MMBench-GUI: A Unified Hierarchical Evaluation Framework for Multi-Platform GUI Agents - Supplementary Materials

Xuehui Wang[1,2*], Zhenyu Wu[1,2*], JingJing Xie[3,2*], Zichen Ding[2*], Bowen Yang[4,2*], Zehao Li[4,2*], Zhaoyang Liu[5,2*], Qingyun Li[6,2], Xuan Dong[7], Zhe Chen[8,2], Weiyun Wang[9,2], Xiangyu Zhao[1,2], Jixuan Chen[8,2], Haodong Duan[2], Tianbao Xie[10], Chenyu Yang[2], Shiqian Su[7], Yue Yu[7], Yanting Zhang[11], Xiangyu Yue[12], Weijie Su[2], Xizhou Zhu[7], Wei Shen[1✉], Jifeng Dai[7], Wenhai Wang[12,2✉]

[1]Shanghai Jiao Tong University, [2]Shanghai AI Laboratory, [3]Xiamen University,
[4]University of Science and Technology of China,
[5]The Hong Kong University of Science and Technology, [6]Harbin Institute of Technology,
[7]Tsinghua University, [8]Nanjing University, [9]Fudan University, [10]University of Hong Kong
[11]Donghua University, [12]The Chinese University of Hong Kong

In this section, we present material that expands the main content. We strongly encourage readers to consult this section in parallel with the main paper, as doing so will facilitate a more comprehensive understanding and assessment of our contributions under space-constrained presentation in the body of the paper. The appendix is organized as follows:

- Section 1: We review related works in GUI Agent and corresponding benchmarks.
- Section 2: We further conduct a comparative analysis against other relevant benchmarks.
- Section 3: We provide more L2 evaluation results on more recent methods.
- Section 4: We conduct a statistical analysis of the benchmark dataset.
- Section 5: We elaborate on the details about data annotation, including prompts, QC procedure.
- Section 6: We formally define all evaluation metrics and detail their computation, including step-by-step formulas and measurement protocols.
- Section 7: We present an in-depth analysis of the benchmark experiments and obtain some findings for the research community.
- Section 8: We provide a detailed description of the evaluation setup and protocol.

1. Related works

GUI Agents. GUI agents have attracted growing interest, driven by advances like Anthropic's Computer-Use Agent[1] and OpenAI's Operator[2]. Currently, GUI Agents mainly fall into two paradigms: a) Modular agent schemes [8, 13, 40, 42, 46, 50, 55], which typically employs general-purpose VLMs (*i.e.*, GPT-4o) as planners, integrated with a specially trained GUI grounding model for focused UI element localization; b) Native agent schemes [22, 35, 38, 43, 48, 50], where planning and grounding are trained in an end-to-end manner. Modular approaches benefit from state-of-the-art components but face challenges in system-level alignment [8, 13]. In contrast, the native agent paradigm aligns capabilities more naturally during training [35, 43, 48]. Both paradigms can use screenshots [23, 34], accessibility trees (A11y Trees) [12], and HTML pages [10, 11] as input. However, A11y Trees and HTML codes vary across platforms, are prone to noise, and may cause excessive token length [8, 15, 56]. Generally, in this work, we focus exclusively on the screenshot-only setting and propose a hierarchical, multi-platform benchmark to evaluate these vision-only native agents.

GUI Benchmarks. Effectively GUIs requires a sophisticated grasp of intertwined visual and textual cues, yet this complex domain remains largely outside the scope of general-purpose multimodal QA benchmarks [26, 31, 54]. While ScreenQA [16] and WebSRC [7] provide large-scale QA datasets based on Android screenshots and web pages respectively, and GUI-World introduces cross-platform GUI QA via video data, these efforts offer limited support for interactive GUI agents. To evaluate visual grounding in GUI contexts, several benchmarks have emerged. ScreenSpot [8]

[1]https://www.anthropic.com/news/3-5-models-and-computer-use

[2]https://openai.com/index/computer-using-agent

and its improved versions [19, 43] support cross-platform UI grounding with progressively enhanced realism and annotation quality. UI-I2E-Bench [25] and UI-Vision [32] further expand this by aligning natural language instructions with GUI elements of varying scale and type. VenusBench-GD [57] mainly targets GUI grounding and proposes a hierarchy within grounding subtasks. For reasoning and planning, offline benchmarks like [6, 9, 18, 20, 29, 36] assess action prediction from fixed trajectories, while online benchmarks [4, 24, 37, 44, 47, 58] enable interactive evaluation across platforms. Besides, MLA-Trust [49] focuses on trustworthiness (safety/privacy/controllability in high-risk scenarios), which is largely orthogonal to capability/efficiency benchmarking. However, macOS remains underexplored. Our MMBench-GUI benchmark addresses this gap by enabling online evaluation on macOS and emphasizing cross-platform robustness, providing a realistic and comprehensive evaluation for GUI agents.

2. Comparison with other benchmarks

Table 2 summarizes the design space of existing GUI-oriented benchmarks in a checklist-style comparison. Most prior work covers only a *subset* of the capabilities required for realistic GUI agents. GUI-World [5] and UI-Vision [33] focus on offline perception: they provide rich QA and grounding-style tasks over static or recorded interfaces but offer no interactive control layer. ScreenSpot [8] and ScreenSpot-Pro [19] specialize in element grounding on static screenshots, largely decoupled from any downstream automations. On the other end of the spectrum, OS-World [44], AndroidLab [47], and WindowsAgentArena [4] concentrate on interactive planning task, providing long-horizon computer-use tasks within specific environments, but they do not supply aligned perception or grounding tasks on the same scenes, and their designs are typically based on single application rather than multiple application.

In contrast, MMBench-GUI is the only benchmark that checks *all* boxes in the checklist. It combines offline GUI content QA (L1) and element grounding (L2) with single-application automation (L3) and explicit multi-application collaboration (L4) in a four-level hierarchy. All levels are built on shared GUI scenes and applications, which is crucial for analyzing how perception and grounding errors propagate to downstream control performance. Moreover, MMBench-GUI systematically spans six major platforms (Windows, macOS, Linux, iOS, Android, web), covering both desktop and mobile settings, whereas existing interactive environments are typically confined to a single OS family. Finally, beyond plain accuracy or success rate, MMBench-GUI introduces an efficiency-quality-aware metric (EQA) for L3/L4, while only AndroidLab includes any explicit efficiency notion among prior benchmarks. Together, these properties justify the need for MMBench-GUI as a unified, hierarchical, and efficiency-aware benchmark, rather than a simple aggregation of existing datasets.

3. Comparison with other benchmarks

In this section, we provide L2 grounding results for additional recent models [14, 21, 27, 28, 30, 39, 41, 45, 51–53], as summarized in Table 1. These findings are current as of the submission of this supplementary material. Further evaluations (e.g., across L1, L3, and L4) will be continuously updated in subsequent preprints or via our official code repository.

4. Benchmark details

4.1. Data source

The data sources of our benchmark are summarized in Table 3. These applications and websites span a wide range of common, real-world usage scenarios.

4.2. Benchmark Statistics

Table 4 enumerates the complete task inventory, 8123 distinct instances, broken down by operating platform, level, and difficulty band. Our benchmark has the following characteristics:

- L1-GUI Content Understanding (3 × QA splits). Each of the six platforms contributes an identical triplet of 271/84/196/115/307/221 items (Windows → Web), yielding 1194 examples per difficulty (Easy, Medium, Hard) and 3582 in total. This symmetry ensures that any performance gap across the three difficulty tiers cannot be attributed to data imbalance.
- L2-GUI Element Grounding (Basic vs. Advanced). The grounding set is roughly 50% larger than Level 1, with 1787 examples per split (Basic=Advanced). Note the deliberate platform skew: mobile platforms (iOS + Android = 686 or 38%) receive more queries than desktop platforms, reflecting the higher UI diversity and screen density of mobile apps.
- L3-GUI Task Automation (single application). A compact but varied set of 719 trajectories focuses on long-horizon planning within one application. Linux dominates (268 tasks) to capture the complexity of desktop productivity apps, while mobile splits are omitted for this level to avoid conflating OS diversity with task length.
- L4-GUI Task Collaboration (multiple applications). The hardest tier comprises 248 cross-application workflows. Although smaller, it intentionally spans all three desktop platforms and major mobile browsers (47 Web tasks, 30 Android tasks) to stress test memory hand-off and state persistence.
- Aggregate balance. Across the whole benchmark Windows (1536) and Android (1758) provide the two largest

Table 1. **Performance on the L2-GUI Element Grounding.** "Adv." stands for advanced, while "Avg." refers to the weighted average of all results in a row, where the weights correspond to the proportion of tasks for each platform and mode relative to the total number of tasks.

