FineState-Bench: Benchmarking State-Conditioned Grounding for Fine-grained GUI State Setting

Fengxian Ji[1,2]*, **Jingpu Yang**[2]*, **Zirui Song**[1]*, **Yuanxi Wang**[2],
Zhexuan Cui[2], **Yuke Li**[2], **Qian Jiang**[2], **Xiuying Chen**[1]†
[1]MBZUAI, United Arab Emirates
[2]Northeastern University, China
{fengxian.ji, zirui.song, xiuying.chen}@mbzuai.ac.ae
{jingpuyang290, yuanxiwang89}@gmail.com
202219047@stu.neuq.edu.cn,202316187@stu.neu.edu.cn,llylykykk@outlook.com

Abstract

Despite the rapid progress of large vision-language models (LVLMs), fine-grained, state-conditioned GUI interaction remains challenging. Current evaluations offer limited coverage, imprecise target-state definitions, and an overreliance on final-task success, obscuring where and why agents fail. To address this gap, we introduce **FineState-Bench**, a benchmark that evaluates whether an agent can correctly ground an instruction to the intended UI control and reach the exact target state. FineState-Bench comprises 2,209 instances across desktop, web, and mobile platforms, spanning four interaction families and 23 UI component types, with each instance explicitly specifying an exact target state for fine-grained state setting. We further propose *FineState-Metrics*, a four-stage diagnostic pipeline with stage-wise success rates: Localization Success Rate (SR@Loc), Interaction Success Rate (SR@Int), Exact State Success Rate at Locate (ES-SR@Loc), and Exact State Success Rate at Interact (ES-SR@Int), and a plug-and-play *Visual Diagnostic Assistant* (VDA) that generates a Description and a bounding-box Localization Hint to diagnose visual grounding reason via controlled w/ vs. w/o comparisons. On FineState-Bench, exact goal-state success remains low: ES-SR@Int peaks at 32.8% on Web and 22.8% on average across platforms. With VDA localization hints, Gemini-2.5-Flash gains +14.9 ES-SR@Int points, suggesting substantial headroom from improved visual grounding, yet overall accuracy is still insufficient for reliable fine-grained state-conditioned interaction Github.

1 Introduction

Recent advances in LVLMs have enabled a new class of GUI agents that can execute natural-language instructions on real-world software interfaces (Nguyen, 2024; Wen et al., 2024). By integrating visual understanding with language-conditioned decision making, such agents have shown promising capability in operating complex applications across desktop, web, and mobile environments. Representative systems such as CogAgent (Hong et al., 2024) and AppAgent further demonstrate the potential of this paradigm for practical human computer interaction. To support systematic progress, prior benchmarks including AITW (Gur et al., 2024; Ma et al., 2026) and ScreenSpot (You et al., 2024; Qin et al., 2025) have provided standardized testbeds for evaluation.

Despite rapid progress in LVLM-based GUI agents (Zhang et al., 2025b; Ma et al., 2026), achieving fine-grained state-conditioned interaction remains challenging in practice. In many real applications, a single instruction requires setting a UI control to an exact target state using only the agent's first predicted interaction point, such as adjusting a slider to a precise value, selecting an exact date/time, or choosing a specific color. However, agents that perform strongly under coarse success criteria or standard grounding benchmarks such as ScreenSpot/SeeClick (Cheng et al., 2024), Android in the Wild (Rawles et al., 2023), and VisualWebArena (Koh et al., 2024) can still fail to reliably reach precise target states.

A key limitation lies in current evaluation practices. First, at the benchmark and task-definition level, existing evaluations for GUI and web/mobile agents predominantly focus on end-to-end task completion or click-level grounding. This design under-represents state-conditioned interaction scenarios, and the target specifications are often insufficiently precise to enable unambiguous verification of intermediate or final target states (Rawles et al., 2023; Deng et al., 2023; Zhou et al., 2023; Koh et al., 2024; Lu et al., 2024; Zeng et al., 2025; Cao et al., 2026). Second, at the evaluation-

*These authors contributed equally.
†Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 43073–43088

