^{(n+1)}, S_{\text{app}}^{(n+1)})$. The revised skill is then used for subsequent task execution, producing new trajectories that may expose missing skill content, better execution strategies, skill defects, or execution lapses. In this way, the skill shapes future task execution, and the resulting trajectories provide new evidence for further skill evolution.

The spiral is driven by the interaction between the evolving skill and the trajectories it produces. When S_{body} is expanded, optimized, or corrected, the executor may handle task situations that were previously difficult, producing trajectories that expose new opportunities for further skill improvement. When S_{app} is updated from execution lapses, the executor is reminded to follow valid skill content more carefully during subsequent task execution. Across evolution steps, the skill therefore accumulates targeted improvements: S_{body} becomes more complete and accurate, while S_{app} makes valid skill content more salient during execution, improving the executability of the overall skill.

4. Experiments

4.1. Experimental Setup

Benchmarks and metrics. We evaluate EmbodiSkill on three embodied task benchmarks: ALFWorld [2], EmbodiedBench-Habitat, and EmbodiedBench-Navigation [3]. ALFWorld contains household task completion problems in interactive environments, involving object search, navigation, container interaction, and object state changes. EmbodiedBench-Habitat evaluates visual object interaction in 3D environments, while EmbodiedBench-Navigation evaluates visual navigation. In our split, ALFWorld contains 3,553 training tasks and 134 test tasks; EmbodiedBench-Habitat contains 1,000 training tasks and six test subsets with 50 tasks each; and EmbodiedBench-Navigation contains 1,000 training tasks and five test subsets with 60 tasks each. We report task success rate for all benchmarks.

Skill evolution protocol. EmbodiSkill evolves skills from trajectories collected on training tasks and evaluates the evolved skill on held-out test tasks. We set the maximum number of reflections per trajectory to $K = 1$ and perform 10 skill revision stages unless otherwise specified. Training tasks are sampled in randomized order, and the training stream is reshuffled when additional trajectories are required. During test evaluation, the evolved skill is fixed.

Table 1 | Task success rate (%) on ALFWorld. We report overall success and six ALFWorld subtask categories. Bold indicates the best result in each column; underline indicates the second-best result on the overall score.

Method	Overall	ALFWorld Subtasks					
		Put	Clean	Heat	Cool	Examine	Puttwo
Closed-source direct agents							
GPT-5.2	70.89	87.50	67.74	56.52	76.19	83.33	52.94
Gemini-3-flash	82.09	91.67	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	0	4.35	0	88.89	0
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
EmbodiSkill (Ours)							
EmbodiSkill w/ GPT-5.2	86.57	91.67	**96.77**	73.91	**95.24**	83.33	70.59
EmbodiSkill w/ Gemini-3-flash	85.82	79.17	93.55	73.91	90.48	**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	74.62	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
EmbodiSkill (Ours)							
EmbodiSkill w/ GPT-5.2	**93.28**	**95.83**	**96.77**	73.91	**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

Models. For ALFWorld, we instantiate the executor π_θ with Qwen2.5-14B-Instruct [23] and Qwen3.5-27B [24]. For EmbodiedBench-Habitat and EmbodiedBench-Navigation, Qwen3-VL-8B-Instruct and Qwen3-VL-32B-Instruct as executors [25]. The skill evolution model F is instantiated with GPT-5.2 [26] or Gemini-3-flash [27]. All executor parameters are kept fixed during skill evolution, so performance gains come from the evolving skill rather than model parameter updates.

Baselines. We compare EmbodiSkill with two groups of baselines in the main results. First, closed-source direct agents use GPT-5.2 or Gemini-3-flash directly for task execution without an evolving skill. Second, memory-based methods, including Mem0 [17], G-Memory [14], and LangMem [19], store and retrieve trajectory-level information to guide task execution. For methods using the same local executor, the executor is kept identical so that performance differences reflect the external skill or memory mechanism rather than changes in the executor.

4.2. Main Results

Table 1 and Table 2 report the main results on ALFWorld and EmbodiedBench. Overall, EmbodiSkill improves task success by evolving an external skill from training trajectories, showing stronger performance

Table 2 | Task success rate (%) on EmbodiedBench. Avg. reports the average success rate across subcategories, while subcategory scores are rounded to integers for readability. Ours-GPT and Ours-Gemini denote EmbodiSkill using GPT-5.2 and Gemini-3-flash as the skill evolution model, respectively. Bold indicates the best Avg. result for each benchmark; underline indicates the second-best Avg. result.