Model	Windows		MacOS		Linux		iOS		Android		Web		Avg
	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	
SE-GUI-3B [2025]	67.16	42.65	66.67	49.71	57.07	36.22	91.72	76.67	90.45	74.65	81.94	56.17	67.81
SE-GUI-7B [2025]	77.12	54.41	74.20	53.47	64.92	43.37	94.27	85.15	94.94	84.23	90.00	68.51	75.48
GUI-G2-7B [2025]	81.18	55.15	81.45	65.90	67.02	48.98	95.54	82.73	96.63	84.51	92.90	76.62	79.18
GUI-G2-3B [2025]	63.47	35.66	70.43	43.06	63.87	39.29	92.36	71.21	91.57	78.03	80.97	52.27	66.84
GUI-R1-3B [2025]	64.21	35.66	69.86	44.80	54.45	36.22	91.08	68.79	88.48	67.04	82.90	55.52	65.06
GUI-R1-7B [2025]	73.06	51.10	74.78	57.23	51.31	38.27	93.31	80.91	94.94	78.87	89.35	70.13	73.42
InfiGUI-G1-3B [2025]	72.69	44.49	78.84	54.62	64.40	43.37	94.59	80.00	91.57	78.03	87.74	62.66	72.85
InfiGUI-G1-7B [2025]	48.34	48.16	19.13	22.54	21.47	26.53	8.28	18.79	21.35	38.03	30.00	32.79	27.58
Jedi-3B-1080p [2025]	78.23	40.81	77.39	52.89	70.16	46.43	94.27	74.24	92.70	77.18	90.00	63.64	72.91
Jedi-7B-1080p [2025]	72.69	49.26	77.97	60.98	68.59	44.39	94.90	80.61	93.54	81.69	88.06	67.21	75.06
UI-AGILE-3B [2025]	68.27	45.59	73.04	51.16	56.02	38.27	90.13	73.03	88.76	76.90	83.55	61.69	69.11
UI-AGILE-7B [2025]	78.97	56.62	77.39	61.85	67.02	45.41	94.27	82.42	95.51	82.54	88.39	72.40	76.95
UI-Venus-Ground-7B [2025]	77.49	57.35	79.13	63.58	65.97	47.45	94.59	86.67	95.22	83.66	92.26	76.62	78.48
GTA1-7B [2025]	80.81	56.25	84.93	67.92	70.16	55.61	94.27	84.24	95.79	84.23	90.32	75.00	79.84
GTA1-32B [2025]	82.66	68.38	88.70	74.86	71.73	52.04	96.18	88.48	96.07	88.45	94.84	79.22	83.56
GTA1-72B [2025]	88.93	66.91	89.86	77.75	70.16	54.59	95.54	88.18	96.35	91.55	94.84	79.22	84.62
GUI-Owl-7B [2025]	83.76	60.29	79.71	61.85	72.77	59.18	95.22	81.82	95.22	81.69	92.26	74.03	79.26
GUI-Owl-32B [2025]	81.92	63.24	83.19	65.32	74.35	61.73	95.86	86.67	95.79	87.04	93.87	81.49	82.09
OpenCUA-7B [2025]	87.82	59.56	83.77	63.58	81.68	52.55	93.31	75.45	94.10	74.65	90.97	69.16	78.11
OpenCUA-32B [2025]	85.61	61.03	67.83	68.21	76.44	59.69	74.52	80.91	48.60	80.85	71.29	81.49	71.33
ScaleCUA-7B [2025]	78.60	54.04	82.32	58.67	74.35	56.63	94.27	81.82	96.07	81.13	92.58	73.05	78.18
ScaleCUA-32B [2025]	83.03	62.87	88.11	64.16	81.15	65.82	95.86	84.85	96.35	81.69	93.87	76.30	82.02

Table 2. **Comparison with other benchmarks.** "QA": GUI content QA, "Grounding": element grounding, "Single-app": single application planning tasks, "Multi-app": multiple applications planning tasks, "Hier.": hierarchical multi-level design, "Shared.": shared understanding/planning scenes, "Multi.": multi-platform coverage, "Desk+Mobile": support both desktop and mobile devices, "Eff.-aware": efficiency-aware metric.

Benchmark	QA	Grounding	Single-app	Multi-app	Hier.	Shared.	Multi.	Desk+Mobile	Eff.-aware
GUI-World [5]	✓	✗	✗	✗	✗	✗	✓	✓	✗
ScreenSpot [8]	✗	✓	✗	✗	✗	✗	✓	✓	✗
ScreenSpot-Pro [19]	✗	✓	✗	✗	✗	✗	✗	✗	✗
UI-Vision [33]	✓	✓	✗	✗	✗	✗	✗	✗	✗
OSWorld [44]	✗	✗	✓	✓	✗	✗	✗	✗	✗
AndroidLab [47]	✗	✗	✓	✓	✗	✗	✗	✓	✓
WindowsAgentArena [4]	✗	✗	✓	✓	✗	✗	✗	✗	✗
MMBench-GUI	✓	✓	✓	✓	✓	✓	✓	✓	✓

pools, but no single platform exceeds 22% of the corpus, guarding against model over-specialisation. The progressive shrinkage, from 3582 (L1) to 248 (L4), mirrors the increasing cost and difficulty of annotation, while still offering enough samples (about 250) for a statistically meaningful evaluation in the top tier.

Overall, the benchmark delivers (1) platform diversity, (2) controlled difficulty gradation, and (3) a realistic taper in task count that matches real-world annotation effort, thereby enabling fine-grained diagnosis of GUI agent capabilities.

4.3. More examples

In Figure 1, we present an additional set of examples that cover tasks from Levels L1 through L4. These examples illustrate evaluation data drawn from different platforms.

5. Annotation details

5.1. Prompt for generating L1 and L2 annotations

We provide the prompts used in constructing the L1/L2 evaluation data. Specifically, Prompts 1 and 2 are employed for

L1 - GUI Content Understanding

Question:

What action is available in the context menu that appears when right-clicking on a calendar in the My calendars list?

Options:

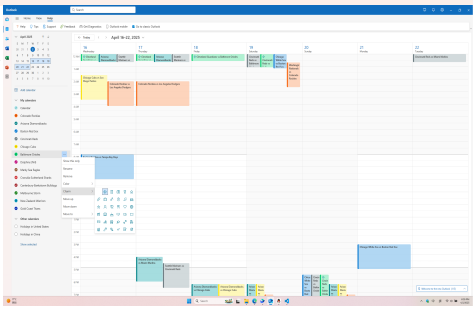
A.Show this only B.Print calendar C.Delete event D.Create new event E.Change time zone

Answer: A

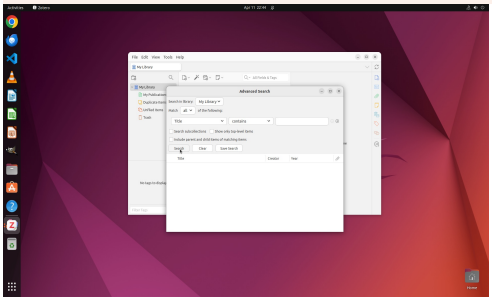
Explanation:

The context menu shows the 'Show this only' option, but does not include print, delete event, create new event, or change time zone.

Difficulty: Hard



L2 - GUI Element Grounding



Basic instruction:

An unchecked checkbox labeled 'Search subcollections' in the Advanced Search dialog box.

Advanced instruction :

Enable the option to include subcollections in your Zotero library search.

Data type: text

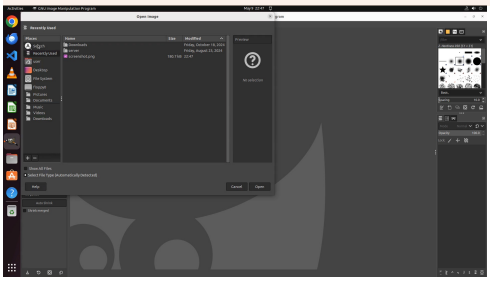
Platform: linux

App name: Zotero

Bbox: [0.3406, 0.4231, 0.4182, 0.4454]

Size: 1920x1080

L3 - GUI Task Automation



Task instruction :

Please adjust the brightness of the image that named as 'Panda' on the desktop as 40.

