MMSkills: Towards Multimodal Skills for General Visual Agents

Kangning Zhang[1,2,†,*] **Shuai Shao**[1,2,†,*] **Qingyao Li**[1,2,*]
Jianghao Lin[1] **Lingyue Fu**[1] **Shijian Wang**[3]
Wenxiang Jiao[2,‡] **Yuan Lu**[2,‡] **Weinan Zhang**[1,‡]
Yong Yu[1,‡] **Weiwen Liu**[1,‡]
[1]Shanghai Jiao Tong University [2]Xiaohongshu Inc.
[3]Southeast University
[†]Equal contribution. [*]Work done during internship at Xiaohongshu Inc.
[‡]Corresponding authors.
zhangkangning@sjtu.edu.cn, wenxiangjiaonju@gmail.com, wwliu@sjtu.edu.cn

Abstract

Reusable skills have become a core substrate for improving agent capabilities, yet most existing skill packages encode reusable behavior primarily as textual prompts, executable code, or learned routines. For visual agents, however, procedural knowledge is inherently multimodal: reuse depends not only on what operation to perform, but also on recognizing the relevant state, interpreting visual evidence of progress or failure, and deciding what to do next. We formalize this requirement as *multimodal procedural knowledge* and address three practical challenges: (I) **what** a multimodal skill package should contain; (II) **where** such packages can be derived from public interaction experience; and (III) **how** agents can consult multimodal evidence at inference time without excessive image context or over-anchoring to reference screenshots. We introduce *MMSkills*, a framework for representing, generating, and using reusable multimodal procedures for runtime visual decision making. Each MMSkill is a compact, state-conditioned package that couples a textual procedure with runtime state cards and multi-view keyframes. To construct these packages, we develop an agentic trajectory-to-skill Generator that transforms public non-evaluation trajectories into reusable multimodal skills through workflow grouping, procedure induction, visual grounding, and meta-skill-guided auditing. To use them, we introduce a branch-loaded multimodal skill agent: selected state cards and keyframes are inspected in a temporary branch, aligned with the live environment, and distilled into structured guidance for the main agent. Experiments on desktop GUI tasks and two games show overall gains for frontier and smaller multimodal agents, with some ties and application-level regressions, suggesting that external multimodal procedural knowledge can complement model-internal priors. Code and skills are available at `https://github.com/zkangning/MMSkills_for_Visual_Agents` and `https://huggingface.co/datasets/zhangkangning/mmskills`.

1 Introduction

Skills have become one of the central abstractions for building useful agents: recent systems store reusable behaviors as prompts, code, execution graphs, or learned routines that can be retrieved and composed later (Wang et al., 2023a; Zheng et al., 2025; Chen et al., 2026; Wang et al., 2026a). Despite differences in implementation, these skills largely share a common representational assumption: reusable knowledge can be expressed as a textual or code-level specification of actions. This design is effective when the relevant state can be adequately abstracted in language, but it is insufficient for

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

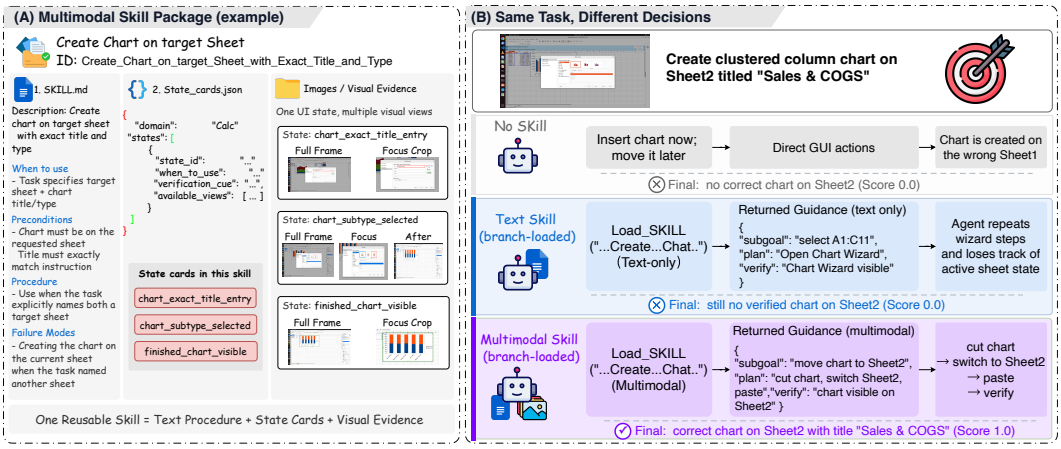

Figure 1: A concrete MMSkills example. A multimodal skill package combines a textual procedure, runtime state cards, and multi-view visual evidence. For the same chart-creation task, text-only guidance can miss the active sheet state, while branch-loaded MMSkills align skill evidence with the live screen and return state-aware guidance for the main agent.

multimodal agents whose decisions depend on visual evidence. For such agents, reusable experience must specify not only what operation to perform, but also how to recognize the relevant state, and how visual evidence should guide the next decision. A desktop agent may know the correct operation but fail to recognize that a dialog is not yet ready; a game agent may know the intended goal but still require visual cues to distinguish progress from completion. Consequently, text-only skills become verbose yet underspecified, whereas demonstrations preserve visual context but are lengthy, instance-specific, and difficult to adapt.

This gap suggests the need for *multimodal procedural knowledge*: reusable guidance that binds action procedures to the visual evidence and state-dependent decisions required for applying them. Such knowledge is not simply a text skill with screenshots attached. To be reusable, it must specify what procedure is being reused, when the procedure should or should not be used, which visible cues matter, and which evidence verifies progress, failure, or completion. Turning this requirement into practical multimodal skill libraries raises three central challenges:

- **Representation.** What should a multimodal skill package contain, and how should it bind procedures, visible, and verification cues into a coherent reusable unit?
- **Generation.** Where can such packages be derived from, if they must use public non-evaluation interaction experience rather than hand-written examples or raw demonstration replay?
- **Utilization.** How can an agent consult multimodal skill evidence at inference time while avoiding excessive image context, distracting state descriptions, and over-anchoring to reference screenshots?

We propose **MMSkills**, a framework for representing, generating, and utilizing reusable multimodal procedures for runtime visual decision making. Each MMSkill couples a *textual procedure*, which describes the reusable action pattern, with *runtime state cards*, which encode when-to-use and when-not-to-use conditions, visible cues, verification cues, and available views, and *multi-view keyframes*, which ground critical states through full-frame, focused, and optional before/after views. The resulting package is not a text instruction with illustrative images attached. It is a state-conditioned procedure whose visual evidence helps the agent decide when to follow, skip, or verify the procedure.

To **generate** the multimodal skill package, we introduce an ***automated trajectory-to-skill Generator*** built around an agentic, meta-skill-guided pipeline. This generation problem is substantially harder than text-skill extraction: while prior pipelines can often compress successful rollouts, failure analyses, or accumulated traces into reusable instructions or action abstractions (Zheng et al., 2025; Wang et al., 2026a; Alzubi et al., 2026; Ma et al., 2026; Xia et al., 2026; Li et al., 2026), generating MMSkills must also identify reusable visual states, select diagnostic frames, and bind each visual cue to the decision rule it supports. Our Generator operates on public trajectories that are **separate from evaluation tasks**: it groups related workflows, induces candidate procedures, merges overlapping candidates, grounds them in real non-test trajectory frames, and audits the resulting packages with

reusable multimodal-skill-factory meta-skills. This process converts public interaction data into compact visual procedural knowledge without storing raw demonstrations as the skill.

For effective **utilization**, we introduce ***branch loading*** to consult the multimodal skills without injecting the entire package into the main trajectory. Existing skill agents commonly insert retrieved skills directly into the main interaction context. This loading pattern becomes problematic for MMSkills: a single package may contain several state cards together with multi-view screenshots, so direct insertion creates substantial context pressure and makes reference images compete with the live observation. More importantly, the main agent can become visually anchored to superficially similar reference screenshots, planning around the skill example rather than the current environment. Branch loading addresses this issue as a multimodal form of progressive disclosure over skill evidence (Xu and Yan, 2026). When the main agent considers a skill, it opens a temporary branch that selects the needed state cards and keyframe views, aligns them with the live screen or scene, and returns compact structured guidance with applicability judgments, subgoals, and next-step plans. The main trajectory receives distilled decision support rather than the full skill package, as illustrated by the example in Figure 1.

We evaluate MMSkills across GUI and game-based visual agent tasks, including OSWorld (Xie et al., 2024), macOSWorld (Yang et al., 2025b), VAB-Minecraft from VisualAgentBench (Liu et al., 2024), and Super-Mario in LMGame-Bench (Hu et al., 2025). Across frontier and smaller multimodal models, MMSkills improve aggregate results in most evaluated settings. Here, general visual agents refers to agents acting from visual observations; our experiments establish within-domain reuse with separately constructed libraries, not cross-domain or held-out-workflow transfer.

Our main contributions are summarized as follows:

- We introduce the **multimodal skill package**, formulating reusable skills for general visual agents as multimodal procedural knowledge: compact, state-conditioned units that organize textual procedures, runtime state cards, and multi-view keyframes for visual decision making.
- We develop an agentic trajectory-to-skill **Generator** that turns public, non-evaluation trajectories into multimodal skill packages through workflow grouping, procedure induction, visual grounding, and meta-skill-guided auditing.
- We propose **branch loading**, a runtime mechanism that selects and aligns multimodal skill evidence in a temporary branch before returning structured decision support to the main agent.
- We demonstrate performance gains across GUI and game-based visual-agent benchmarks and multiple model families, showing that external multimodal procedural knowledge complements model-internal priors.

2 Methods

2.1 Overview

MMSkills are designed around three components: a *multimodal skill package* that stores reusable visual procedural knowledge, a *Skill Generation pipeline* that constructs such packages from public trajectories, and a *branch-loaded multimodal skill agent* that isolates skill-environment grounding in a temporary branch and returns distilled decision support to the main trajectory at inference time. Figure 2 gives the system overview.

At a high level, the Generator maps non-evaluation trajectories $\mathcal{T} = \{\tau_i\}$ into a multimodal skill library $\mathcal{M} = \{M_i\}_{i=1}^{N}$. Before an episode begins, the runtime agent pre-recalls a task-level candidate set $\mathcal{C}_I \subset \mathcal{M}$ from the instruction I and compact skill descriptors. During execution, the main agent observes the current visual observation O_t, maintains a short history H_t, and either acts directly or consults a temporary skill branch for some $M_t \in \mathcal{C}_I$:

$$\begin{aligned} \text{direct}: &\quad A_t = \pi_{\text{main}}(O_t, H_t, \mathcal{C}_I), \\ \text{branch}: &\quad G_t = \text{Branch}(O_t, H_t, M_t), \quad A_t = \pi_{\text{main}}(O_t, H_t, \mathcal{C}_I, G_t). \end{aligned} \tag{1}$$

The branch output is a structured guidance tuple

$$G_t = (\text{applicable}_t, \text{subgoal}_t, \text{plan}_t, \text{do_not_do}_t, \text{verify}_t), \tag{2}$$

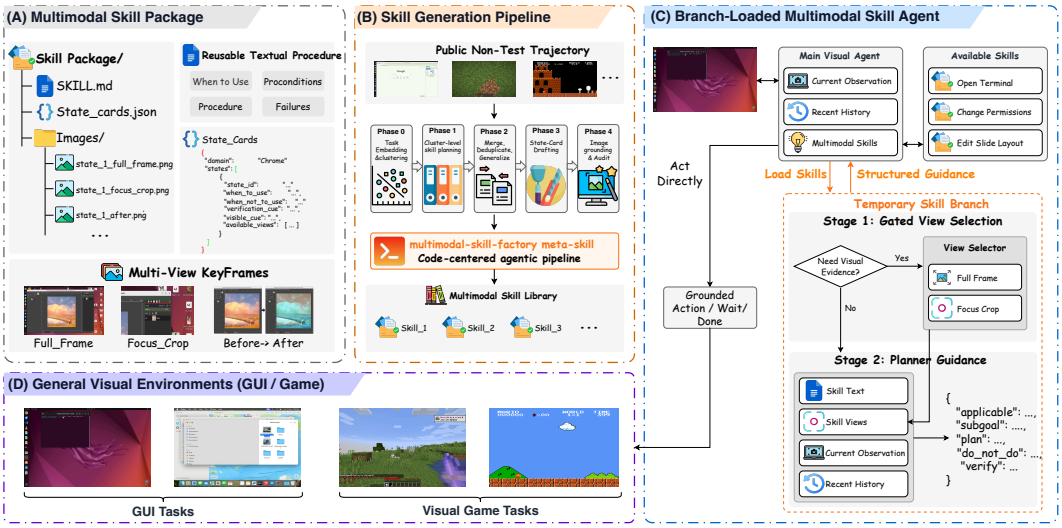

Figure 2: Overview of the MMSkills framework. A multimodal skill package stores a reusable textual procedure, runtime state cards, and multi-view keyframes. A meta-skill-guided Generator converts public non-test trajectories into a reusable multimodal skill library. At inference time, the main visual agent uses branch loading to inspect selected skill evidence in a temporary branch and receives compact structured guidance before acting.

where the fields respectively give the applicability judgment, local subgoal, skill-conditioned plan, negative constraints, and visual verification check. The main agent uses G_t as decision support, while executable action grounding remains tied to the live observation.