Method	Avg.	EB-Habitat						Avg.	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	72	68	65	18
Gemini-3-flash	46.00	78	42	38	56	32	30	56.00	62	60	55	47	57
Executor: Qwen3-VL-8B-Instruct													
Memory-based baselines													
No memory	24.00	58	20	20	18	18	10	45.00	58	53	62	50	2
Mem0	20.33	60	16	8	16	14	8	46.33	58	57	62	55	0
G-Memory	25.66	58	14	12	34	30	6	37.33	62	43	43	38	0
LangMem	19.67	60	12	8	16	14	8	47.00	60	57	62	57	0
EmbodiSkill (Ours)													
Ours-GPT	45.33	76	46	36	52	38	24	50.33	68	65	62	57	0
Ours-Gemini	41.00	78	34	32	52	34	16	49.00	60	63	67	52	3
Executor: Qwen3-VL-32B-Instruct													
Memory-based baselines													
No memory	34.67	78	26	28	34	28	14	46.33	60	53	60	53	5
Mem0	38.33	84	24	26	42	34	20	52.00	62	60	65	67	7
G-Memory	45.00	92	24	36	50	32	34	50.33	52	53	58	65	23
LangMem	38.33	86	26	28	40	32	18	52.00	62	60	65	67	7
EmbodiSkill (Ours)													
Ours-GPT	<u>50.33</u>	92	46	50	54	34	26	**61.33**	65	68	73	67	33
Ours-Gemini	**52.33**	96	44	52	62	36	24	**61.33**	67	70	70	68	32

than direct model execution and trajectory-level memory retrieval.

Results on ALFWorld. On ALFWorld, EmbodiSkill achieves the best overall performance among all evaluated methods. With Qwen3.5-27B as the executor and GPT-5.2 as the skill evolution model, EmbodiSkill reaches 93.28% task success. This outperforms GPT-5.2 used as a direct agent by 31.58% and Gemini-3-flash used as a direct agent by 13.63%. Compared with memory-based methods under the same Qwen3.5-27B executor, EmbodiSkill exceeds the strongest memory baseline, G-Memory, by 25.01%. These results indicate that consolidating training trajectories into an evolving skill provides stronger guidance than directly executing with a frontier model or retrieving trajectory-level memory.

The subtask breakdown further shows that the improvement is not concentrated in a single task type. EmbodiSkill with Qwen3.5-27B and GPT-5.2 achieves the best or tied-best performance on five of the six ALFWorld subtask categories, including Put, Clean, Cool, Examine, and Puttwo. The gain is especially clear on Puttwo, where EmbodiSkill reaches 100.00%, compared with 52.94% for G-Memory under the same executor. This suggests that the evolved skill is particularly helpful for multi-step household tasks that require object search, state tracking, and correct action ordering.

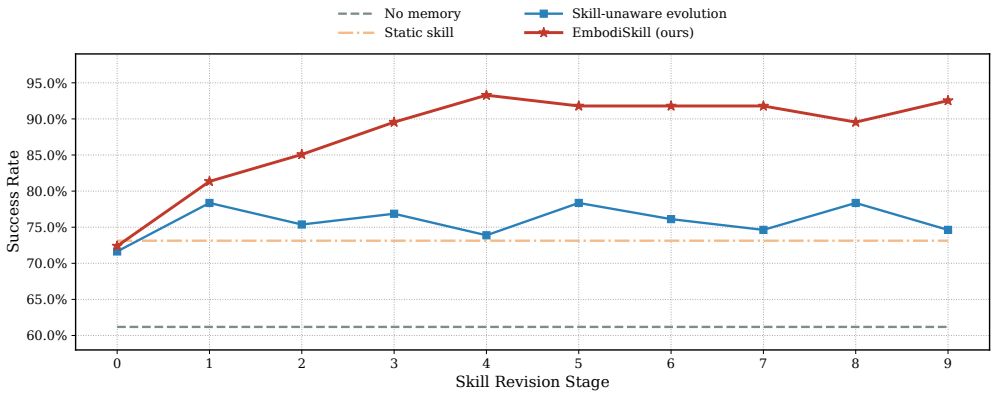

Figure 3 | ALFWorld test success rate across skill revision stages. EmbodiSkill quickly improves from the static skill and remains at a high success rate, while skill-unaware evolution converges to a lower performance range.

Results on EmbodiedBench. Table 2 reports results on visual embodied object interaction and navigation. On EB-Habitat, EmbodiSkill with Qwen3-VL-32B-Instruct and Gemini-3-flash achieves the best average score of 52.33%, outperforming the strongest memory-based baseline by 16.29% and the strongest closed-source direct agent by 13.76%. On EB-Navigation, EmbodiSkill with Qwen3-VL-32B-Instruct reaches the best average score of 61.33% with both GPT-5.2 and Gemini-3-flash as the skill evolution model, outperforming the strongest memory-based baseline by 17.94% and the strongest closed-source direct agent by 6.98%. These results show that EmbodiSkill also improves task success in visual embodied settings, where agents must rely on visual observation and spatial grounding to complete object interaction and navigation tasks.