Evaluation function:

Check_Gimp_Status(content='image', items=['brightness', 40])

Platform : linux

Type: Single

App name: [Gimp]

Max steps: 50

L4 - GUI Task Collaboration

Task instruction :

Calculate how many years, months, weeks and days are between 10/08/1980 (MM/DD/YYYY) and 8/2/2024 using the calculator app, and save the result in a file called 'Differences.txt' on the Desktop (e.g. X years, Y months, Z weeks, W days)

Evaluation function :

Exact_Match(type='is_file_saved_desktop', filename='Differences.txt', textcontent='43 years, 9 months, 3 weeks, 4 days')

Platform : windows

Type : Multi

App name : [Calculator, Notepad]

Max steps : 50

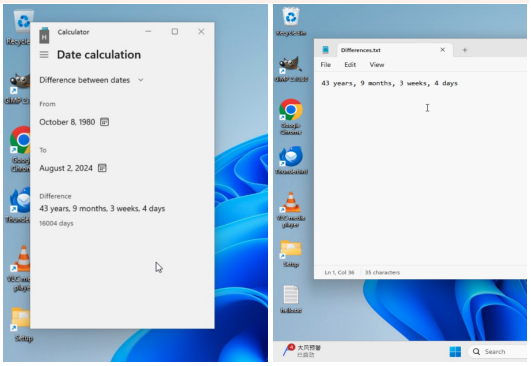


Figure 1. **More examples for all levels.**

the L1 task, for which detailed descriptions are given in the main text, while Prompt 3 is used for annotating the L2 task.

5.2. Quality Control

We adopt multiple quality control (QC) mechanisms across all levels of MMBench-GUI to ensure that evaluation results faithfully reflect GUI agents' capabilities rather than artifacts of noisy data or unstable environments. Below we summarize the key procedures for L1/L2 offline annotations and L3/L4 online tasks.

Table 3. **Data sources for all platforms.** We additionally sampled some examples from Screenspot_v2 to serve as supplementary data.

Platform	Data source
Windows	bilibili, calendar, disney+, excel, feishu, file_explorer, hulu, onenote, outlook, ppt, qq_music, settings, spotify, tencent_docs, telegram, word, youtube
Linux	chrome, gimp, libreoffice calc, libreoffice impress libreoffice writer, thunderbird, vscode, firefox, onenote, slack, zotero
MacOS	app store, apple music, calendar, disk utility safari, settings, shortcut, tencent video, tripsy, weather, xmind, zotero
Android	amazon, app market, bilibili, camera, douyin, health, hupu, kugou, qq music, qidian, reddit, settings, tencent map, trip, twitter, weibo, ximalaya, youtube, zhihu, douban
iOS	calendar, clock, fitness, github, health, home, podcasts, setting, twitter, youtube, outlook, system, spotify, screenspot_v2(ios)
Web	adidas.com.cn, adobe.com, amazon.com, apple.com, iqiyi.com, arxiv.org, cars.com, azure.microsoft.com, buaa.edu.cn, ebay.com, coursera.org, github.com, map.google.com, hupu.com, leetcode.cn, linkedin.com, modelscope.cn, music.163.com, newegg.com, screenspot_v2(forum), screenspot_v2(gitlab), screenspot_v2(shop), screenspot_v2(tool), rottentomatoes.com, us.trip.com, zhipin.com

5.2.1. QC for L1/L2 Annotations

L1: Question-answer quality. As described in the main paper, L1 Question-Options-Answer triples are generated by a multi-stage pipeline involving Claude 3.7, GPT-o4-mini, GPT-o3, and human review. To quantify annotation quality, we performed manual sampling over the final L1 pool:

- We randomly sample 900 L1 instances across platforms and difficulty tiers. For each instance, annotators verify (i) that the question is answerable from the screenshot, (ii) that the correct option is unique, and (iii) that the explanation (if present) is logically consistent.
- Among the sampled items, approximately 4.78% require minor edits (e.g., wording refinement or clarification of references), and 9.45% are discarded due to ambiguity or multiple plausible answers. The remaining **85.77%** are accepted without modification.
- The most frequent issues include: (i) ambiguous deictic expressions (e.g., "the button on the left" when multiple similar buttons exist), (ii) questions depending on information not visible in the screenshot, and (iii) options that are all technically correct but differ in granularity.

These statistics suggest that the LLM-based pipeline already filters most problematic items, and human review mainly serves to polish edge cases and remove a small fraction of ambiguous questions.

L2: Grounding instructions and element labels. For L2, we control quality at two levels:

- *Element labels.* Interactive elements (buttons, icons, input fields, menus) are manually annotated with bounding boxes and categorized as Text or Icon. Annotators double-check overlaps and resolve cases where two elements have highly overlapping regions. In a random subset of 536 screenshots, the agreement on whether an element is correctly localized (within a small tolerance) is above 98.7%.
- *Instructions.* For each annotated element, Claude 3.7 proposes Basic and Advanced instructions, which are then reviewed in our internal annotation tool. Annotators ensure that (i) each instruction uniquely maps to a single element, and (ii) the target element can be identified without external knowledge beyond the screenshot.
- During this process, approximately 35% of candidate instructions are edited (e.g., to resolve ambiguity or remove redundant cues), and 4.3% are discarded and replaced. Typical issues include over-specific references to transient content, underspecified location descriptions, and instructions that could apply to multiple visually similar elements.

These QC steps ensure that L2 instructions provide a clear and realistic grounding target while preserving diversity in descriptive style and difficulty.

5.2.2. QC for L3/L4 Online Tasks

Task stability and environment robustness. L3 and L4 operate in virtualised GUI environments sourced from multiple public benchmarks (OSWorld, AndroidWorld, WebArenaLite-v2, WindowsAgentArena) and our MacOSArena). To minimize evaluation noise from environment instabilities, we apply the following filtering:

- For each platform, we run all candidate tasks with a reference agent or scripted policy to verify that the environment loads correctly, required applications are available, and network-dependent content is accessible.
- Tasks that frequently fail due to external factors (e.g., unstable websites, mandatory account logins, time-dependent content, platform-specific crashes) are removed. In total, we discard/fix approximately 18/13 out of 967 candidate tasks, respectively, ensuring that

Table 4. **Statistics of the evaluation data in MMBench-GUI.** Owing to the inherent restrictions of the iOS ecosystem, we were unable to include online tasks for iOS in L3&L4. All other platforms are covered in full.

	Windows	MacOS	Linux	iOS	Android	Web	Overall
L1				L1 - Easy			
	271	84	196	115	307	221	1194
				L1 - Medium			
	271	84	196	115	307	221	1194
				L1 - Hard			
	271	84	196	115	307	221	1194
L2				L2 - Basic			
	271	345	191	314	356	310	1787
				L2 - Advanced			
	272	346	196	330	335	308	1787
L3	145	35	268	-	116	155	719
L4	35	35	101	-	30	47	248
Total	**1536**	**1013**	**1344**	**989**	**1758**	**1483**	**8123**

remaining failures during evaluation can be attributed to agent capacity rather than environment anomalies.
- For MacOSArena, all tasks are manually validated at least 3 times to confirm that the success condition is well-defined and reliably checkable from the final state.

Success checks and reproducibility. All L3/L4 tasks are equipped with programmatic success checks that inspect the final environment state (e.g., file system, application UI, form content) to decide whether a task has been completed. The same checks are applied across all agents, and we fix the maximum step budgets and random seeds for environment initialization to improve reproducibility. For multi-application L4 tasks, we additionally test that variations in intermediate action sequences (e.g., different but equivalent navigation paths) do not affect the final success decision.

Overall, these QC measures for L1-L4 jointly ensure that MMBench-GUI provides reliable and interpretable signals about the ability of GUI agents.

6. Details of Evaluation Metrics

6.1. Evaluation Metrics for L1-GUI Conetnt Understanding.

Formally, the accuracy for an evaluation set comprising N Question-Options-Answer pairs can be defined as:

$$\text{Acc} = \frac{1}{N}\sum_{i=1}^{N}\Theta(o_i^* = o_i), \tag{1}$$

where $\Theta(o_i^* = o_i)$ is an indicator function that equals 1 if the predicted answer o_i^* for the i-th pair matches the ground-truth answer o_i and 0 otherwise.

To account for variations in the number of answer choices, we introduce a simple dynamic adjustment factor α to rescale the original accuracy of each question. Taking Windows platform which has N_{win} questions as an example, the accuracy of L1 is computed as:

$$\text{Acc}_{win} = \frac{1}{N_{win}}\sum_{i=1}^{N_{win}}\alpha\cdot\Theta(o_i^* = o_i), \quad \alpha = \frac{m_i - 1}{m_i} \tag{2}$$

where m_i is the number of options for question i. Accordingly, for any given difficulty level, the agent's understanding ability (i.e., accuracy) can be computed as:

$$\text{Score} = \sum_{j\in\mathcal{O}}\frac{N_j}{N}\cdot\text{Acc}_j \tag{3}$$

where $\mathcal{O} = \{\texttt{win}, \texttt{linux}, \texttt{mac}, \texttt{ios}, \texttt{android}, \texttt{web}\}$ denotes the set of operation platforms, N_j is the number of questions for platform j, $N = \sum_{j\in\mathcal{O}} N_j$ is the total number of questions across all platforms.