2.2 Multimodal Skill Package

We represent each MMSkill as a state-conditioned procedure package

$$M = (D, P, S, K), \tag{3}$$

where D is a compact descriptor, P is a reusable textual procedure, $S = \{S_j\}_{j=1}^{m}$ is a set of runtime state cards, and $K = \{K_j\}_{j=1}^{m}$ is a set of keyframe bundles aligned with those cards. Each pair (S_j, K_j) corresponds to one decision-relevant procedural state. The procedure specifies the reusable workflow; the state card specifies when the workflow is valid or invalid; and the keyframes make the state visually recognizable at runtime.

A runtime state card is an agent-facing state node rather than an image caption. It links a point in the procedure to when-to-use conditions, when-not-to-use conditions, visible cues, verification cues, and available views:

$$\begin{aligned} S_j = (&\text{when_to_use}_j, \text{when_not_to_use}_j, \text{visible_cues}_j, \\ &\text{verification_cue}_j, \mathcal{V}_j), \qquad \mathcal{V}_j = \text{available_views}_j. \end{aligned} \tag{4}$$

The first two fields define when the state should be followed or skipped, $\texttt{visible_cues}_j$ states what evidence to inspect, $\texttt{verification_cue}_j$ defines the progress or completion check, and $\mathcal{V}_j$ lists which views may be loaded. This schema makes the skill useful for decision making: the agent can decide whether to follow, skip, or verify the procedure.

Each key state is grounded by a small multi-view bundle. Let

$$\mathcal{V} = \{\text{full_frame}, \text{focus_crop}, \text{before}, \text{after}\}. \tag{5}$$

Then

$$K_j = \{K_j^v : v \in \mathcal{V}_j,\ v \in \mathcal{V}\}. \tag{6}$$

The full-frame view preserves global context, the focus crop localizes the visual cue, and optional before/after views expose useful transitions. These images are reference evidence, not coordinates to copy. Under this representation, a text-only skill is the degenerate package $(D, P, \emptyset, \emptyset)$; MMSkills extend it by binding procedure, decision conditions, and visual evidence into one reusable unit.

2.3 Skill Generator from Public Trajectories

We build MMSkills from public interaction trajectories that are separate from evaluation tasks. A trajectory is

$$\tau_i = (I_i, O_{i,1:T_i}, A_{i,1:T_i}), \tag{7}$$

where I_i is the task instruction, $O_{i,t}$ are visual observations, $A_{i,t}$ are executed actions. The Generator is controlled by a reusable multimodal-skill-factory meta-skill $\mathcal{F}$:

$$\mathcal{G}_{\mathcal{F}} : \mathcal{T}_d \mapsto \mathcal{M}_d, \tag{8}$$

where $\mathcal{T}_d$ is the public trajectory pool for domain d and $\mathcal{M}_d$ is the generated domain skill library. The library is teacher-generated offline and frozen throughout evaluation; downstream agents do not write back their test trajectories. The pipeline comprises five stages:

$$\begin{aligned} \mathcal{T}_d \xrightarrow{\text{Phase 0: embed+cluster}} \mathcal{C}_d \xrightarrow{\text{Phase 1: cluster plan}} \mathcal{A}_d \xrightarrow{\text{Phase 2: merge}} \mathcal{R}_d \\ \xrightarrow{\text{Phase 3: text draft}} \widehat{\mathcal{M}}_d \xrightarrow{\text{Phase 4: image ground+audit}} \mathcal{M}_d. \end{aligned} \tag{9}$$

- **Phase 0: task embedding and clustering.** The pipeline embeds task instructions and trajectory metadata, then groups a broad domain into semantically focused clusters $\mathcal{C}_d$.
- **Phase 1: cluster-level skill planning.** For each cluster, an LLM-based agent proposes atomic skills with workflow boundaries, completion conditions, and covered task ids, producing a domain planning table $\mathcal{A}_d$.
- **Phase 2: skill merging.** Cluster-level plans are deduplicated, merged, and generalized into merged skill specifications $\mathcal{R}_d$, while overly broad umbrella skills are rejected.
- **Phase 3: text-first drafting.** Without reading images, the Generator selects reference tasks and drafts the descriptor D, textual procedure P, and planned state cards, yielding $\widehat{\mathcal{M}}_d$.
- **Phase 4: image grounding and audit.** The Generator reads selected keyframes, grounds focus regions, constructs multi-view bundles, and audits the final packages.

For a merged skill $r \in \mathcal{R}_d$, finalization is written as

$$\widehat{M}_r = (D_r, P_r, \widehat{S}_r, \widehat{K}_r) \xrightarrow{\text{ground+audit}} M_r = (D_r, P_r, S_r, K_r). \tag{10}$$

The visual grounding policy is conservative: views are added only for state recognition, transition comparison, or completion verification, so the skill stores diagnostic states rather than replaying demonstrations. The meta-skill $\mathcal{F}$ supplies reusable scripts, schemas, and quality gates for the LLM-based Generator, while external services are limited to bounded support steps such as embedding/clustering and grounding.

The implementation uses Codex guided by the meta-skill for construction, and Qwen3-VL-235B-A22B-Thinking for visual grounding. Codex selects decision-relevant source frames from trajectory text; the grounding model localizes each card's cue or action target for cropping, while retaining the full frame. Appendix D specifies the sources, models, and audit checks.

2.4 Branch-loaded Multimodal Skills Agent

Most skill-using agents load a retrieved skill directly into the main interaction context. For short text skills, this is reasonable: the skill is read as an additional instruction alongside the observation. For MMSkills, direct loading is brittle because state cards, multi-view keyframes, and transition examples add substantial context pressure, and irrelevant reference views can anchor the agent away from the live environment. Figure 2(C) illustrates the branch-loaded alternative, which moves skill-environment grounding out of the main trajectory.

Stage 1: gated view selection. Suppose the main agent calls $M_t = (D_t, P_t, S_t, K_t) \in \mathcal{C}_I$. The branch first selects which state cards and view types are relevant to the live observation:

$$(J_t, R_t) = \text{SelectViews}(O_t, H_{t-1}, P_t, S_t), \qquad V_t = \{K_j^v : j \in J_t,\ v \in R_{t,j}\}, \tag{11}$$

where J_t indexes selected state cards and $R_{t,j} \subseteq \mathcal{V}_j$ selects views for state j. The selector reads the live observation, recent history, textual procedure, and state-card descriptions before loading images. If text and state cards are sufficient, $R_{t,j}$ may be empty.

Table 1: OSWorld application-level success rates. All entries are percentages. "Calc", "Impress", and "Writer" denote LibreOffice applications.

Base model	Skill condition	Chrome	GIMP	Calc	Impress	Writer	Multi-app	OS	Mail	VLC	VS Code	Overall
Gemini 3.1 Pro	No skill	53.47	34.62	**57.45**	40.43	47.82	**31.97**	54.17	40.00	35.29	56.52	44.08
	Text-only	44.35	34.62	38.30	40.34	56.52	22.38	**70.83**	**66.67**	41.18	56.52	40.76
	MMSkills	**59.91**	**50.00**	53.19	**53.19**	**60.86**	24.11	**70.83**	**66.67**	**70.59**	**65.22**	**50.11**
Gemini 3 Flash	No skill	37.78	50.00	38.30	29.73	52.17	21.51	54.17	66.67	52.39	47.83	36.65
	Text-only	51.02	23.08	38.30	34.00	56.52	19.16	54.17	60.00	58.82	52.17	40.27
	MMSkills	**55.37**	42.31	**53.19**	**40.34**	**56.52**	**30.98**	**75.00**	**66.67**	52.94	**60.87**	**47.97**
Qwen3-VL-235B	No skill	15.56	38.46	17.02	25.53	43.48	9.48	25.00	26.67	17.65	34.78	21.34
	Text-only	42.22	50.00	10.64	21.31	34.78	14.86	33.33	60.00	35.29	47.83	28.57
	MMSkills	**59.91**	**69.23**	**23.40**	**32.01**	**47.82**	**19.35**	**41.67**	**73.33**	**41.18**	**56.52**	**39.17**
GLM-5V	No skill	37.78	19.23	21.28	29.70	26.08	18.70	54.17	53.33	11.76	47.83	28.71
	Text-only	**53.24**	**53.85**	**31.91**	**31.98**	**52.17**	20.24	20.83	**46.67**	**35.29**	**65.22**	36.61
	MMSkills	51.02	**53.85**	**31.91**	31.83	43.47	**22.26**	**66.67**	40.00	23.53	**65.22**	**38.51**
Kimi-K2.6	No skill	51.02	34.62	34.04	35.32	30.43	14.86	54.17	66.67	32.60	52.17	34.98
	Text-only	**57.69**	40.00	**40.43**	36.14	17.38	22.38	62.50	53.33	**58.82**	43.48	39.66
	MMSkills	**57.69**	**42.31**	**40.43**	**48.92**	**60.86**	**23.40**	**79.17**	**73.33**	41.18	**69.57**	**46.59**
Qwen3-VL-8B-Instruct	No skill	15.47	7.69	2.13	8.59	4.34	7.33	25.00	13.33	29.41	17.39	10.78
	Text-only	19.91	11.54	6.38	16.99	**17.39**	7.33	16.67	33.33	17.65	34.78	14.93
	MMSkills	**39.91**	**42.31**	**8.51**	**23.37**	**17.39**	**13.43**	**25.00**	**60.00**	**29.41**	**47.83**	**25.40**

Note: Due to the substantially higher inference cost and wall-clock time of Gemini 3.1 Pro and Kimi-K2.6, we report their full three-condition results only on OSWorld.

Stage 2: branch planning. The branch then aligns the selected evidence with the live state and returns structured guidance:

$$G_t = \text{PlanBranch}(O_t, H_{t-1}, P_t, \{S_j : j \in J_t\}, V_t), \tag{12}$$

where G_t follows Eq. 2. The main agent does not execute G_t mechanically; it uses G_t as an intermediate planning signal and still chooses a grounded action from the live screenshot. This is intended to preserve procedural guidance while keeping action grounding tied to the current observation; erroneous view selection or branch advice can still mislead the agent. Live-state matching, invocation, and view selection are controlled by the evaluated model, rather than an oracle matcher. Appendix J gives the full runtime loop in Algorithm 1, and Appendix K reports the prompt templates used by the main agent and the two branch stages.

3 Experiments

We evaluate whether MMSkills provide useful external procedural knowledge for visual agents. The experiments are organized around four research questions:

- **RQ1: Overall performance on GUI and game tasks.** Do MMSkills improve visual agents across realistic desktop environments and open-ended visual game tasks?
- **RQ2: Ablations of skill content and branch loading.** Which parts of MMSkills matter, and how do branch loading and view selection affect multimodal skill use?
- **RQ3: Skill usage and interaction dynamics.** How often are MMSkills invoked, how do they affect interaction length, and which visual views are selected at runtime?
- **RQ4: Behavioral shift analysis.** How do MMSkills change the agent's low-level action patterns beyond final success rate?