Table 3 | Ablation of skill self-evolution and skill-aware reflection on ALFWorld. We report task success rate (%). Δ_{aware} denotes the improvement of EmbodiSkill over skill-unaware evolution.

Executor	**Skill Model**	**Configurations**				Δ_{aware}
		No skill	**Static Skill**	**Skill-unaware**	**EmbodiSkill**	
Qwen2.5-14B	GPT-5.2	46.27	65.67	70.90	86.57	+15.67
	Gemini-3-flash		58.95	67.91	85.82	**+17.91**
Qwen3.5-27B	GPT-5.2	61.19	73.13	78.36	**93.28**	+14.92
	Gemini-3-flash		79.85	85.82	87.31	+1.49

4.3. Ablation and Analysis

Ablation of skill self-evolution and skill-aware reflection. To analyze where the improvement of EmbodiSkill comes from, we compare four settings on ALFWorld. *No memory* directly uses the executor without external memory or skill evolution. *Static Skill* uses the initial skill without further revision. *Skill-unaware* updates the skill from trajectories, but revises the skill coarsely without explicitly identifying the implicated skill content, revision type, or separation between skill-body revision and skill-appendix update. EmbodiSkill performs the full skill-aware reflection and revision process.

As shown in Table 3, both skill self-evolution and skill-aware reflection contribute to performance. With Qwen3.5-27B as the executor and GPT-5.2 as the skill evolution model, the static skill improves over the no-memory setting from 61.19% to 73.13%, corresponding to a 19.51% relative improvement and showing the benefit of skill-guided execution. Skill-unaware evolution further improves the result to 78.36%, but remains substantially below EmbodiSkill, which reaches 93.28%. This corresponds to a 27.55% relative improvement over the static skill and a 19.04% relative improvement over skill-unaware evolution. Similar gains are observed with Qwen2.5-14B-Instruct, where EmbodiSkill improves over skill-unaware evolution by 22.10% with GPT-5.2 and 26.37% with Gemini-3-flash. These results show that the gain does not come only from updating the skill, but from making the update skill-aware and targeted.

Evolution over skill revision stages. We further examine how performance changes during skill self-evolution. Figure 3 shows the ALFWorld test success rate across 10 skill revision stages under the Qwen3.5-27B executor and GPT-5.2 skill evolution model. EmbodiSkill rapidly improves from the static skill performance of 73.13% and reaches 93.28% during evolution, after which it remains in a high performance range. In contrast, skill-unaware evolution improves over the static skill but converges to a lower performance range and exhibits larger fluctuations. This suggests that directly revising the skill without skill-aware attribution can make evolution less stable, while EmbodiSkill produces more reliable improvement by updating targeted skill content.

5. Conclusion

We presented EmbodiSkill, a framework for embodied skill self-evolution. EmbodiSkill uses skill-aware reflection to examine trajectories with respect to the current skill and produce targeted revision signals, avoiding coarse whole-skill rewriting. These signals are integrated through a Skill-Aware Evolution Spiral, where revised skills guide subsequent task execution and new trajectories provide further evidence for skill improvement. Experiments on ALFWorld and EmbodiedBench show that EmbodiSkill improves task success over direct agents and memory-based baselines, and ablations confirm the importance of skill-aware revision. These results suggest that skill-aware self-evolution is a promising way to build embodied agents that accumulate reusable procedural knowledge from their own trajectories.

References

[1] 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*, pages 10740–10749, June 2020.

[2] 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.

[3] 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*, pages 70576–70631. PMLR, 2025.

[4] 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*, pages 287–318. PMLR, 2023.

[5] 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*, pages 9493–9500. IEEE, 2023.

[6] 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.

[7] 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.

[8] 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*, 2026. Accepted to ACL 2026 Findings. arXiv:2508.06433.

[9] 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*, 2026. Accepted to ACL 2026 Findings. arXiv:2512.10696.

[10] 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.

[11] 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, 2026.

[12] 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.

[13] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147. PMLR, 2022.

[14] 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, pages 12988–13018. Curran Associates, Inc., 2025.

[15] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc., 2023.

[16] 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.

[17] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory, 2025.

[18] 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, pages 17577–17604. Curran Associates, Inc., 2025.

[19] LangChain AI. Langmem: A framework for long-term memory in llm agents. https://github.com/langchain-ai/langmem, 2026. Accessed: 2026-05-04.

[20] 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.

[21] 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, 2026.

[22] 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. Accepted to ICML 2026. arXiv:2603.12056.

[23] 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.

[24] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026.

[25] 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.

[26] OpenAI. Update to GPT-5 system card: GPT-5.2. OpenAI system card update, December 2025.

[27] Google DeepMind. Gemini 3 Flash model card. Model card, December 2025.