6.2. Evaluation Metrics for L3 and L4

We define EQA as a continuous-time recall metric over cumulative agent effort. Consider an ordered set of N tasks. For each task $i \in \{1, 2, \ldots, N\}$, let:
- $s_i = 1$ if the agent successfully completes task i, and $s_i = 0$ otherwise,
- $t_i > 0$ be the number of steps the agent takes to complete task i.

Prompt 1:

You are an expert GUI analyst for {os_name} and item-writer.

Input:

1. One screenshot of a GUI application.
2. The application's name ({app_name} or "Not available") — optional and for background only; do **not** mention it in any question text.

Task:

Create *exactly one* multiple-choice question about the screenshot at **each** of three difficulty levels (easy, medium, hard). For **every** question you generate:

- Write the stem in clear English that can be answered *only* by understanding the screenshot. Avoid trivial facts (e.g., "What color is the button?") unless color is functionally meaningful.
- Focus on tasks, labels, hierarchy, states, or affordances shown in the UI.
- Provide **4–6** answer options labeled "A", "B", "C", ... in a JSON sub-object called `"options"`.
- Ensure **one and only one** option is strictly correct; the others must be clearly incorrect but plausible. Give the answer key (the letter of the correct option).
- Double-check yourself that the correct answer is indeed unique and unambiguous.
- Do **not** include the {app_name} or any other identifying text of the app in the stem or options.
- Give a concise `"explanation"` stating *why* the correct option is right and the others are not in 1–3 sentences.
- The hard question should require the answerer to think more about the screenshot, the question, and the options (you can also make options be easy to confuse).

Output format:

Return a single valid JSON array containing three objects (one per difficulty), in *English*, structured exactly like this schema:

```
[
  {
    "difficulty": "easy",
    "question": "<stem>",
    "options": {
      "A": "<option text>",
      "B": "<option text>",
      "C": "<option text>",
      "D": "<option text>"   // add "E","F" only if needed
    },
    "answer": "A",
    "explanation": "<brief rationale>"
  },
  ...
]
```

Important Constraints:

1. Produce only the JSON text—no markdown, headings, or commentary.
2. Validate that the JSON is syntactically correct before outputting.
3. After generation, internally review each Q&A for accuracy and compliance.

We define the cumulative cost and cumulative success after the first k tasks as:

$$T_k = \sum_{j=1}^{k} t_j, \qquad S_k = \sum_{j=1}^{k} s_j. \tag{4}$$

Let the global budget be $T_{\text{max}} = N \cdot t_{\text{max}}$, where t_{max} is the maximum step limit per task. We normalize the cumulative effort as:

$$u_k = \frac{T_k}{T_{\text{max}}} \in [0, 1]. \tag{5}$$

The instantaneous recall at normalized time u is defined as:

$$R(u) = \max_{k:\, u_k \le u} \frac{S_k}{N}, \qquad u \in [0, 1]. \tag{6}$$

Finally, EQA is computed as the area under the step-wise

Prompt 2:

You are a meticulous GUI-QA evaluator.

Input:

1. One screenshot (`image`) of a GUI application running on a {os_name}.
2. The application's name (`app_name`) – optional and strictly for background; *never* mention it in your output.
3. A JSON-like array (`qa_items`) containing three single-choice questions about the screenshot (intended levels: `easy`, `medium`, `hard`). Each object is expected to have the keys question, options, answer, difficulty, and optionally explanation.

* Ignore cosmetic or syntactic issues in the supplied JSON (e.g., extra backticks, missing quotes, inconsistent key order, markdown fences).
* Focus **only** on the content of `question`, `options`, and `answer` when deciding validity.

Task:

For each question, decide whether it is content-valid for use in a test. A question is valid only if all the following hold:

- The stem can be answered solely by inspecting the screenshot (no outside knowledge).
- Exactly one option is correct and that option is the one listed in `answer`.
- Incorrect options are clearly wrong yet still plausible.
- Neither stem nor options reveal the `app_name`.
- The difficulty label is reasonable (honor system; do not reject only for minor mislabelling).

The hard level should allow the answerer to think more deeply about the screenshot, the question, and the options. You may make the options easy to confuse.

* Do **not penalise** minor formatting faults that do not affect the five substantive criteria.

Output format:

Return a JSON array of three objects in the original order, each with:

```
{
  "difficulty": "<same as input>",
  "valid": "yes" | "no",
  "comment": "<if valid: empty string; if not valid: brief reason why>",
  "fix": <if valid: null; if not valid: a *fully corrected* object that replaces the
      faulty one (same schema as above, with all issues fixed)>
}
```

Notes:

1. Provide an empty string (" ") for `comment` and "null" for `fix` when `valid` is "yes".
2. When `valid` is "no", supply both an actionable `comment` and a complete `fix` object that meets all criteria.
3. Do not wrap the result in markdown or add explanations outside the JSON.
4. Verify that the final JSON is syntactically correct before sending it.

non-decreasing recall curve:

$$\mathrm{EQA} = \int_0^1 R(u)\,du \;\approx\; \frac{1}{M}\sum_{m=0}^{M-1} R\left(\frac{m}{M-1}\right), \qquad (7)$$

where $M = 101$ denotes the number of uniformly spaced evaluation points. This metric encourages agents to complete more tasks in fewer steps, offering a holistic measure of task performance.

6.3. Sensitivity to the selection of M

EQA is defined as a Riemann-sum approximation to the continuous-time success-vs-step-budget curve, where M controls the number of discrete budgets $\{B_m\}_{m=1}^{M}$ used for numerical integration. A natural question is how sensitive the metric is to the choice of M, and in particular whether our default setting $M = 101$ materially affects the conclusions.

To study this, we vary M over a range of values:

$$M \in \{M_{21}, M_{51}, M_{101}, M_{201}\} \qquad (8)$$

and recompute EQA for all agents on the full L3 benchmark. [3] Figure 4 reports the resulting scores and ranking correlations.

[3] We keep the minimum and maximum step budgets fixed and only change the number of intermediate evaluation points. Concretely, we linearly space the budgets between 1 and $T_{\max}$ for each level.

Prompt 3:

You are a GUI agent currently operating on a {os_name}.

Input:

1. The first image is a screenshot from the {application} {app_or_web}, in which a selected element is highlighted with a distinctive red box and a red arrow.
2. The second image is the cropped region containing the selected element and corresponding box and arrow.
3. A simple and coarse description of the selected element.

Task:

Your task is to understand the possible role, function, and related global contextual information of the selected element on the current page from the first image. Then, from the second image, you can combine the global information from the first image to further analyze the relationship between the selected element and its surrounding information. The simple and coarse description can be regarded as a prior for the selected element. Finally, you are required to conclude two types of instructions for the selected element:

* *Basic Instruction*: Informative description that summarizes key information.
* *Advanced Instruction*: An indirect yet specific instruction that refers to the selected element.

Guidelines for Generating Descriptions:

Basic Instruction:

- Concise summary including appearance and position.
- Avoid referencing the red box or arrow.
- Examples:
  - "A circular icon with a white background and a magnifying glass symbol in black."
  - "Located in the top-right corner, to the right of the profile avatar icon."

Advanced Instruction:

- Focus on function and reasoning.
- Avoid visual/positional terms.
- Examples:
  - "Search some latest posts"
  - "Type in text to discover related content"

Output format:

Return a dictionary with:

```
{
   "basic_instruction": ["xxxx", "xxx", "xxx"],
   "advanced_instruction": ["xxxxx", "xxx", "xxx"]
}
```

Notes:

1. Ensure instructions are clear, unambiguous, and concise.
2. Do not mention the red box and arrow.
3. Coarse descriptions are only priors.

Table 4. **Ablation study on the selection of M.**

Model	M_{21}	M_{51}	M_{101}	M_{201}
GPT-4o + UI-TARS-1.5-7B	18.1	18.5	**18.7**	18.7

Empirically, we observe that once M is moderately large (e.g., $M \geq M_{51}$), EQA becomes very stable:

- **Scores change smoothly.** Increasing M from M_{51} to M_{101} and further to M_{201} leads to only minor changes in absolute EQA values, consistent with the intuition that we are refining the Riemann-sum approximation of a smooth curve.
- **Rankings are robust.** The ranking is identical for the top-K agents and only differs slightly among low-performing agents. In particular, the key qualitative conclusions reported in the main paper (e.g., which agents

are more efficient at similar success rates) remain unchanged.