3.1 Experimental Setup

In all settings, agents plan from visual observations, namely desktop or game screenshots. We evaluate on OSWorld (Xie et al., 2024), macOSWorld (Yang et al., 2025b), VAB-Minecraft from VisualAgentBench (Liu et al., 2024), and Super Mario Bros from LMGame-Bench (Hu et al., 2025), covering both realistic GUI tasks and open visual game environments. Detailed benchmark descriptions and test-case distributions are illustrated in Appendix A; implementation details, evaluation protocols, model choices, and runtime variants are given in Appendix C.

Table 2: Auxiliary GUI and game-based visual-agent results. macOSWorld reports domain-level and overall success rates; VAB-Minecraft reports success rate and average score; Super Mario Bros reports total performance and total reward.

		macOSWorld						VAB-Minecraft		Super Mario Bros	
Base model	Skill condition	File	Media	Prod.	Sys/IF	Apps	Overall	Success	Avg. score	Total perf.	Total reward
Gemini 3 Flash	No skill	41.38	33.33	60.00	62.07	55.79	55.94	67.24	0.7462	411.00	766.67
	Text-only	31.03	25.00	62.86	**75.86**	55.26	53.85	68.96	0.7541	548.00	912.00
	MMSkills	**58.62**	**50.00**	**77.14**	65.52	**65.73**	**65.73**	**73.28**	**0.7884**	**624.00**	**1081.33**
Qwen3-VL-235B	No skill	31.03	**58.33**	51.43	58.62	44.74	47.55	52.59	0.6308	454.50	955.50
	Text-only	34.48	33.33	37.14	51.72	52.63	43.36	55.17	0.6634	610.50	1138.25
	MMSkills	**37.93**	33.33	**54.29**	**62.07**	**57.89**	**51.75**	**62.07**	**0.7114**	**788.00**	**1514.25**
GLM-5V	No skill	24.14	16.67	40.00	41.38	39.47	34.97	56.03	0.6701	612.75	1191.50
	Text-only	31.03	**66.67**	**62.86**	**58.62**	47.37	**51.75**	61.20	0.6938	794.50	1218.00
	MMSkills	**44.83**	**66.67**	48.57	**58.62**	**50.00**	**51.75**	**68.10**	**0.7495**	**950.50**	**1384.50**
Qwen3-VL-8B-Instruct	No skill	**10.34**	0.00	**14.29**	**3.45**	0.00	**6.29**	23.28	0.3017	415.25	928.75
	Text-only	0.00	**8.33**	2.86	**3.45**	**10.53**	4.90	29.31	0.3754	596.50	997.25
	MMSkills	6.90	**8.33**	8.57	**3.45**	5.26	**6.29**	**38.79**	**0.4668**	**764.00**	**1128.75**

All skills are extracted from non-test data. We evaluate frontier and smaller multimodal models and compare *no-skill*, *text-only skill*, and *MMSkills* conditions, with direct-loading variants studied in the ablations. Dataset-specific skill sources, source statistics, and skill-package distributions are provided in Appendix B.

3.2 RQ1: Overall Performance on GUI and Game Tasks

Table 1 reports OSWorld application-level success rates, and Table 2 reports the auxiliary GUI and game results. **MMSkills improve OSWorld overall performance across all evaluated model families.** Overall success increases for Gemini 3.1 Pro ($44.08\% \rightarrow 50.11\%$), Gemini 3 Flash ($36.65\% \rightarrow 47.97\%$), Qwen3-VL-235B ($21.34\% \rightarrow 39.17\%$), GLM-5V, and Kimi-K2.6. Text-only skills exceed no skill in 14 of 18 model–benchmark pairs. This control is exactly $(D, P, \emptyset, \emptyset)$ under the same branch protocol; occasional regressions do not imply a deliberately weakened baseline. **External multimodal procedural knowledge is especially valuable for weaker visual agents.** For Qwen3-VL-8B-Instruct, MMSkills raise OSWorld from 10.78% to 25.40% and VAB-Minecraft from 23.28% to 38.79%, indicating that explicit visual procedural knowledge can compensate for limited model-internal priors.

The framework also supports separately constructed libraries beyond Ubuntu. On macOSWorld, MMSkills improve the completed large-model runs, including Gemini 3 Flash and GLM-5V, while VAB-Minecraft shows consistent gains in both success rate and average score across all evaluated models. Super Mario Bros follows the same pattern in the completed runs, with higher total performance and reward under MMSkills. These evaluations use domain-specific libraries and therefore support reuse of the same package format, rather than transfer of an individual skill across benchmarks. On macOSWorld, Qwen3-VL-8B ties its no-skill baseline and GLM-5V ties text-only.

Matched controls. Table 3 separates branch deliberation from external experience. Self-deliberation uses a branch without external experience; Raw-RAG retrieves raw trajectories; adapted XSkill extracts structured textual skills from the same non-test OpenCUA pool. The latter controls and MMSkills use the same test agents and two-stage branch protocol. MMSkills exceeds adapted XSkill by 6.34 and 6.72 percentage points for Gemini and Qwen, respectively. This compares the complete packages, while the component ablations below more directly isolate visual evidence. Appendix E reports application-level results and adaptation scope.

3.3 RQ2: Ablations of Skill Content and Branch Loading

Figure 3 combines the skill-content and branch-loading ablations. Unless otherwise stated, skill variants use the branch-loaded agent; the main exception is *Direct load*, which inserts skill content into the main context. For skill content, we compare text-only skills, MMSkills without state cards, MMSkills without images, and the complete MMSkills package. **State cards and multi-view visual evidence both improve skill utility.** Text-only branch loading already improves over the no-skill

Table 3: Matched OSWorld controls: overall success (%). XSkill is our source- and runtime-matched adaptation.

Setting	Gemini 3 Flash	Qwen3-VL-235B
No skill	36.65	21.34
Self-deliberation	39.41	25.80
Raw-RAG	40.52	31.50
Adapted XSkill	41.63	32.45
Text-only MMSkills	40.27	28.57
MMSkills	**47.97**	**39.17**

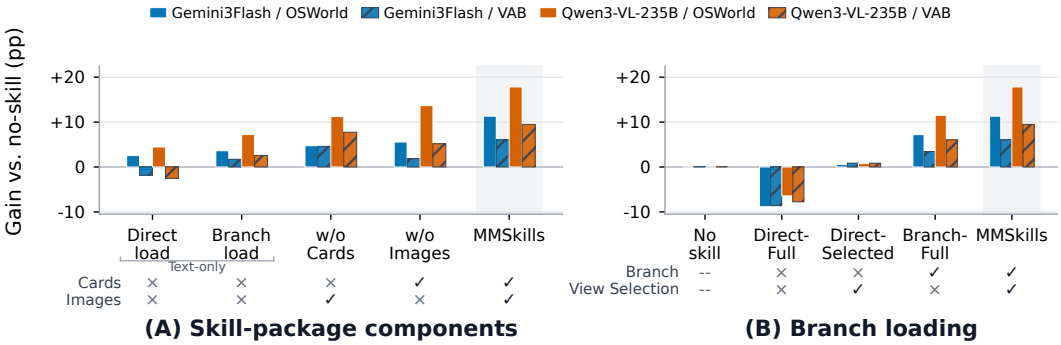

Figure 3: Ablation results for MMSkills components and branch loading. Bars report percentage-point gains over the no-skill baseline. Panel (A) removes runtime state cards or visual keyframes from the skill package. Panel (B) compares direct loading with branch loading and with or without view selection.

baseline, but the complete MMSkills package is consistently stronger. Removing state cards weakens the agent's ability to distinguish relevant runtime states, while removing images preserves decision rules but removes visual grounding evidence. Both removals reduce performance on OSWorld and VAB-Minecraft, supporting complementary roles for state cards and keyframes: one supports state discrimination, and the other helps the agent recognize the corresponding visual evidence. **Branch loading helps even for text-only skills.** The branch-loaded text-only variant is stronger than direct text loading in most model–benchmark pairs, indicating that the temporary branch improves skill interpretation even before multimodal evidence is introduced.

For branch loading, we ablate whether skill evidence is inspected in a temporary branch and whether Stage-1 view selection filters state cards and keyframes. **Branch loading and view selection address different failure modes.** Direct-full loading hurts performance because unfiltered images and state descriptions pollute the main context; view selection alone reduces this damage but stays near baseline. Branch loading already gives clear gains, and the full two-stage design performs best, indicating that separated evidence inspection and filtered visual evidence are useful together in the tested configurations.

3.4 RQ3: Skill Usage and Interaction Dynamics

Table 4 analyzes when and how agents call skills. **MMSkills are invoked more often than text-only skills.** Invocation coverage increases on both OSWorld and VAB-Minecraft for Gemini 3 Flash and Qwen3-VL-235B, with the largest OSWorld change rising from 37.50% to 65.28% for Qwen3-VL-235B. This suggests that multimodal skills make external knowledge easier to recognize as relevant: state cards expose when-to-use and when-not-to-use conditions, and visual cues help the agent detect when its current observation matches a reusable procedural state.

MMSkills shorten trajectories rather than merely adding extra consultation. Text-only skills can add overhead when they provide procedural hints without visual grounding, but MMSkills reduce average steps in every setting, with the largest reductions appearing for Qwen3-VL-235B. These reductions are consistent with shorter interaction paths; they are averages over all episodes, including failures, rather than measurements of optimal successful paths. **Focus crops dominate selected visual evidence.** The branch does not load all views uniformly: focus crops are selected most

Table 4: Skill invocation, interaction length, and selected views. "Invoked" is the percentage of cases with at least one skill call, and "Step Δ" is relative to the no-skill baseline.

Benchmark	Model	Skill condition	Invoked (%)	Calls/case	Steps	Step Δ	Views (Full/Focus/Before/After)
OSWorld	Gemini 3 Flash	No skill	–	–	13.11	0.00	–
		Text-only	41.11	0.7139	16.64	+3.53	–
		MMSkills	**62.50**	**0.9556**	**11.86**	**-1.25**	79/241/8/24
	Qwen3-VL-235B	No skill	–	–	15.22	0.00	–
		Text-only	37.50	0.4917	13.34	-1.88	–
		MMSkills	**65.28**	**0.9222**	**9.87**	**-5.35**	40/27/17/13
VAB-Minecraft	Gemini 3 Flash	No skill	–	–	16.92	0.00	–
		Text-only	68.97	1.8706	17.30	+0.38	–
		MMSkills	**81.90**	**2.4310**	**13.75**	**-3.17**	105/205/15/12
	Qwen3-VL-235B	No skill	–	–	34.74	0.00	–
		Text-only	54.31	1.5776	31.36	-3.38	–
		MMSkills	**64.66**	**2.3534**	**27.07**	**-7.67**	98/196/13/10

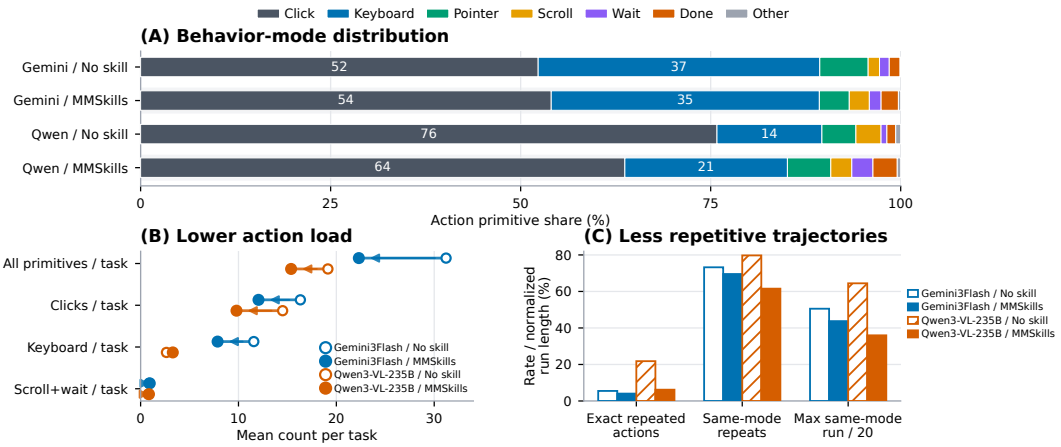

Figure 4: Behavioral shifts induced by MMSkills on OSWorld. Panel (A) reports the distribution of executed action primitives. Panel (B) compares the average number of low-level primitives per task. Panel (C) measures repetitive behavior through exact repeated actions, repeated action modes, and the longest same-mode run normalized by the 20-step budget.

frequently in three of four settings, while full-frame, before, and after views provide global context, transition evidence, and completion references when local crops alone are insufficient.