- **Small M behaves as a coarse approximation.** When M is very small (e.g., $M = M_{21}$), EQA becomes a coarse approximation that may slightly over- or under-emphasise performance at specific budgets, leading to modest ranking fluctuations among agents with very similar trajectories. This is expected as the integral is undersampled in this regime.

Based on these observations, we choose $M = 101$ as a default that offers a good trade-off between numerical smoothness and computational overhead. Computationally, computing EQA is inexpensive compared to running the agents: success at all budgets $\{B_m\}$ can be obtained from a single trajectory per task by truncating at different step limits, without additional environment rollouts. Overall, our ablation confirms that EQA is *not* overly sensitive to the exact choice of M, and that our main conclusions are robust under reasonable variations of this hyperparameter.

6.4. Comparison with existing efficiency metrics.

Several recent benchmarks have begun to consider efficiency-related aspects for GUI or computer-use agents, but in more limited and environment-specific ways than our EQA metric. For example, AndroidLab [47] reports indicators such as Reversed Redundancy and Reasonable operation (RRR) to penalize obviously unnecessary actions, and some environments also log average step counts or impose step limits. These quantities are useful diagnostic statistics, but they are largely heuristic and tied to a particular task design or platform, rather than formulated as a unified metric that can be applied consistently across single- and multi-application tasks or across different operating systems. For example, RRR measures redundancy relative to a human reference trajectory (path-length ratio), which is useful when reliable trajectories are available, but it is not designed to jointly reflect success and efficiency in one score and requires costly human baselines that are hard to standardize across heterogeneous multi-OS/app tasks.

Our EQA metric is conceptually related to the family of "success $\times$ efficiency" measures, such as the Weighted Efficiency Score (WES) proposed in OSWorld-Human [1]. OSWorld-Human augments OSWorld with human-annotated minimal trajectories and defines WES scores that weigh task success by the ratio between human and agent step counts, thereby quantifying how many more steps an agent takes compared to an efficient human. This design is well suited for analyzing human-likeness and over-thinking on OSWorld, but it relies on carefully collected human reference trajectories and is tied to a single environment.

EQA follows the same high-level philosophy of jointly evaluating quality and cost but makes a different trade-off that is more appropriate for a heterogeneous, multi-platform benchmark like MMBench-GUI. First, EQA does *not* require human gold trajectories: it is defined purely from success-vs-step-budget curves, and can therefore be applied uniformly to all L3/L4 task sets (single- and multi-application, across Windows, macOS, Linux, iOS, Android, and web) without additional annotation effort. Second, instead of comparing to a single "optimal" length, EQA aggregates success across a range of step budgets and normalizes the resulting area to $[0, 1]$, capturing how quickly an agent's success saturates as more steps are allowed. In our experiments, this yields different rankings for agents that have similar success rates but very different interaction costs, revealing efficiency gaps that success-only metrics or ad-hoc step statistics would obscure. In this sense, EQA complements existing efficiency indicators (AndroidLab's heuristic measures and OSWorld-Human's WES) by providing a principled, platform-agnostic, and task-agnostic efficiency metric for hierarchical GUI benchmarks.

7. Further Analysis and Key Findings

In this section, we conduct an in-depth analysis to delve into the underlying causes and implications reflected in our benchmark results. Our investigation is structured around three primary dimensions: platform, task, and model, and adheres to a single-variable control principle to ensure the validity of our comparisons. By processing the empirical results, we identify the essential challenges facing contemporary GUI agents, offering valuable guidance to advance future development.

Finding 1: General-purpose language models excel at task decomposition, planning, and self-reflection but struggle with fine-grained visual interactions. Across different model categories, proprietary models, exemplified by GPT-4o and Claude, demonstrate pronounced limitations in fine-grained GUI tasks. As shown in Table 2 of the main paper and the right part of Figure 2, their average scores in L2 are merely 2.87 for GPT-4o and 4.66 for Claude-3.7, in contrast to the specialized visual grounding model UGround-V1-7B, which achieves a score of 65.68%. A similar trend emerges in L3 tasks. For instance, GPT-4o alone achieves success rates (SR) of only 4.05%/6.13% in single-app automation scenarios (Max Step = 15/50, see Table 3 of the main paper). However, when paired with domain-specific grounding modules such as UGround-V1-7B or UI-TARS-1.5-7B, the SR of GPT-4o rises substantially to 11.93%/17.50%. These phenomena indicate that proprietary models are inherently deficient in the precise perception and localization of UI components, which can be effectively remedied by specialized perception modules.

Thus, beyond incorporating auxiliary localization modules during training and increasing the amount of fine-grained perceptual data, a more fundamental and forward-looking direction lies in embracing a modular architecture.

This approach enables the model to dynamically interface with external modules based on its own capability gaps (e.g., visual grounding), effectively allowing for targeted augmentation through specialized "external agents". Within this synergistic and collaborative framework, the capabilities of general-purpose models can be significantly augmented and tailored for complex GUI automation tasks.

Finding 2: Accurate visual grounding significantly determines the success rate of GUI task execution. The full decision-making pipeline of a GUI agent can be abstracted into three stages: perceive accurately ⇒ reason properly ⇒ act precisely. An initial failure in element localization will cause cascading errors, rendering subsequent steps ineffective. To examine the critical role of localization, we designed two complementary experimental setups as shown in the left part of Figure 2: (1) fixing the planner while incrementally improving the grounder, and (2) fixing the grounder while varying the planner. Correlation analyses revealed a clear pattern: with the same planner, improving localization alone led to a 2.8× ($\Delta = 17.25$) increase in SR. In contrast, when localization performance remained roughly constant, replacing the planner with a stronger VLM yielded marginal returns (1.15×, $\Delta = 3.58$). This finding indicates that the key to improving GUI task automation lies in advancing visual localization. Consequently, the visual grounder should be the primary and most critical component in any modular architecture, providing the stable foundation required for higher-level functions like planning, memory, and reflection.

Finding 3: Efficiency, including step minimization and early stopping, is a critical yet underexplored dimension of GUI agent performance.

The introduction of the EQA metric enables us to move beyond evaluating whether an agent simply completes a task, by shifting attention to how efficiently the task is accomplished. This novel perspective facilitates deeper insights through a more fine-grained analysis of agent behavior.

We additionally compute two derived metrics, $\frac{EQA}{SR}$ and $SR - EQA$, to facilitate a more comprehensive analysis. Based on the definition of the EQA and SR, we further reformulate them as:

$$EQA = \frac{1}{N}\sum_{i\in\mathcal{C}}(1-u_i), \quad SR = \frac{|\mathcal{C}|}{N}, \tag{9}$$

where $\mathcal{C}$ denotes the set of all successfully completed tasks, and $u_i = \frac{T_i}{T_{max}} \in (0, 1]$ represents the normalized completion step of task i within the global step budget. From this, $\frac{EQA}{SR}$ and $SR - EQA$ can be derived as:

$$\frac{EQA}{SR} = \frac{1}{|\mathcal{C}|}\sum_{i\in\mathcal{C}}(1-u_i) = 1 - \frac{1}{|\mathcal{C}|}\sum_{i\in\mathcal{C}}u_i, \tag{10}$$

$$SR - EQA = \frac{1}{N}\sum_{i\in\mathcal{C}}u_i = SR \times (\frac{1}{|\mathcal{C}|}\sum_{i\in\mathcal{C}}u_i), \tag{11}$$

where $\frac{1}{|\mathcal{C}|}\sum_{i\in\mathcal{C}}u_i$ denotes the average steps in which a task is completed.

Therefore, $\frac{EQA}{SR}$ has an intuitive physical interpretation: it reflects the average remaining steps per successful task. Its upper bound is 1, which corresponds to the idealized case where all successful tasks are completed almost immediately (i.e., at the first step). Conversely, its lower bound is 0, indicating that all successful completions occur only at the very end of the allowed budget. $\frac{EQA}{SR}$ quantifies how many steps, on average, are consumed before successful completion. Meanwhile, $SR - EQA$ also has an intuitive physical interpretation: it is approximately proportional to the total normalized time consumed across all successful tasks, and can be interpreted as a "redundant step bill". A larger difference between EQA and SR implies a greater average normalized completion time u_i for the successful set, meaning that tasks tend to be completed closer to the end of the budget—i.e., with more redundant steps. Conversely, a smaller difference (approaching zero) indicates that most successful tasks are completed early, near the beginning of the budget, suggesting minimal or no redundancy. Thus, the magnitude of the gap between EQA and SR effectively captures how "wasteful" the agent is, even among the tasks it completes.

We re-organize the $\frac{EQA}{SR}$ and $SR - EQA$ using the average results in Table 3 of the main paper as EQ^1 and EQ^2, and present the aggregated findings in Table 5. Combining with Figure 3, we can disclose four complementary patterns. First, the modular pairing of a powerful planner with a specialized grounder, exemplified by GPT-4o + UGround-V1-7B and GPT-4o + UI-TARS-1.5-7B, elevates the success rate under a 50-step budget by roughly 5.7%, yet still incurs a substantial redundant step cost (EQ^2 = 7-8), signaling that cross-module coordination and early termination heuristics remain inadequate. Second, the large-scale DPO-aligned UI-TARS-72B-DPO achieves the strongest efficiency profile, increasing EQ^1 to 0.773 while compressing EQ^2 from 8.96 to 5.75 (ΔEQ^2 = -3.21); this demonstrates that aligning to human preferences that explicitly reward rapid task completion can translate directly into tangible efficiency gains. Third, general-purpose agents such as GPT-4o and Claude-3.7 extract minimal benefit from a longer budget (ΔSR<2.5%) and even exhibit higher redundant step costs (EQ^2 increases from 3.09 to 3.22 and 3.40 to 3.65, respectively), underscoring that simply extending the interaction horizon cannot compensate for their limited visual granularity and action precision, therefore, integrating specialized perception or actuation modules is becoming indispensable. Lastly, none of the curves in Figure 3 attains the ideal "hug-the-top-left-corner" profile, underscoring a pervasive lack of effective early-stopping heuristics and cost-aware search strategies.

To mitigate the efficiency bottlenecks aforementioned,

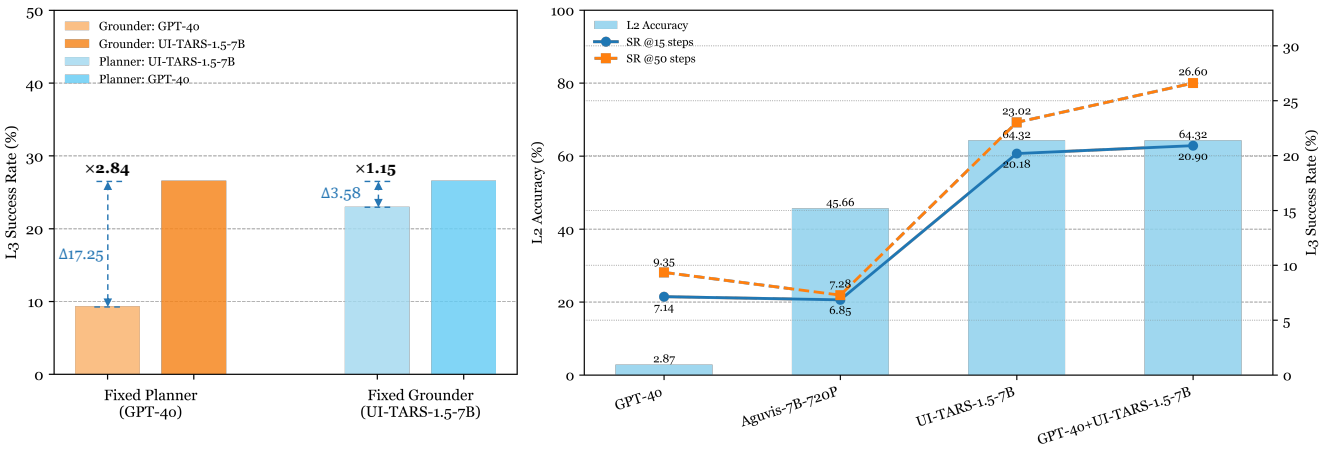


Figure 2. **Left:** Demonstrates the relative contribution of visual grounding versus planning in driving performance gains under current conditions. We consider two experimental conditions—fixing the planner while varying the grounder, and vice versa—and examine how different combinations affect task success rate. Similar color hues denote groups with the same fixed planner or grounder. **Right:** Task success grows roughly linearly with visual-grounding accuracy. General-purpose language models are virtually "blind" at the L2 grounding stage, which drives their L3 automation success rate (SR) sharply down. Plugging in a dedicated visual grounder restores precise perception and, in turn, lifts SR dramatically—highlighting fine-grained grounding as the principal bottleneck.

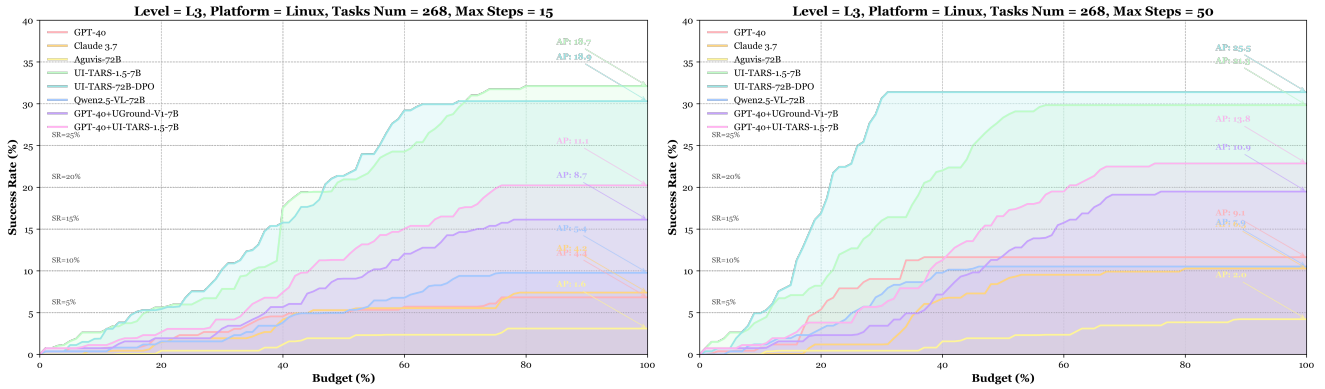


Figure 3. **EQA visualization across different models under L3 for different allowed steps.** As discussed before, EQA reflects a combination of task completion and efficiency (i.e., the number of steps used upon completion). In practice, we compute it by interpolating both the step budget and the success rate (SR) 100 times. The area under the curve formed by these interpolated SR values yields the final EQA score.

we identify three possible research avenues. (1) Confidence- or value-based early-termination policies: equip agents with stopping rules that immediately end an episode when the marginal utility of further actions falls below a threshold, rather than passively consuming the entire step budget. (2) Cost-sensitive fine-tuning: during reinforcement-learning (or DPO-style) alignment, impose explicit penalties for every superfluous action so that optimization shifts from maximizing success rate (SR) alone to jointly maximizing the success-conditioned efficiency score EQA. (3) Progress-aware self-reflection: require the planner to periodically estimate the set of remaining sub-goals and, upon detecting that all objectives are satisfied, issue an immediate FINISH action. Together, these interventions target the twin goals of cutting redundant steps and encouraging agents to "know when to stop", thereby narrowing the gap between current GUI agents and human-level operational efficiency.

Finding 4: The limitation of action space restricts the agent's ability to execute planned actions, especially in GUI task collaboration scenarios.

In Table 3 of the main paper, a notable fraction of models fail to complete the task on the web platform. The underlying cause is that, during web-interaction execution, the models lack the ability to trigger action `switch_tab` to enable 'press Tab to switch tabs'. In headless-browser settings, this omission blocks seamless navigation across multiple tabs,

Table 5. Additional metrics derived by SR and EQA. Here, EQ^1_{15} and EQ^2_{15} denotes for $\frac{EQA}{SR}$ and SR − EQA, respectively, when the maximal step is 15. $\Delta EQ^1 = EQ^1_{50} - EQ^1_{15}$ and so is the ΔEQ^2. Similarly, $\Delta SR = SR_{50} - SR_{15}$ and $\Delta EQA = EQA_{50} - EQA_{15}$

Model	ΔSR	ΔEQA	EQ^1_{15}	EQ^1_{50}	EQ^2_{15}	EQ^2_{50}	ΔEQ^1	ΔEQ^2
GPT-4o [2024]	2.21	2.08	0.57	0.66	3.09	3.22	0.09	0.13
Claude-3.7 [2025]	1.20	0.95	0.50	0.55	3.40	3.65	0.04	0.25
Aguvis-72B [2024]	0.43	0.56	0.52	0.57	3.29	3.16	0.05	-0.13
UI-TARS-1.5-7B [2025]	2.84	3.23	0.64	0.70	7.31	6.92	0.06	-0.39
UI-TARS-72B-DPO [2025]	2.06	5.27	0.62	0.77	8.96	5.75	0.16	-3.21
Qwen2.5-VL-72B [2025]	1.57	2.86	0.60	0.74	4.91	3.62	0.14	-1.29
GPT-4o+UGround-V1-7B [2024]	5.71	5.57	0.62	0.70	7.43	7.57	0.08	0.14
GPT-4o+UI-TARS-1.5-7B [2025]	5.70	7.53	0.53	0.70	9.74	7.91	0.17	-1.83
Avg.	2.72	3.51	0.57	0.67	6.02	5.23	0.10	-0.79

preventing cross-window information from being transferred from one context to another and ultimately derailing task completion.