Inference cost and failure attribution. Branch loading increases model calls on OSWorld (Gemini: 16.01 to 17.59; Qwen: 17.29 to 17.55), while measured mean wall-clock time decreases from 216.99 to 192.64 seconds and 257.38 to 215.04 seconds. API cost changes from $1.409 to $1.436 and from $0.311 to $0.290 per task, respectively; fewer environment steps do not imply uniformly lower compute. Appendix F gives all six configurations. An audit of 293 failed episodes that invoked MMSkills attributes 55 to misleading keyframes, 57 to branch-guidance errors, and 173 to execution/verification after usable guidance. These model-assisted categories describe observed failures, not causal guarantees (Appendix H).

3.5 RQ4: Behavioral Shift Analysis

Figure 4 shows that the effect of MMSkills is not merely a success-rate gain. **MMSkills reduce low-level action load.** Gemini 3 Flash uses substantially fewer primitives per task, and Qwen3-VL-235B shows a similar reduction, especially in click actions. This supports the view that multimodal state cards and visual evidence constrain the agent's search space: the agent performs fewer exploratory GUI operations before reaching a useful state. **The behavioral shift is strongest for Qwen3-VL-235B.** Its click share drops from 75.8% to 63.7%, while keyboard and DONE actions increase, suggesting that MMSkills help click-heavy agents move toward more structured input and stronger completion judgments.

MMSkills reduce observed repetitive actions and increase DONE frequency. The effect is clearest for Qwen3-VL-235B: exact repeated actions fall from 21.8% to 6.2%, and the longest same-mode run decreases substantially. Gemini 3 Flash shows the same direction of change, though from a stronger baseline. MMSkills also increase DONE behavior for both models, although this frequency change does not by itself establish correct completion verification. Overall, MMSkills reshape agent behavior from exploratory trial-and-error toward grounded, state-aware execution; Appendix L provides the GLM-5V and Kimi-K2.6 analysis.

4 Related Work

Skills for agents. Recent LLM agents store reusable behavior as language, code, APIs, or learned libraries (Ahn et al., 2022; Liang et al., 2023; Yao et al., 2023; Shinn et al., 2023; Wang et al., 2023a; Zheng et al., 2025; Chen et al., 2026; Wang et al., 2026a; Alzubi et al., 2026; Ma et al., 2026; Xia et al., 2026). A complementary line treats accumulated experience as long-term agent memory (Park et al., 2023; Packer et al., 2024), while surveys and benchmarks evaluate skill relevance, selection, and safety (Xu and Yan, 2026; Li et al., 2026; Wang et al., 2026b; Liu et al., 2026). Memento, Memento 2, Memento-Skills, and Skill-Pro study memory or skill updating (Zhou et al., 2025; Wang, 2026; Zhou et al., 2026; Mi et al., 2026). MMSkills evaluates an offline frozen library and its selective read path. MMSkills follows this modular view but stores state-conditioned multimodal packages and uses branch loading instead of inserting full skill memory; Appendix P expands the discussion.

Visual agents. Visual-agent benchmarks span web, mobile, desktop, and embodied environments (Deng et al., 2023; Zhou et al., 2024; Koh et al., 2024; He et al., 2024; Rawles et al., 2025; Xie et al., 2024; Yang et al., 2025b; Liu et al., 2024), and model and framework work improves screenshot grounding and GUI control (Cheng et al., 2024; Wu et al., 2024; Qin et al., 2025; Agashe et al., 2024; Hong et al., 2024; Zheng et al., 2024; Zhang et al., 2026a; Lu et al., 2024). Dedicated grounding benchmarks measure how reliably models localize UI elements from instructions (Li et al., 2025a; Gou et al., 2025; Wang et al., 2025b; Xu et al., 2025). MMSkills builds on these capabilities but operates higher: it tells the agent which procedural state matters and what visual evidence confirms it.

Closest to our work, Mirage-1 introduces hierarchical multimodal skills, XSkill extracts skills from visually grounded experience, and CUA-Skill represents computer-use skills as parameterized procedures and execution graphs (Xie et al., 2025; Jiang et al., 2026; Chen et al., 2026). MMSkills differs by organizing skills around runtime state cards and multi-view evidence, and by using branch loading to align selected evidence with the live observation before the main agent acts.

5 Conclusion and Limitations

We introduced **MMSkills**, combining textual procedures, runtime state cards, multi-view keyframes, and branch-loaded consultation. The evaluated desktop and game settings show aggregate gains across model families, with ties and application-level regressions. The teacher-generated library is constructed offline and remains frozen during testing; this work does not evaluate same-agent continual learning. Source coverage, generation and grounding errors, misleading evidence, and branch overhead limit its utility. We do not establish transfer across benchmarks, domains, or held-out workflow families, or robustness to controlled changes in resolution, scaling, window arrangement, theme, locale, or application version. The small-model macOSWorld result also shows that external knowledge does not remove limitations in skill selection and GUI execution. Future online updates require disjoint experience, failure attribution, and held-out improvement and regression checks (Appendix I).

Acknowledgements and Disclosures

Funding and support. This work was supported by the National Natural Science Foundation of China under Grant Nos. 62502310 and 62322603. Kangning Zhang, Shuai Shao, and Qingyao Li conducted this work during internships at Xiaohongshu Inc. The authors' affiliations with Xiaohongshu Inc. are disclosed in the author list. The authors report no other third-party support for this work during the last 36 months.

Competing interests. Apart from the funding and employment/internship affiliations disclosed above, the authors declare no competing interests or additional financial relationships outside the submitted work that could be perceived as influencing it.

References

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human, 2024. https://arxiv.org/abs/2410.08164.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. https://arxiv.org/abs/2204.01691.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems, 2026. https://arxiv.org/abs/2603.02766.

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. https://arxiv.org/abs/2511.21631.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. https://arxiv.org/abs/2308.14508.

Tianyi Chen, Yinheng Li, Michael Solodko, Sen Wang, Nan Jiang, Tingyuan Cui, Junheng Hao, Jongwoo Ko, Sara Abdali, Leon Xu, Suzhen Zheng, Hao Fan, Pashmina Cameron, Justin Wagle, and Kazuhito Koishida. Cua-skill: Develop skills for computer using agent, 2026. https://arxiv.org/abs/2601.21123.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents, 2024. https://arxiv.org/abs/2401.10935.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023. https://arxiv.org/abs/2306.06070.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents, 2025. https://arxiv.org/abs/2410.05243.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models, 2024. https://arxiv.org/abs/2401.13919.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents, 2024. https://arxiv.org/abs/2312.08914.

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-bench: How good are llms at playing games?, 2025. https://arxiv.org/abs/2505.15146.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. Xskill: Continual learning from experience and skills in multimodal agents, 2026. https://arxiv.org/abs/2603.12056.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks, 2024. https://arxiv.org/abs/2401.13649.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use, 2025a. https://arxiv.org/abs/2504.07981.

Qingyao Li, Xinyi Dai, Weiwen Liu, Xiangyang Li, Yasheng Wang, Ruiming Tang, Yong Yu, and Weinan Zhang. Atgen: Adversarial reinforcement learning for test case generation, 2025b. https://arxiv.org/abs/2510.14635.

Qingyao Li, Wei Xia, Kounianhua Du, Xinyi Dai, Ruiming Tang, Yasheng Wang, Yong Yu, and Weinan Zhang. Rethinkmcts: Refining erroneous thoughts in monte carlo tree search for code generation, 2025c. https://arxiv.org/abs/2409.09584.

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. 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, 2023. https://arxiv.org/abs/2209.07753.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. https://arxiv.org/abs/2307.03172.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. Visualagentbench: Towards large multimodal models as visual foundation agents, 2024. https://arxiv.org/abs/2408.06327.

Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. How well do agentic skills work in the wild: Benchmarking llm skill usage in realistic settings, 2026. https://arxiv.org/abs/2604.04323.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent, 2024. https://arxiv.org/abs/2408.00203.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver, 2026. https://arxiv.org/abs/2604.08377.

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. https://arxiv.org/abs/2602.01869.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023. https://arxiv.org/abs/2304.03442.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents, 2025. https://arxiv.org/abs/2501.12326.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control, 2023. https://arxiv.org/abs/2307.10088.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2025. https://arxiv.org/abs/2405.14573.

Shuai Shao, Yixiang Liu, Bingwei Lu, and Weinan Zhang. Monoscale: Scaling multi-agent system with monotonic improvement, 2026a. https://arxiv.org/abs/2601.23219.

Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, and Jing Shao. Your agent may misevolve: Emergent risks in self-evolving llm agents, 2026b. https://arxiv.org/abs/2509.26354.

Shuai Shao, Kangning Zhang, Qingyao Li, Shijian Wang, Hao Wang, Wenxiang Jiao, Yuan Lu, Yi Guo, Weiwen Liu, and Weinan Zhang. Harness-r1: Learning to edit executable runtime harnesses from agent failure trajectories, 2026c. https://arxiv.org/abs/2608.02276.

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

V Team, Wenyi Hong, Xiaotao Gu, Ziyang Pan, Zhen Yang, Yuting Wang, Yue Wang, Yuanchang Yue, Yu Wang, Yanling Wang, Yan Wang, Xijun Liu, Wenmeng Yu, Weihan Wang, Wei Li, Shuaiqi Duan, Sheng Yang, Ruiliang Lv, Mingdao Liu, Lihang Pan, Ke Ning, Junhui Ji, Jinjiang Wang, Jing Chen, Jiazheng Xu, Jiale Zhu, Jiale Cheng, Ji Qi, Guobing Gan, Guo Wang, Cong Yao, Zijun Dou, Zihao Zhou, Zihan Wang, Zhiqi Ge, Zhijie Li, Zhenyu Hou, Zhao Xue, Zehui Wang, Zehan Qi, Zehai He, Yutao Zhang, Yusen Liu, Yukuo Cen, Yuchen Li, Yuan Wang, Yu Yang, Yongbin Liu, Yijian Lu, Yifan Xu, Yanzi Wang, Yanxiao Zhao, Yanfeng Wang, Yadong Xue, Yabo Xu, Xinyu Zhang, Xinyu Liu, Xiao Liu, Wenyi Zhao, Wenkai Li, Tianyu Tong, Tianshu Zhang, Shudan Zhang, Shengdong Yan, Qinkai Zheng, Mingde Xu, Licheng Bao, lat Long long, Jiaxing Xu, Jiaxin

Fan, Jiawen Qian, Jiali Chen, Jiahui Lin, Jiadai Sun, Haozhi Zheng, Haoran Wang, Haochen Li, Hanyu Lai, Han Xu, Fan Yang, Dan Zhang, Da Yin, Chuangxin Zhao, Chengcheng Wu, Boyan Shi, Bowen Lv, Bowei Jia, Bo Li, Bin Chen, Baoxu Wang, Peng Zhang, Debing Liu, Bin Xu, Juanzi Li, Minlie Huang, Yuxiao Dong, and Jie Tang. Glm-5v-turbo: Toward a native foundation model for multimodal agents, 2026. https://arxiv.org/abs/2604.26752.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. Skillx: Automatically constructing skill knowledge bases for agents, 2026a. https://arxiv.org/abs/2604.04804.

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, 2023a. https://arxiv.org/abs/2305.16291.

Jun Wang. Memento 2: Learning by stateful reflective memory, 2026. https://arxiv.org/abs/2512.22716.

Leye Wang, Zixing Wang, and Anjie Xu. Skilltester: Benchmarking utility and security of agent skills, 2026b. https://arxiv.org/abs/2603.28815.

Shijian Wang, Jiarui Jin, Runhao Fu, Zexuan Yan, Xingjian Wang, Mengkang Hu, Eric Wang, Xiaoxi Li, Kangning Zhang, Li Yao, Wenxiang Jiao, Xuelian Cheng, Yuan Lu, and Zongyuan Ge. Museagent: A multimodal reasoning agent with stateful experiences, 2026c. https://arxiv.org/abs/2603.27813.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Dikang Du, Hao Hu, Huarong Chen, Zaida Zhou, Haotian Yao, Ziwei Chen, Qizheng Gu, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Flood Sung, Y. Charles, Zhilin Yang, and Tao Yu. Opencua: Open foundations for computer-use agents, 2025a. https://arxiv.org/abs/2508.09123.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, Weiyun Wang, Xiangyu Zhao, Jixuan Chen, Haodong Duan, Tianbao Xie, Chenyu Yang, Shiqian Su, Yue Yu, Yuan Huang, Yiqian Liu, Xiao Zhang, Yanting Zhang, Xiangyu Yue, Weijie Su, Xizhou Zhu, Wei Shen, Jifeng Dai, and Wenhai Wang. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents, 2025b. https://arxiv.org/abs/2507.19478.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models, 2023b. https://arxiv.org/abs/2311.05997.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents, 2024. https://arxiv.org/abs/2302.01560.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. Os-atlas: A foundation action model for generalist gui agents, 2024. https://arxiv.org/abs/2410.23218.

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. https://arxiv.org/abs/2602.08234.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. https://arxiv.org/abs/2404.07972.

Yuquan Xie, Zaijing Li, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Dongmei Jiang, and Liqiang Nie. Mirage-1: Augmenting and updating gui agent with hierarchical multimodal skills, 2025. https://arxiv.org/abs/2506.10387.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward, 2026. https://arxiv.org/abs/2602.12430.

Yibin Xu, Liang Yang, Hao Chen, Hua Wang, Zhi Chen, and Yaohua Tang. Deskvision: Large scale desktop region captioning for advanced gui agents, 2025. https://arxiv.org/abs/2503.11170.

Jingyi Yang, Shuai Shao, Dongrui Liu, and Jing Shao. Riosworld: Benchmarking the risk of multimodal computer-use agents, 2025a. https://arxiv.org/abs/2506.00618.

Pei Yang, Hai Ci, and Mike Zheng Shou. macosworld: A multilingual interactive benchmark for gui agents, 2025b. https://arxiv.org/abs/2506.04135.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems, 2025c. https://arxiv.org/abs/2504.00587.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. https://arxiv.org/abs/2210.03629.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users, 2026a. https://arxiv.org/abs/2312.13771.

Kangning Zhang, Wenxiang Jiao, Kounianhua Du, Yuan Lu, Weiwen Liu, Weinan Zhang, and Yong Yu. Looptool: Closing the data-training loop for robust llm tool calls, 2025. https://arxiv.org/abs/2511.09148.

Kangning Zhang, Haotian Fang, Xukun Luo, Hao Yin, Yang Gao, Peng Yan, Weiwen Liu, Weinan Zhang, and Yong Yu. Learning from unreachable rewards: Hint-conditioned reinforcement learning for generative recommendation, 2026b. https://arxiv.org/abs/2608.11980.

Kangning Zhang, Jiarui Jin, Yingjie Qin, Ruilong Su, Jianghao Lin, Yong Yu, and Weinan Zhang. Motor: Learning id-free item representation with token crossing for embedding-based multimodal recommendation, 2026c. https://arxiv.org/abs/2410.19276.

Kangning Zhang, Yixing Li, Shuai Shao, Qingyao Li, Zhengxi Lu, Zhiyuan Yao, Jianghao Lin, Wenxiang Jiao, Yuan Lu, Weiwen Liu, Weinan Zhang, and Yong Yu. Vad: Attributing visual evidence for target reconstruction in multimodal on-policy distillation, 2026d. https://arxiv.org/abs/2607.28590.

Kangning Zhang, Yingjie Qin, Weinan Zhang, Yong Yu, and Jianghao Lin. Diffcold: A diffusion-based generative model for cold-start item recommendation, 2026e. https://arxiv.org/abs/2606.12245.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024. https://arxiv.org/abs/2401.01614.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills, 2025. https://arxiv.org/abs/2504.07079.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning llm agents without fine-tuning llms, 2025. https://arxiv.org/abs/2508.16153.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. Memento-skills: Let agents design agents, 2026. https://arxiv.org/abs/2603.18743.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. https://arxiv.org/abs/2307.13854.

A Benchmark Statistics

We use four visual-agent benchmarks. **OSWorld** is the primary GUI benchmark and contains Ubuntu desktop tasks across browsers, office software, creative tools, media applications, system settings, code editors, email, and multi-application workflows (Xie et al., 2024). **macOSWorld** provides an auxiliary cross-operating-system GUI evaluation with file management, media, productivity, system/interface, and system-application tasks (Yang et al., 2025b). **VAB-Minecraft** is the Minecraft subset of VisualAgentBench and evaluates item-acquisition tasks that require visual grounding, inventory tracking, recipe reasoning, tool use, and handling failed actions (Liu et al., 2024). **LMGame-Bench** evaluates game-playing agents through a unified interface (Hu et al., 2025); we use Super Mario Bros because its recurring visual situations naturally align with reusable multimodal skills.

Table 5: Test-case distributions for OSWorld and macOSWorld. OSWorld contains 360 test cases; macOSWorld contains 143 test cases. "Share" is the percentage of test cases in each domain within the corresponding benchmark.

Benchmark	Domain	Count	Share	Snapshot-en	Snapshot-apps
OSWorld	Multi-app	93	25.83	–	–
OSWorld	LibreOffice Calc	47	13.06	–	–
OSWorld	LibreOffice Impress	47	13.06	–	–
OSWorld	Chrome	45	12.50	–	–
OSWorld	GIMP	26	7.22	–	–
OSWorld	OS	24	6.67	–	–
OSWorld	LibreOffice Writer	23	6.39	–	–
OSWorld	VS Code	23	6.39	–	–
OSWorld	VLC	17	4.72	–	–
OSWorld	Thunderbird	15	4.17	–	–
macOSWorld	File management	29	20.28	29	0
macOSWorld	Media	12	8.39	0	12
macOSWorld	Productivity	35	24.48	16	19
macOSWorld	System and interface	29	20.28	29	0
macOSWorld	System apps	38	26.57	38	0

B Skill Source Statistics

All MMSkills are extracted from non-test trajectories. For OSWorld and macOSWorld, we use the Ubuntu and macOS subsets of OpenCUA trajectories as GUI skill sources (Wang et al., 2025a). For macOS, the raw OpenCUA trajectories do not directly follow the five macOSWorld categories; we therefore perform additional clustering and relevance filtering before assigning trajectories to the analysis categories below.

For VAB-Minecraft, we use the official training set as the source for extracting multimodal skill packages. For Super Mario Bros from LMGame-Bench, MMSkills are extracted from multiple runs over four source cases. In both settings, the skill-source data are disjoint from the final evaluation cases.

Table 6: OpenCUA trajectory statistics used for GUI skill extraction. "Tasks" counts source trajectories, "Share" is the within-platform percentage, and "Clusters" is the number of Phase-0 semantic trajectory clusters used for downstream skill planning.

Platform	Domain	Tasks	Share	Clusters
Ubuntu	Chrome	718	17.1	17
Ubuntu	LibreOffice Impress	605	14.4	11
Ubuntu	VS Code	605	14.4	4
Ubuntu	OS	497	11.8	2
Ubuntu	GIMP	492	11.7	14
Ubuntu	LibreOffice Writer	490	11.7	3
Ubuntu	Thunderbird	300	7.1	11
Ubuntu	LibreOffice Calc	298	7.1	3
Ubuntu	VLC	200	4.8	8
macOS	Productivity	1,424	45.1	20
macOS	System apps	768	24.3	11
macOS	File management	341	10.8	9
macOS	Media	315	10.0	7
macOS	System and interface	309	9.8	12

Table 7: OSWorld MMSkill package statistics. "#Skills" counts unique skill packages, while "Skills/Task" reports the average number of skill matches assigned to evaluation tasks and therefore need not equal #Skills/#Tasks. Word statistics are median/mean over skill procedures. "Full/Focus" and "Before/After" report counts of those view types; "Transition Cards" counts state cards with at least one before/after transition view, with percentages over state cards. The Total/Avg row reports total counts and weighted averages; † marks a fitted value estimated from domain-level medians.

Domain	#Tasks	#Skills	Skills/Task	Words Med/Mean	#Cards	Cards/Skill	#Views	Views/Card	Full/Focus	Before/After	Transition Cards
Chrome	45	34	1.20	653 / 630.9	134	3.94	292	2.18	134/134	13/11	24 (17.9%)
GIMP	26	26	1.19	470 / 400.2	77	2.96	190	2.47	77/77	14/22	36 (46.8%)
Calc	47	26	1.36	278 / 278.1	79	3.04	184	2.33	79/79	7/19	26 (32.9%)
Impress	47	20	1.32	498 / 466.2	60	3.00	140	2.33	60/60	1/19	20 (33.3%)
Writer	23	23	1.13	264 / 289.2	71	3.09	144	2.03	71/71	1/1	2 (2.8%)
Multi-apps	93	20	1.19	574 / 502.0	82	4.10	164	2.00	82/82	0/0	0 (0.0%)
OS	24	37	1.21	544 / 539.8	139	3.76	283	2.04	139/139	5/0	5 (3.6%)
Thunderbird	15	25	1.20	508 / 542.5	87	3.48	192	2.21	87/84	6/15	21 (24.1%)
VLC	17	18	1.00	260 / 275.3	61	3.39	122	2.00	61/61	0/0	0 (0.0%)
VS Code	23	18	1.09	391 / 389.3	89	4.94	187	2.10	89/89	9/0	9 (10.1%)
Total / Avg.	**360**	**247**	**1.21**	**498.0† / 447.8**	**879**	**3.56**	**1898**	**2.16**	**879/876**	**56/87**	**143 (16.3%)**

Table 8: macOSWorld MMSkill package statistics. "#Skills" counts unique skill packages, while "Skills/Task" reports the average number of skill matches assigned to evaluation tasks. Word statistics are median/mean over skill procedures. "Full/Focus" and "Before/After" report counts of those view types; "Transition Cards" counts state cards with at least one before/after transition view, with percentages over state cards.

Domain	#Tasks	#Skills	Skills/Task	Words Med/Mean	#Cards	Cards/Skill	#Views	Views/Card	Full/Focus	Before/After	Transition Cards
File management	29	30	1.03	358 / 374.5	62	2.07	128	2.06	62/62	4/0	4 (6.5%)
Media	12	25	2.08	378 / 400.8	55	2.20	116	2.11	55/55	6/0	6 (10.9%)
Productivity	35	59	1.69	324 / 330.2	125	2.12	261	2.09	125/125	11/0	11 (8.8%)
System/interface	29	88	3.03	282 / 285.5	182	2.07	380	2.09	182/182	16/0	16 (8.8%)
System apps	38	46	1.21	347 / 352.0	98	2.13	212	2.16	98/98	6/10	16 (16.3%)
Total / Avg.	**143**	**248**	**1.73**	**324 / 330.9**	**522**	**2.10**	**1097**	**2.10**	**522/522**	**43/10**	**53 (10.2%)**

Table 9: Game benchmark MMSkill package statistics. Word statistics are median/mean over skill procedures and plans. "Full/Focus" and "Before/After" report counts of those view types; "Transition Cards" counts state cards with at least one before/after transition view, with percentages over state cards. † marks a fitted value estimated from the available before/after view counts.