On the other hand, due to the inherent heterogeneity of interactions across desktop, mobile, and web platforms, the current prompt-based definition of action functions struggles to comprehensively capture the full spectrum of platform-specific operations. Moreover, during inference, models may confuse actions across platforms, producing incorrect or incompatible output actions. Such issues can directly lead to task failure, even in single-platform, multi-app scenarios, and become particularly pronounced in multi-platform, multi-app settings, for example, when copying text from a web page and pasting it into a desktop application like Word for further formatting.

Building on these observations, we argue that a more generalizable, extensible, and potentially platform-agnostic definition of the action space is worth pursuing. One intuitive and straightforward direction is to construct a unified API abstraction layer that comprehensively covers multi-platform operations. Under this design, the agent interacts with the environment by invoking platform-independent APIs, while the backend of the API is responsible for platform-specific adaptations. An alternative route focuses on operation atomization. Unlike current action spaces that rely on fixed, platform-tied commands, an ideal action space would emphasize a set of primitive operations, decoupled from any particular environment. Agent-issued instructions are then mapped to these primitives via a many-to-many translation schema, where each high-level intent may correspond to a combination of atomic steps. These atomic units can then be recompiled into platform-specific execution commands, enabling robust and consistent interaction across environments. Beyond these two approaches, we believe that the research community should continue to explore better formulations of the action space, those characterized by strong generality, high extensibility, and minimal platform dependence.

Finding 5: Although many GUI agents excel in simple cases, their effectiveness diminishes significantly as task complexity rises, revealing limited generalization capabilities.

As shown in Figure 4, although many systems perform impressively on easy scenarios, their accuracy/success rate deteriorates sharply as soon as either (i) the local difficulty within a level increases (easy → medium → hard; basic → advanced) or (ii) the global task complexity rises from L1 to L4. These steep drops - especially pronounced for general-purpose LLMs - indicate that today's agents still lack robust generalization to harder, less stereotyped GUI situations. For example, the GUI understanding score of GPT-4o drops from 60.2% (easy) to 53.5% (hard), a -11% decrease, while even the highly tuned InternVL3-72B loses 4% (Table 1 of the main paper). In element grounding, switching from 'Basic' to semantically implicit 'Advanced' queries slashes GPT-4o's mean accuracy by nearly 40% and still costs the specialist UI-TARS-72B-DPO 16% (Table 2 of the main paper). The effect compounds across levels: the strongest agent (GPT-4o + UI-TARS-1.5-7B) succeeds in 26.6% of tasks at L3 but only 8.8% once multi-app collaboration is required in L4, a 67% collapse that is mirrored by other models (Tables 3 in the main paper). Concomitant declines in EQA confirm that agents not only fail more often but also waste proportionally more steps before failing.

These sharp drops expose three intertwined bottlenecks: (1) ill-posed perceptual clues (small widgets, non-salient text), (2) longer credit-assignment chains, and (3) noisy action spaces inflate the search space exponentially. Current models, trained largely on static screenshots, lack the robust abstract representations and error-driven exploration strategies needed to cope.

Possible targeted remedies include: (1) Curriculum & hard-negative mining. Intentionally up-sample adversarial layouts (occlusion, theme changes, deceptive affordances) during instruction tuning to inoculate perceptual modules against distribution shift. (2) Dynamic skill routing. Teach planners to self-diagnose uncertainty and automatically invoke auxiliary skills (OCR, vision transformers, memory retrieval) as difficulty rises. (3) Hierarchical planners with macro-actions. Introduce option-level abstractions (e.g., `open-browser-tab`) so that sparse EQA-style rewards can flow to high-level decisions instead of individual clicks. (4) Unified state schema for all applications. Store "App →

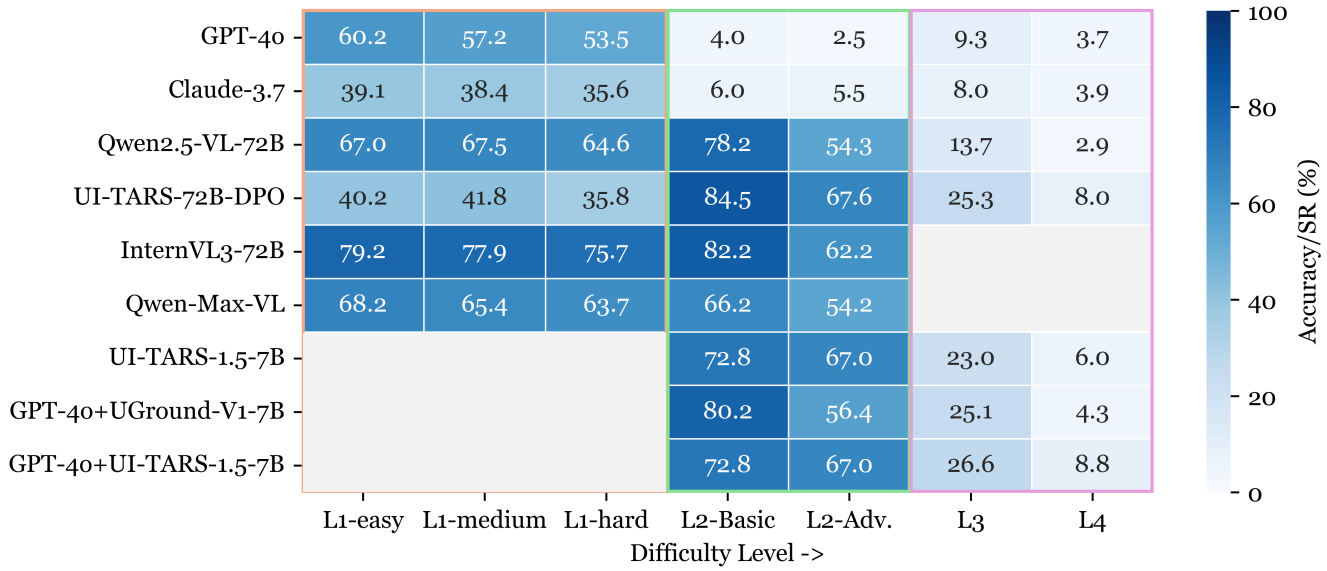


Figure 4. **Difficulty-Gradient Heatmap.** Models' scores across difficulty levels are encoded with a single-hue palette whose saturation fades from high (dark) to low (light). Colored rectangles outline comparable model groups. Within and across these groups, the color consistently fades from L1, L2 to L3 and L4, indicating that higher task complexity amplifies each model's weaknesses and causes a steep performance drop-off.

Page → Element" graphs in an external memory that survives context switches, allowing the planner to reason over shared entities rather than raw pixel buffers.

We believe that by attacking these verified failure modes, the community can turn today's hardest cases, from implicitly described buttons to multi-window workflows, into stepping-stones toward truly general-purpose GUI agents.

Finding 6: The failures in multi-application environments primarily stem from limited cross-context memory and action space, rather than issues with perception or planning.

Success drops that cannot be explained by harder screenshots or longer action chains alone appear as soon as the agent must pass information between applications. The strongest single-app system, GPT-4o+UI-TARS-1.5-7B, falls from **26.6%** SR on L3 to just **8.8%** on L4 (Tables 3 in the main paper); UI-TARS-72B-DPO shows an almost identical collapse (25.3% to 8.0%). Failures concentrate at window or tab boundaries: five models are labeled '-' on the Web platform simply because they cannot express the primitive `switch_tab`. At the same time, EQA shrinks far more than the accompanying $SR - EQA$ penalty (e.g., 18.7% → 6.4% for GPT-4o + UI-TARS), signaling that agents waste many steps rediscovering the context they have just lost. These phenomena point to a deficit in working memory and action-space coverage, rather than in perception or generic planning.

Addressing these failures may require agents to focus on memory-centric research avenues, including: (1) External episodic buffer. Log every UI observation and write-back (*copy*, *navigate*, *paste* ...) to an append-only timeline that the language planner can query with natural language—much like retrieval-augmented generation, but for GUI states. (2) Semantic anchors. Tag entities (e.g., "flight-price $514") with stable IDs when first seen; subsequent references use the anchor, so the planner no longer depends on window focus to recall an object. (3) Cross-context consistency checks. Inject lightweight assertions, for example, "clipboard should now contain X" and "target window title equals Y". Violations trigger immediate self-repair instead of long, fruitless trial-and-error loops, cutting the redundant steps that dominate L4 failures.