Benchmark	#Skills	Skill Words Med/Mean	Plan Words Med/Mean	#Cards	Cards/Skill	#Views	Views/Card	Full/Focus	Before/After	Transition Cards
VAB-Minecraft	24	278.5 / 281.7	68.0 / 68.4	79	3.29	185	2.34	79/79	8/19	20 (25.3%)
Super Mario Bros	10	374.0 / 370.8	280.0 / 271.0	34	3.40	48†	1.41†	34/0	5/9	14 (41.2%)†

C Experiment Details

Across all evaluations, agents plan from visual environment observations rather than privileged state, using desktop screenshots for GUI tasks and game screenshots for game tasks. For OSWorld and macOSWorld, we run the full evaluations primarily on Amazon Web Services using the official benchmark images and task definitions. The agent interacts through the benchmark harness, and we use a maximum interaction budget of 20 steps for both GUI benchmarks. VAB-Minecraft and Super Mario Bros follow their official evaluation protocols.

For VAB-Minecraft, we use the official test set for evaluation. The training trajectories described in Appendix B are used only to generate reusable procedures, state cards, and keyframes; no test episodes are used during skill construction.

For Super Mario Bros from LMGame-Bench, we split the available game cases into disjoint source and evaluation subsets. The source cases are described in Appendix B, while a separate set of four held-out cases is used for final evaluation. This separation ensures that the generated skills capture reusable game situations rather than memorizing the measured episodes.

We evaluate both frontier and smaller multimodal models: Gemini 3.1 Pro, Gemini 3 Flash[1], Qwen3-VL-235B-A22B-Thinking (Bai et al., 2025), GLM-5V-Turbo (Team et al., 2026), Kimi-K2.6, and Qwen3-VL-8B-Instruct (Bai et al., 2025). For each base model, we compare *no-skill*, *text-only skill*, and *MMSkills* conditions. Unless otherwise stated, skill conditions use branch loading: text-only skills use the same branch mechanism without state cards or images, while MMSkills inspect selected state cards and multi-view keyframes before returning structured guidance to the main agent. Direct text-skill loading and direct multimodal loading are evaluated only as ablation variants.

D Source and Model Provenance

Table 10 distinguishes source collection, skill generation, and downstream evaluation. The same teacher-generated library is consumed by heterogeneous test agents and remains frozen during evaluation. "Codex" names the product used for construction, rather than a separately specified model checkpoint; no finer model identifier is available in the reported configuration.

Table 10: Stage-wise provenance. All rows use Codex with the meta-skill for construction and Qwen3-VL-235B-A22B-Thinking for visual grounding. Test-agent abbreviations: G = Gemini 3 Flash; Q = Qwen3-VL-235B-A22B-Thinking; Q8 = Qwen3-VL-8B-Instruct; V = GLM-5V-Turbo.

Benchmark	Source trajectories	Source rollout	Test agents
OSWorld	OpenCUA/AgentNet Ubuntu (4,205)	Human demonstrations	G, Q, Q8, V; Gemini 3.1 Pro; Kimi-K2.6
macOSWorld	OpenCUA/AgentNet macOS (3,157)	Human demonstrations	G, Q, Q8, V
VAB-Minecraft	Official VAB training split	VAB LMM bootstrapping (e.g., GPT-4o)	G, Q, Q8, V
Super Mario Bros	Four non-evaluation LMGame cases	Gemini 3 Flash	G, Q, Q8, V

Construction and live matching. The Codex agent selects decision-relevant keyframes using textual trajectory information rather than fixed-rate frame sampling. Qwen3-VL-235B-A22B-Thinking localizes a state cue or action target to produce a focus crop, retaining the full frame for context. The meta-skill checks schema completeness, procedural executability, card–keyframe consistency, and image references, flagging or repairing issues before release. These construction checks do not determine applicability at test time: the evaluated model decides whether to invoke a skill, which state cards and views to inspect, and how to align them with the live observation.

E Matched Baselines

We compare additional branch reasoning without external experience (Self-deliberation), retrieval of raw trajectories (Raw-RAG), and structured textual skills adapted from XSkill (Jiang et al., 2026).

[1] `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf`

Raw-RAG, adapted XSkill, and MMSkills use the same non-test OpenCUA source pool, test agents, and two-stage runtime protocol. The XSkill row is a matched adaptation, not the original system's published result. No skill uses only the main agent. The text-only MMSkills control retains D and P with $S = K = \emptyset$ and uses the same branch protocol as MMSkills.

Mirage-1 combines hierarchical multimodal skills with search (Xie et al., 2025), and CUA-Skill uses parameterized procedures and execution graphs (Chen et al., 2026). Their original mobile/web and Windows settings, respectively, differ from Ubuntu OSWorld. We do not treat their published scores as matched desktop baselines. Table 11 gives the application-level results underlying Table 3.

Table 11: Matched OSWorld application-level success (%). XSkill denotes the adapted textual-skill control.

Model	Setting	Chrome	GIMP	Calc	Impress	Writer	Multi	OS	Mail	VLC	VS Code	Overall
Gemini 3 Flash	No skill	37.78	50.00	38.30	29.73	52.17	21.51	54.17	66.67	52.39	47.83	36.65
	Self-deliberation	43.50	36.00	40.43	32.00	53.50	24.00	58.33	66.67	52.94	52.17	39.41
	Raw-RAG	42.16	39.25	41.32	32.84	52.91	24.63	61.25	62.18	51.76	53.42	40.52
	XSkill	43.85	40.37	42.74	34.16	53.46	25.72	63.18	63.04	52.12	54.63	41.63
	MMSkills	55.37	42.31	53.19	40.34	56.52	30.98	75.00	66.67	52.94	60.87	**47.97**
Qwen3-VL 235B	No skill	15.56	38.46	17.02	25.53	43.48	9.48	25.00	26.67	17.65	34.78	21.34
	Self-deliberation	26.64	46.15	18.62	27.15	44.56	11.95	29.17	38.33	23.53	40.22	25.80
	Raw-RAG	40.84	56.00	20.66	29.22	45.95	15.11	34.50	53.27	31.06	47.17	31.50
	XSkill	41.63	57.12	21.08	29.68	46.34	15.63	35.28	54.01	31.72	48.02	32.45
	MMSkills	59.91	69.23	23.40	32.01	47.82	19.35	41.67	73.33	41.18	56.52	**39.17**

F Inference-Time Accounting

Table 12 reports per-task OSWorld measurements across the three primary conditions and two agents. Model calls include the inference used during evaluation, and branch invocations count skill consultations. Image instances count repeated image presentations, rather than unique source images. Overall tokens include text and image accounting and should not be interpreted as interchangeable across providers. API costs reflect the provider accounting for these runs, not a forecast at future prices. Wall-clock values report the mean and standard deviation across tasks. These measurements exclude offline skill construction and do not constitute total project compute.

MMSkills reduces measured latency and environment steps for both agents, while model calls and text context increase. Relative to no skill, Gemini's API cost increases by approximately 1.9% and Qwen's decreases by approximately 6.8%. Thus, efficiency depends on the model and accounting metric; branch consultation is not computationally free.

Table 12: Complete six-setting inference accounting on OSWorld. Values are per task. Time is mean $\pm$ across-task standard deviation, in seconds; k denotes thousands of tokens.

Gemini 3 Flash			
Metric	No skill	Text-only	MMSkills
Success (%)	36.65	40.27	47.97
Wall-clock time (s)	216.99 ± 142.35	278.48 ± 91.33	192.64 ± 133.43
Environment steps	13.11	16.64	11.86
Model calls	16.01	17.25	17.59
Branch invocations	–	0.71	0.96
Input text tokens	13.82k	15.80k	16.32k
Output tokens	0.68k	0.77k	1.11k
Image instances	57.06	58.25	57.42
Overall tokens (text + image)	77.73k	81.04k	80.63k
API cost (USD)	1.409	1.428	1.436
Qwen3-VL-235B			
Metric	No skill	Text-only	MMSkills
Success (%)	21.34	28.57	39.17
Wall-clock time (s)	257.38 ± 145.86	232.51 ± 99.44	215.04 ± 108.55
Environment steps	15.22	13.34	9.87
Model calls	17.29	17.87	17.55
Branch invocations	–	0.49	0.92
Input text tokens	17.33k	18.93k	20.38k
Output tokens	0.69k	0.89k	1.01k
Image instances	64.20	57.87	56.09
Overall tokens (text + image)	148.43k	137.10k	134.92k
API cost (USD)	0.311	0.292	0.290

G Frozen-Library Quality Audit

We audited the complete frozen OSWorld library of 247 skills along six dimensions (Table 13). Six procedure texts were flagged for minor refinement; the reported evaluation uses the frozen library, not a library retroactively repaired using test outcomes. Package integrity, card completeness, and reference checks establish structural properties, while semantic alignment is an audit judgment rather than a guarantee of correct runtime use. Task-skill coverage records the presence of candidate references for evaluation tasks; it does not establish successful applicability or held-out-workflow generalization. The image-file count and multi-view-keyframe count have different scopes and are reported separately.

Table 13: Audit of the frozen OSWorld skill library.

Criterion	Scope	Result
Package/schema validity	247 skills	247/247 valid
Procedure-text quality	247 skills	241/247 passed; 6 flagged
Runtime-card completeness	879 cards	879/879 complete
Card–keyframe relevance	879 cards	1,898 semantically matched multi-view keyframes
Image/reference integrity	1,910 images	1,910/1,910 readable; 0 broken references
Task-skill coverage	360 tasks	360/360 covered; 0 missing references

H Failure Attribution and Heterogeneous Effects

Using Qwen3-VL-235B-A22B-Thinking, we reviewed all 293 failed OSWorld trajectories from Gemini 3 Flash and Qwen3-VL-235B that invoked MMSkills. Table 14 assigns each audited episode to a reported failure category. This is a model-assisted diagnostic analysis of a selected set of failures, not an independently validated causal attribution over all episodes. No state-card mismatch was assigned in this audit; this does not establish that such failures cannot occur.

The largest category is execution/verification after usable guidance (173/293, 59.0%). Misleading keyframes (55 episodes) and branch-guidance errors (57) nevertheless show that the skill-consultation

Table 14: Failure attribution among OSWorld episodes that invoked MMSkills and failed.

Failure category	Gemini ($N = 129$)	Qwen ($N = 164$)
Skill retrieval/applicability	2	6
State-card mismatch	0	0
Misleading keyframes	24	31
Branch guidance error	23	34
Execution/verification after usable guidance	80	93

path can fail. These observations are consistent with application-level negative transfer and do not support a claim of monotonic improvement.

Gemini 3.1 Pro text-only regressions. The diagnostic review identified 42 text-only regressions, of which ten occurred without consultation. Among the other 32, the audit assigned 24 to execution/recovery, seven to premature completion/verification, and one to applicability. Because this condition contains no keyframes, these regressions cannot be attributed to reference-image anchoring; procedural advice and rollout variation remain possible contributors. These diagnostic counts do not constitute a significance test for the aggregate success-rate table.

Qwen3-VL-8B on macOSWorld. MMSkills ties the no-skill overall score at 6.29%. External knowledge still requires the agent to invoke a relevant skill, interpret evidence, and execute GUI actions. The observed tie shows that the package alone does not overcome limitations of the small-model/harness configuration; we do not infer a component-specific causal explanation without a controlled intervention.

I Scope and Prospective Online Extension

The present contribution is a multimodal representation, offline construction pipeline, and selective runtime read mechanism. Each domain has its own library. Disjoint source and evaluation episodes support reuse within the evaluated domains, but do not establish cross-benchmark, cross-domain, or held-out-workflow transfer. Browser tasks within OSWorld also do not replace evaluation on dedicated web benchmarks. Controlled changes in UI resolution, scaling, arrangement, theme, locale, and application version remain untested.

An online extension would need to distinguish retrieval/applicability, evidence grounding, branch planning, and execution/verification errors before editing a skill. It should collect experience from a disjoint task stream, propose localized updates to $D/P/S/K$, and admit them only after held-out improvement and unrelated-task regression checks. Deduplication, conflict resolution, versioning, and retirement of stale cards/images are also necessary to avoid reinforcing incorrect experience. The failure audit identifies possible repair targets but does not implement or evaluate this update loop.

J Branch-Loaded Runtime Algorithm

Algorithm 1 summarizes the branch-loaded runtime loop. Candidate skills are selected before task execution, while branch calls occur only when the main agent decides to consult a specific skill. The main trajectory receives the structured guidance G_t rather than the full multimodal skill package.

Algorithm 1 Branch-loaded MMSkill Agent

Require: Skill library $\mathcal{M}$, task instruction I, visual environment `Env`

Initialize history $H_0 \leftarrow \emptyset$
Pre-recall candidate skills $\mathcal{C}_I \leftarrow \text{PreRecall}(I, \mathcal{M})$
for $t = 1, 2, \ldots$ **do**
    Observe current visual observation O_t from `Env`
    Main agent chooses either action A_t or skill request $M_t \in \mathcal{C}_I$
    if the main agent chooses action A_t **then**
        Execute A_t in `Env` and update H_t
    else
        Unpack $M_t = (D_t, P_t, S_t, K_t)$
        Stage 1: $(J_t, R_t) \leftarrow \text{SelectViews}(O_t, H_{t-1}, P_t, S_t)$
        Load $V_t \leftarrow \{K_j^v : j \in J_t,\ v \in R_{t,j}\}$
        Stage 2: $G_t \leftarrow \text{PlanBranch}(O_t, H_{t-1}, P_t, \{S_j : j \in J_t\}, V_t)$
        Choose grounded action $A_t \leftarrow \pi_{\text{main}}(O_t, H_{t-1}, G_t)$
        Execute A_t in `Env` and update H_t
    end if
    if the task is verified complete **then**
        return success
    end if
end for

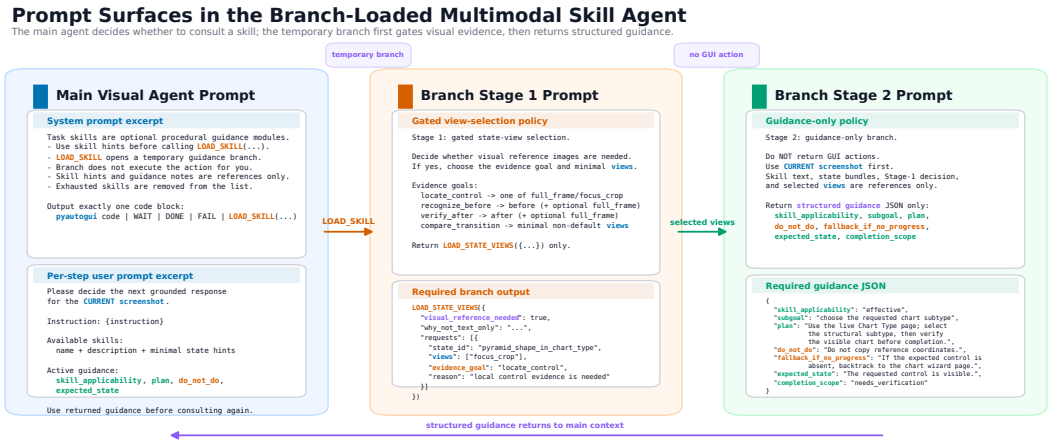

Figure 5: Prompt surfaces used by the branch-loaded multimodal skill agent. The main agent prompt decides whether to act directly or consult a skill branch, Stage 1 selects the relevant state cards and keyframe views, and Stage 2 returns compact structured guidance to the main agent.

K MMSkillAgent Prompt Interface

The main-agent prompt decides whether to act directly or invoke `LOAD_SKILL`. The first branch stage selects state cards and views using the live observation; the second returns applicability, subgoal, plan, negative constraints, and verification guidance. Figure 5 summarizes these surfaces. Full prompt templates and their JSON interfaces are provided in the separate *Supplementary Prompt Templates* document and the public repository, at a readable text size.

L Additional Behavioral Shift Analysis

Figure 6 complements Figure 4 with the same OSWorld behavioral analysis for GLM-5V and Kimi-K2.6.

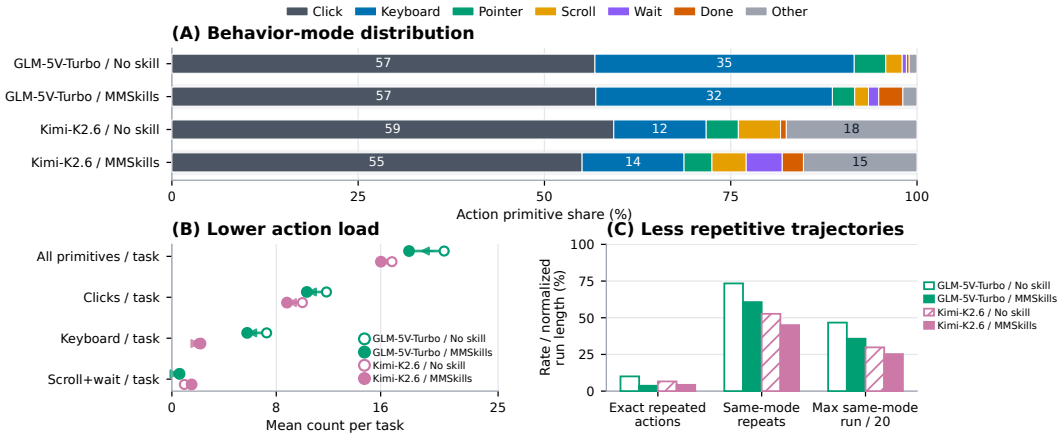

Figure 6: Behavioral shifts induced by MMSkills on OSWorld for GLM-5V and Kimi-K2.6. The panels follow the same metrics as Figure 4: action primitive distribution, low-level primitives per task, and repetitive behavior statistics.

M Broader Impact

MMSkills are intended to make visual agents more reliable by externalizing reusable multimodal procedural knowledge. Potential benefits include improved desktop automation, reduced repeated trial-and-error interactions, better support for smaller models, and more reusable agent knowledge across GUI and game-like visual environments. At the same time, more capable visual agents may also increase the risk of unwanted automation, misuse in interactive software, or accidental actions in sensitive environments. Multimodal skill packages can also contain screenshots or cropped visual evidence, so their construction should avoid private or proprietary user data unless appropriate consent, filtering, and access controls are in place. In this work, we construct skills from public non-evaluation trajectories and store compact state evidence rather than raw demonstrations whenever possible. Future deployments should combine MMSkills with permission controls, task-level safety policies, sensitive-information filtering, and auditing of generated skill packages before they are made available to autonomous agents.

N Use of LLMs

Large language models are used in this work as both research artifacts and research assistants. Methodologically, LLM-based agents are used in the skill-generation pipeline to process and filter trajectories, propose reusable procedures, draft state cards, and generate multimodal skill packages under human-designed schemas and quality checks. LLMs also serve as the evaluated visual agents in the benchmark results. In addition, LLM tools were used during manuscript preparation for editing, polishing, and organizing written content. The authors remained responsible for experimental design, result interpretation, citation checking, and final paper content.

O Interaction Case Studies

Figures 7 and 8 show two representative OSWorld interaction traces. The first case illustrates a LibreOffice Calc workflow in which the agent consults different spreadsheet skills at different stages of table construction. The second case illustrates a terminal file-organization task where branch guidance helps move past an initially brittle command and then verifies the final archive structure.

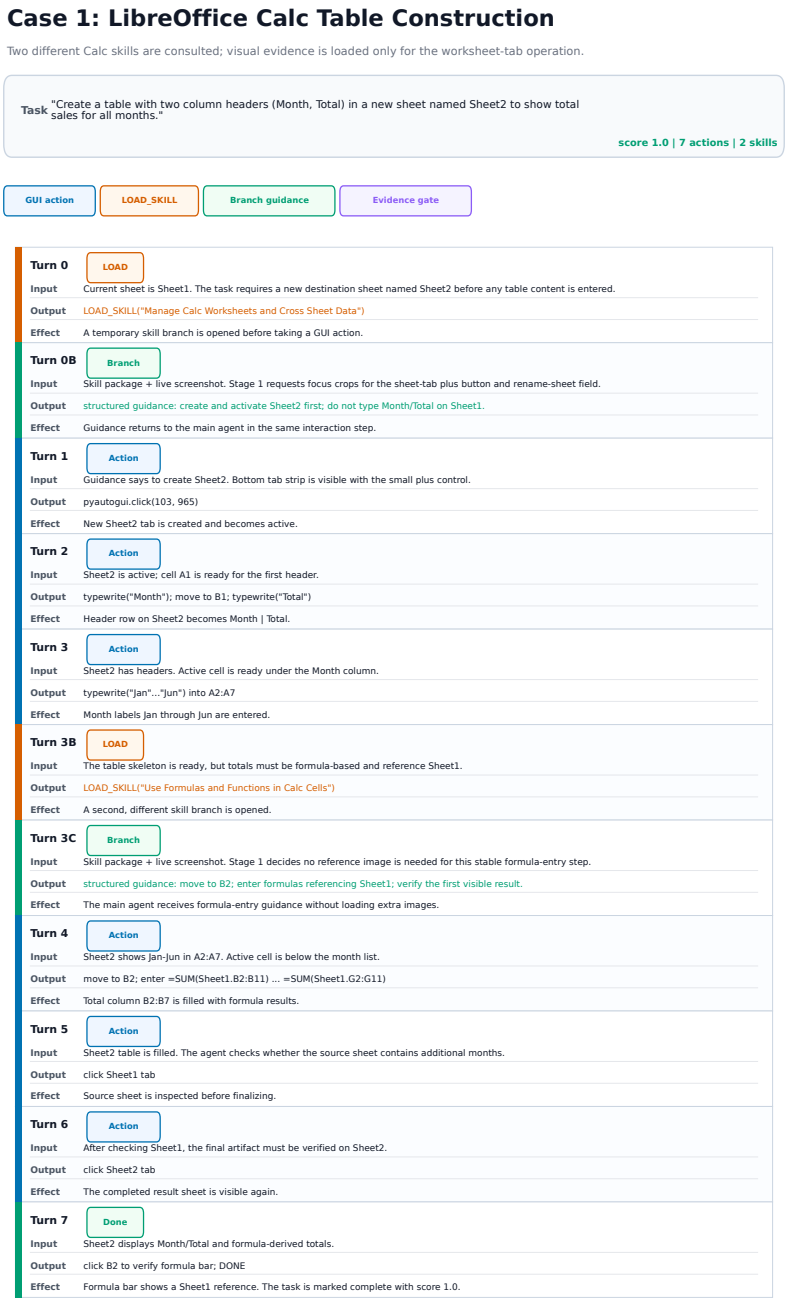

Figure 7: Representative interaction case with branch-loaded MMSkills: LibreOffice Calc table construction. Colored turn labels distinguish direct GUI actions, skill loading, branch guidance, evidence-gated reasoning, and final completion.

Case 2: Terminal File Organization and Compression

Two different OS terminal skills are consulted; the branch gates off images and returns command-level guidance.