8. Evaluation setup

To ensure fairness, we evaluated all candidate models through a unified interface compatible with the OpenAI API protocol. Specifically, each model was deployed as an API-style service, and outputs were obtained by sending POST requests to the service endpoint along with the conversation input. For each model, we crafted both system and user prompts strictly based on official documentation or released code. For proprietary models, we designed detailed and effective prompts to elicit high-quality responses as faithfully as possible. Apart from model-specific settings, all other parameters, such as temperature and top-p, were kept consistent across evaluations.

During evaluation, the input and output processing pipeline was tailored to the requirements of each task level.

For L1-GUI Content Understanding and L2-GUI Element Grounding, the input to the model comprised the GUI screenshot paired with either the relevant instruction or the question-options set. Model outputs were assessed using `exact-match` evaluation protocol, analogous to standard practices in grounding and QA tasks. However, given the variability in instruction-following abilities across different models, for example, the QA tasks in L1, we observed that some model outputs could not be reliably parsed. To address this, we implemented a hybrid parsing mechanism based on multiple regular expressions to robustly extract valid answers. In our codebase, we expose a customizable `parse_function` for each method, enabling tailored post-processing strategies to accommodate the unique output formats of various models.

For L3-GUI Task Automation and L4-GUI Task Collaboration, evaluation focused solely on whether the agent successfully achieved the desired end state, without the need to interpret intermediate natural language outputs. Therefore, parsing functions were not required for these levels; instead, we compared the final state directly against predefined success criteria to determine task completion.

References

[1] Reyna Abhyankar, Qi Qi, and Yiying Zhang. Osworld-human: Benchmarking the efficiency of computer-use agents, 2025. 10

[2] Sonnet Anthropic. Claude 3.7 sonnet system card. 2025. 13

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 13

[4] Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, et al. Windows agent arena: Evaluating multi-modal os agents at scale. *arXiv preprint arXiv:2409.08264*, 2024. 2, 3

[5] Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Huichi Zhou, Qihui Zhang, Zhigang He, Yilin Bai, Chujie Gao, Liuyi Chen, et al. Gui-world: A video benchmark and dataset for multimodal gui-oriented understanding. In *The Thirteenth International Conference on Learning Representations*, 2024. 2, 3

[6] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*, 2024. 2

[7] Xingyu Chen, Zihan Zhao, Lu Chen, Danyang Zhang, Jiabao Ji, Ao Luo, Yuxuan Xiong, and Kai Yu. Websrc: a dataset for web-based structural reading comprehension. *arXiv preprint arXiv:2101.09465*, 2021. 1

[8] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, Bangkok, Thailand, 2024. Association for Computational Linguistics. 1, 2, 3

[9] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023. 2

[10] 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. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. 1

[11] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. *arXiv preprint arXiv:2305.11854*, 2023. 1

[12] Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. Assistgui: Task-oriented desktop graphical user interface automation. *arXiv preprint arXiv:2312.13108*, 2023. 1

[13] 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. *arXiv preprint arXiv:2410.05243*, 2024. 1, 13

[14] Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, Yue Wen, Jingya Dou, Fei Tang, Jinzhen Lin, Yulin Liu, Zhenlin Guo, Yichen Gong, Heng Jia, Changlong Gao, Yuan Guo, Yong Deng, Zhenyu Guo, Liang Chen, and Weiqiang Wang. Ui-venus technical report: Building high-performance ui agents with rft, 2025. 2, 3

[15] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024. 1

[16] Yu-Chung Hsiao, Fedir Zubach, Gilles Baechler, Victor Carbune, Jason Lin, Maria Wang, Srinivas Sunkara, Yun Zhu, and Jindong Chen. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199*, 2022. 1

[17] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 13

[18] Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, pages 161–178. Springer, 2024. 2

[19] 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. *arXiv preprint arXiv:2504.07981*, 2025. 2, 3

[20] Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154, 2024. 2

[21] Shuquan Lian, Yuhang Wu, Jia Ma, Zihan Song, Bingqi Chen, Xiawu Zheng, and Hui Li. Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding, 2025. 2, 3

[22] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent, 2024. 1

[23] Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024. 1

[24] Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*, 2024. 2

[25] Xinyi Liu, Xiaoyi Zhang, Ziyun Zhang, and Yan Lu. Ui-e2i-synth: Advancing gui grounding with large-scale instruction synthesis. *arXiv preprint arXiv:2504.11257*, 2025. 2

[26] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer, 2024. 1

[27] Yuhang Liu, Zeyu Liu, Shuanghe Zhu, Pengxiang Li, Congkai Xie, Jiasheng Wang, Xueyu Hu, Xiaotian Han, Jianbo Yuan, Xinyao Wang, et al. Infigui-g1: Advancing gui grounding with adaptive exploration policy optimization. *arXiv preprint arXiv:2508.05731*, 2025. 2, 3

[28] Zhaoyang Liu, Jingjing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, Shenglong Ye, Qingyun Li, Xuan Dong, Yue Yu, Chenyu Lu, YunXiang Mo, Yao Yan, Zeyue Tian, Xiao Zhang, Yuan Huang, Yiqian Liu, Weijie Su, Gen Luo, Xiangyu Yue, Biqing Qi, Kai Chen, Bowen Zhou, Yu Qiao, Qifeng Chen, and Wenhai Wang. Scalecua: Scaling open-source computer use agents with cross-platform data. *arXiv preprint arXiv:2509.15221*, 2025. Preprint. 2, 3

[29] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024. 2

[30] Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025. 2, 3

[31] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022. 1

[32] Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M Tamer Özsu, Aishwarya Agrawal, David Vazquez, et al. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. *arXiv preprint arXiv:2503.15661*, 2025. 2

[33] Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, Nicolas Chapados, M. Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. UI-vision: A desktop-centric GUI benchmark for visual perception and interaction. In *Forty-second International Conference on Machine Learning*, 2025. 2, 3

[34] Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. Screenagent: A vision language model-driven computer control agent. 2024. 1

[35] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025. 1, 13

[36] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728, 2023. 2

[37] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024. 2

[38] Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, et al. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. *arXiv preprint arXiv:2412.19723*, 2024. 1

[39] Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. Gui-g^2: Gaussian reward modeling for gui grounding, 2025. 2, 3

[40] Bowen Wang, Xinyuan Wang, Jiaqi Deng, Tianbao Xie, Ryan Li, Yanzhe Zhang, Gavin Li, Toh Jing Hua, Yu Su, Diyi Yang, Yi Zhang, Zhiguo Wang, Victor Zhong, and Tao Yu. Computer agent arena: Compare & test computer use agents on crowdsourced real-world tasks, 2025. 1

[41] 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, 2025. 2, 3

[42] Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *arXiv preprint arXiv:2506.03143*, 2025. 1

[43] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024. 1, 2

[44] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024. 2, 3

[45] Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis, 2025. 2, 3

[46] Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang,

Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025. 1

[47] Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. Androidlab: Training and systematic benchmarking of android autonomous agents. *arXiv preprint arXiv:2410.24024*, 2024. 2, 3, 10

[48] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*, 2024. 1, 13

[49] Xiao Yang, Jiawei Chen, Jun Luo, Zhengwei Fang, Yinpeng Dong, Hang Su, and Jun Zhu. Mla-trust: Benchmarking trustworthiness of multimodal llm agents in gui environments. *arXiv preprint arXiv:2506.01616*, 2025. 2

[50] Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*, 2024. 1

[51] Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, Ran Xu, Liyuan Pan, Caiming Xiong, and Junnan Li. Gta1: Gui test-time scaling agent, 2025. 2, 3

[52] Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, et al. Mobile-agent-v3: Foundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*, 2025. 3

[53] Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025. 2, 3

[54] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024. 1

[55] Junlei Zhang, Zichen Ding, Chang Ma, Zijie Chen, Qiushi Sun, Zhenzhong Lan, and Junxian He. Breaking the data barrier–building gui agents through task generalization. *arXiv preprint arXiv:2504.10127*, 2025. 1

[56] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024. 1

[57] Beitong Zhou, Zhexiao Huang, Yuan Guo, Zhangxuan Gu, Tianyu Xia, Zichen Luo, Fei Tang, Dehan Kong, Yanyi Shang, Suling Ou, et al. Venusbench-gd: A comprehensive multi-platform gui benchmark for diverse grounding tasks. *arXiv preprint arXiv:2512.16501*, 2025. 2

[58] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. 2