Task "Compress files older than 30 days into old_files, and move all other files into new_files."

score 1.0 | 8 actions | 2 skills

GUI action | LOAD_SKILL | Branch guidance | Evidence gate

Turn 0	Action	
	Input	Terminal is active. The task requires separating files by modification time into old_files and new_files.
	Output	mkdir -p /tmp/test_files/old_files /tmp/test_files/new_files
	Effect	Both destination directories are created.
Turn 1	Action	
	Input	Destination directories exist. The agent starts moving files older than 30 days.
	Output	find ... -mtime +30 -exec mv {} old_files/ \;
	Effect	The shell command needs correction because the exec terminator is fragile in typed terminal input.
Turn 2	LOAD	
	Input	Recent command did not make reliable progress. The issue is file movement in a terminal.
	Output	LOAD_SKILL("Modify Files and Folders in Terminal")
	Effect	A first temporary branch is opened for command-repair guidance.
Turn 2B	Branch	
	Input	Skill package + live terminal screenshot. Stage 1 gates off reference images because textual terminal context is sufficient.
	Output	structured guidance: correct the failed find command; avoid repeating the unescaped semicolon path.
	Effect	Guidance returns to the main agent before the next typed command.
Turn 3	Action	
	Input	The branch identifies the current failure mode: fragile find -exec syntax.
	Output	retry find with a quoted semicolon terminator
	Effect	The agent attempts a repaired command path.
Turn 4	Action	
	Input	The semicolon form remains brittle in the terminal interaction.
	Output	find ... -mtime +30 -exec mv -t old_files/ {} +
	Effect	A robust find command moves old files into old_files.
Turn 5	LOAD	
	Input	Old files have been moved. The remaining subgoal is archive creation and final verification.
	Output	LOAD_SKILL("Archive, Extract, and Convert Files in Terminal")
	Effect	A second, different skill branch is opened.
Turn 5B	Branch	
	Input	Skill package + live terminal screenshot. Stage 1 again gates off reference images for a text-only shell workflow.
	Output	structured guidance: run ls -R, compress old_files, then verify the directory tree.
	Effect	The main agent receives archive-and-verification guidance.
Turn 6	Action	
	Input	The next checkpoint is to inspect which files are in each directory.
	Output	ls -R /tmp/test_files
	Effect	The terminal prints the current file distribution.
Turn 7	Action	
	Input	old_files contains the target old files and is ready to be archived.
	Output	tar -czf old_files.tar.gz -C old_files . --remove-files; ls -R
	Effect	The old-file archive is created and the final directory tree is verified.
Turn 8	Done	
	Input	The terminal output shows the archive and the expected file organization.
	Output	DONE
	Effect	The task is marked complete with score 1.0.

Figure 8: Representative interaction case with branch-loaded MMSkills: terminal file organization and compression. Colored turn labels distinguish direct GUI actions, skill loading, branch guidance, evidence-gated reasoning, and final completion.

P Detailed Related Work

This section provides the expanded related-work discussion summarized in Section 4.

Skills for agents. Recent LLM agents have made skills a practical interface for storing and composing procedural knowledge in language-conditioned environments. Early systems connected language models to action by grounding language in affordances (Ahn et al., 2022), emitting executable programs (Liang et al., 2023), or interleaving reasoning and acting (Yao et al., 2023); adjacent code- and tool-agent work studies robust tool-call data loops, search-based code refinement, and adversarial test-case generation (Zhang et al., 2025; Li et al., 2025c,b). Harness-R1 learns failure-conditioned edits to executable runtime harnesses (Shao et al., 2026c); this changes the agent runtime, while MMSkills supplies external procedural evidence to a fixed runtime. Reflection mechanisms then made agent behavior more persistent across attempts (Shinn et al., 2023). In open-ended environments, systems such as DEPS, Voyager, and JARVIS-1 showed that large models can use language, stored experience, and self-generated programs to acquire or reuse behaviors over extended task horizons (Wang et al., 2024, 2023a,b). These works motivate our focus on procedural reuse, but their reusable knowledge is primarily textual, symbolic, or programmatic.

More recent work treats skills as an explicit substrate for agent improvement. SkillWeaver distills web exploration into reusable API-like skills (Zheng et al., 2025); CUA-Skill builds a parameterized skill base with execution and composition graphs for computer-using agents (Chen et al., 2026); SkillX automatically constructs hierarchical skill knowledge bases from agent experience (Wang et al., 2026a); EvoSkill studies automated skill discovery through failure analysis in multi-agent settings (Alzubi et al., 2026), where decentralized coordination and scalable improvement are also central concerns (Yang et al., 2025c; Shao et al., 2026a); SkillClaw evolves shared skills from multi-user trajectories (Ma et al., 2026); and SkillRL co-evolves a hierarchical skill library with reinforcement learning (Xia et al., 2026). A recent survey frames agent skills as portable packages of instructions, code, and resources loaded through progressive disclosure (Xu and Yan, 2026). A complementary perspective treats accumulated agent experience as long-term memory: Generative Agents maintain a memory stream that supports recall, reflection, and planning (Park et al., 2023), while MemGPT introduces an OS-style memory hierarchy that pages information in and out of the model's working context (Packer et al., 2024). MMSkills follows this broader move toward modular procedural knowledge, but changes the unit being stored: instead of treating skills mainly as text, code, APIs, or execution graphs, we define a skill package whose central evidence is a set of visually grounded runtime states. Branch loading also takes inspiration from memory-paging ideas, by inspecting selected multimodal evidence in a temporary branch rather than flooding the main context.

This emerging ecosystem has also motivated dedicated evaluation of skill utility. SkillsBench measures how skills affect agent performance across diverse tasks (Li et al., 2026), SkillTester evaluates utility and security risks of agent skills (Wang et al., 2026b), and recent work studies skill usage under more realistic retrieval and adaptation settings (Liu et al., 2026). These benchmarks show that skills are not automatically beneficial; their value depends on relevance, compactness, selection, and safe use, especially as self-evolving agents may introduce emergent risks (Shao et al., 2026b). Our work addresses a complementary question for visual agents: what evidence should a skill expose, and how should that evidence be loaded, when correct use depends on the current visual state?

The closest line to our work is multimodal and GUI-specific skill augmentation. Mirage-1 introduces hierarchical multimodal skills for GUI agents and uses them with search to support long-horizon control (Xie et al., 2025); XSkill continually extracts experiences and skills for multimodal agents from visually grounded rollouts (Jiang et al., 2026); MuSEAgent studies stateful experiences for multimodal reasoning agents (Wang et al., 2026c); and CUA-Skill builds computer-use skills as parameterized procedures and execution graphs (Chen et al., 2026). MMSkills organizes the artifact around runtime state cards and multi-view evidence, and selectively reads these cues through a temporary branch aligned with the live observation.

Visual agents. Visual agents have rapidly advanced from web navigation to general computer use. Benchmarks such as Mind2Web and WebArena established realistic web-agent evaluation beyond synthetic interfaces (Deng et al., 2023; Zhou et al., 2024); VisualWebArena showed that many web tasks require visual grounding rather than text-only reasoning (Koh et al., 2024); and WebVoyager demonstrated end-to-end web interaction with large multimodal models on real websites (He et al.,

2024). The same trend appears in mobile, desktop, and embodied settings: Android in the Wild and AndroidWorld study device control from visual UI observations (Rawles et al., 2023, 2025), OSWorld and macOSWorld evaluate agents in real operating-system environments (Xie et al., 2024; Yang et al., 2025b), RiOSWorld evaluates risks in multimodal computer-use agents (Yang et al., 2025a), and VisualAgentBench includes VAB-Minecraft and VAB-OmniGibson for open-world and household embodied interaction (Liu et al., 2024).

Model and framework work has likewise moved toward visually grounded action. SeeClick trains GUI grounding for screenshot-only agents (Cheng et al., 2024); CogAgent introduces a visual language model dedicated to GUI understanding and operation (Hong et al., 2024); OS-ATLAS learns a foundation action model for GUI control (Wu et al., 2024); UI-TARS develops native GUI agents that perceive screenshots and emit keyboard/mouse actions (Qin et al., 2025); SeeAct builds web agents around general-purpose vision-language models (Zheng et al., 2024); AppAgent learns smartphone skills from on-device demonstrations (Zhang et al., 2026a); OmniParser provides a pure-vision parser that turns screenshots into structured GUI elements (Lu et al., 2024); and Agent S provides a general computer-use framework built around GUI interaction (Agashe et al., 2024). These systems improve perception and action; MMSkills provides external evidence about procedural state, applicability, and verification.

VAD estimates the visual-evidence component of teacher corrections through counterfactual interventions and uses it to reconstruct distillation targets (Zhang et al., 2026d). It addresses training-time supervision; MMSkills instead packages state-conditioned evidence for inference-time consultation without updating the agent's parameters.

Content-based representations and learning. Recommendation research also studies how content and auxiliary guidance support learning. MOTOR learns ID-free item representations from shared multimodal tokens (Zhang et al., 2026c); DiffCold generates cold-start item embeddings through content-conditioned diffusion (Zhang et al., 2026e); and HCGRec uses target-prefix hints to recover learning signals in reward-sparse generative recommendation (Zhang et al., 2026b). These methods organize representations or supervision for recommendation, while MMSkills organizes visual evidence into external procedures consulted during agent interaction.

GUI grounding benchmarks. Alongside task-completion benchmarks, a separate line of work measures how reliably GUI agents can localize UI elements from natural-language instructions. ScreenSpot-Pro extends earlier ScreenSpot evaluations to high-resolution, professional desktop environments, where target elements often occupy less than 0.1% of the screen and the strongest grounding models still fall well below human performance (Li et al., 2025a). Gou et al. (2025) push toward universal visual grounding that lets agents identify GUI elements purely from screenshots, in the spirit of how humans navigate digital interfaces. MMBench-GUI organizes evaluation hierarchically, from content understanding and element grounding to task automation and multi-agent collaboration (Wang et al., 2025b), and DeskVision contributes a large-scale desktop dataset and evaluation suite that broadens grounding research across operating systems (Xu et al., 2025). These benchmarks isolate the perceptual layer of visual agents. MMSkills is complementary: rather than improving where to click, it provides procedural and visual evidence about which state matters at each step, and lets the underlying grounding capability translate that evidence into precise actions.

Long-context reliability. Recent studies have shown that simply enlarging the context window does not guarantee that all evidence is used effectively. Liu et al. (2023) report that language models often fail to retrieve information placed in the middle of long contexts, and benchmarks such as LongBench reveal challenges in long-context understanding (Bai et al., 2024). These observations motivate our branch-loaded design: rather than directly inserting state cards, multi-view keyframes, and transition examples into the main agent context, the runtime first inspects selected evidence in a temporary branch and returns a compact structured guidance tuple. The branch is intended to reduce competition between reference views and live observations in the action-generation context.

Continual memory and procedural learning. Memento formulates adaptation through episodic memory, case selection, and memory rewriting without fine-tuning the underlying LLM (Zhou et al., 2025). Memento 2 studies a stateful reflective read/write process, treating writing as policy evaluation and retrieval as policy improvement (Wang, 2026). Memento-Skills applies a read/write loop to structured skills, with a router selecting skills and experience updating the library (Zhou et al.,

2026). Skill-Pro learns and maintains reusable procedural skills through non-parametric PPO, using semantic-gradient proposals and a verification gate (Mi et al., 2026). These works emphasize how memory is learned or updated; MMSkills emphasizes what multimodal procedural evidence is stored and how it is selectively read. Applying a proposal-and-gating loop to the $D/P/S/K$ fields is a potential integration, subject to the safeguards in Appendix I, rather than an evaluated part of the current method.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Section 1 state the representation, construction, and runtime contributions; Section 5 bounds the evidence to the evaluated domains and frozen-library setting.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Please refer to the conclusion and limitations section.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Please refer to the Experiment section and our public repository (`https://github.com/zkangning/MMSkills_for_Visual_Agents`).

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Please refer to the Experiment section and our public repository (`https://github.com/zkangning/MMSkills_for_Visual_Agents`).

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: Please refer to the Experiment section and our public repository (`https://github.com/zkangning/MMSkills_for_Visual_Agents`).

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: The main success-rate and game-score comparisons are point estimates; confidence intervals and significance tests are not reported. We interpret these differences descriptively rather than as established statistical significance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: Appendix C specifies the evaluation environment and budgets, and Appendix F reports OSWorld inference usage, latency, and API costs for six configurations. Full hardware specifications and total project compute, including construction and preliminary runs, are not reported.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses public non-evaluation trajectories and discusses privacy, misuse, and deployment risks in the Broader Impact appendix.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Please refer to the Broader Impact section in Appendix.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This work releases procedural skill packages rather than a new pretrained model. Potential automation misuse and screenshot-privacy risks, together with recommended deployment controls, are discussed in Appendix M.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [No]

    Justification: The source datasets, benchmarks, and models are credited, but the paper does not enumerate the license and terms of use of every upstream asset. Users of the released packages must also consult the original asset terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: Please refer to our public repository (`https://github.com/zkangning/MMSkills_for_Visual_Agents`).

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: Please refer to the Usage of LLMs section in the Appendix.

    Guidelines:

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