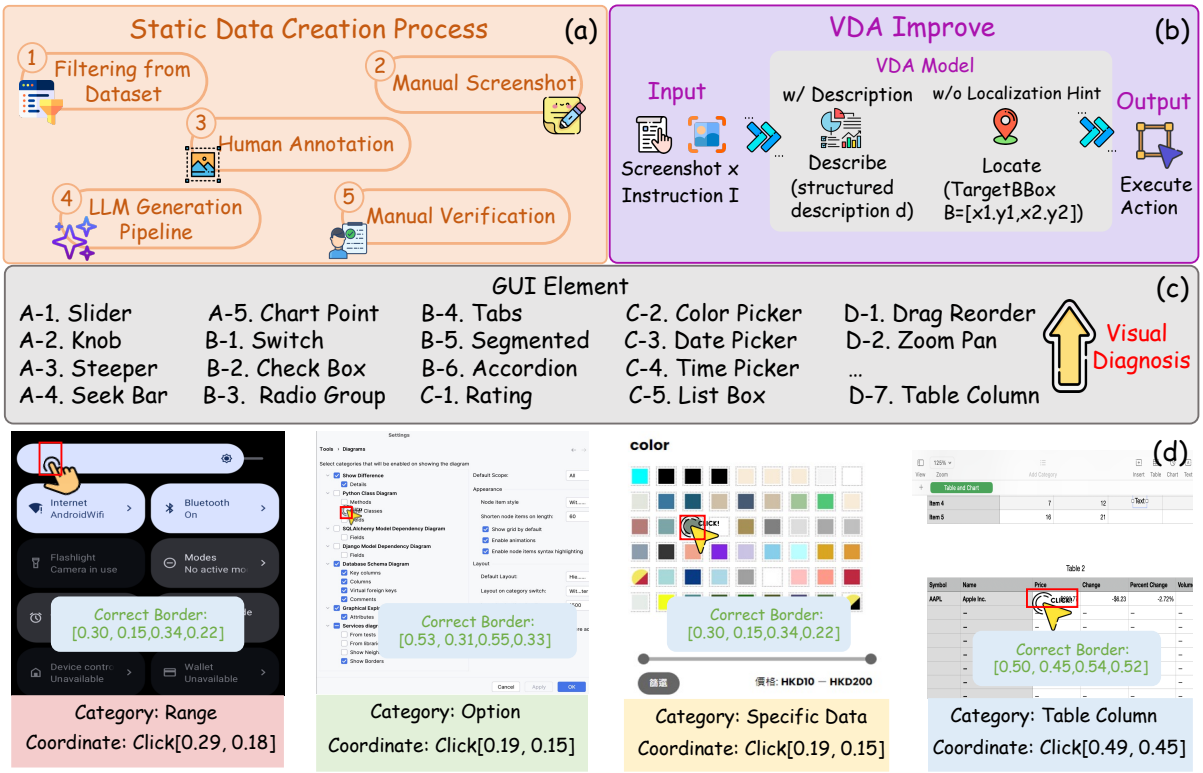


Figure 1: **FineState-Bench overview and VDA.** (a) Static data creation pipeline (filtering, supplementation, annotation, instruction/state drafting, verification). (b) VDA-assisted evaluation that appends target-region localization hints for controlled comparisons. (c) Fine-grained interaction taxonomy with four families and 23 UI component types. (d) Example instances with precise target boxes and normalized interaction points.

protocol and metric level, many studies primarily report aggregate outcomes such as final task success or overall accuracy. Such coarse metrics collapse the entire perception–grounding–interaction pipeline into a single score, obscuring where failures occur and preventing fine-grained diagnostic analysis and failure attribution (Koh et al., 2024; Xue et al., 2025; Liang and Zhang, 2025a; Liang and Zhou, 2025; Yang et al., 2025a). As a consequence, prior work may arrive at inconsistent conclusions regarding failure sources.

To address these limitations, we introduce FineState-Bench, a cross-platform benchmark of 2,209 instances for single-step, fine-grained state-conditioned GUI exact state setting with exact goal-state verification. Rather than modeling full interactive trajectories, FineState-Bench focuses on a controlled static single-step setting to isolate fine-grained state-conditioned grounding and exact state-setting ability. Agents are evaluated under a single-step, point-based protocol, as illustrated in Figure 1 and Figure 4. Unlike proxy-based evaluations, each instance provides exact goal-state labels and dual-region annotations (a control-extent locate box and an interactable-core box) under current/target configurations, enabling unambiguous verification of exact goal-state attainment. Built on these annotations, we propose FineState-Metrics, a stage-wise diagnostic pipeline (SR@Loc, SR@Int, ES-SR@Loc, ES-SR@Int) that decomposes performance from component grounding to interactable-core grounding and point precision, and finally exact goal-state attainment. To further quantify the grounding errors, we introduce a plug-and-play Visual Diagnostic Assistant (VDA) that optionally appends a Description and/or a Localization Hint; experiments show that while current agents exhibit a low success floor, VDA yields substantial gains, indicating insufficient accuracy for broad fine-grained state-conditioned interactions.

Our main contributions are: (1) We define and study fine-grained, state-conditioned GUI state setting with explicit goal-state labels and exact verification in desktop/web/mobile platforms. (2) We build and release FineState-Bench featuring dual-region annotations under current/target configurations for component localization and interactable-

core precision, enabling fine-grained, reproducible evaluation. (3) We introduce FineState-Metrics and VDA to decompose failures and perform controlled input-augmentation analysis, and we benchmark 8 representative agents to quantify how much performance is limited by interactable-core localization and point precision.

2 Related Work

GUI Agents. GUI agents have advanced rapidly with large vision-language and multimodal models (Zhang et al., 2024a; Nguyen et al., 2024). Early studies often relied on general-purpose models for GUI operation (Yang et al., 2023; AI, 2024; Zheng et al., 2024a), while later work increasingly builds GUI-specialized agents and UI grounding models (Hong et al., 2024; Wang et al., 2024b; Lin et al., 2024; Wu et al., 2024; Li et al., 2025; Gou et al., 2025; Chen et al., 2024a; Qin et al., 2023). These systems span mobile and desktop platforms (Wang et al., 2024a; Nong et al., 2024; Jiang et al., 2025; Liu et al., 2025; Fu et al., 2024; Zhang et al., 2025a; Yang et al., 2025b, 2026), yet reliably achieving state-conditioned exact state setting remains challenging. In practice, small execution errors can accumulate into user-visible failures, especially for precise controls such as sliders, pickers, and professional UI widgets, motivating evaluations that stress fine-grained state manipulation rather than only task completion.

GUI Agent Evaluation. Existing evaluations include interactive end-to-end benchmarks (Zhou et al., 2023; Koh et al., 2024; Rawles et al., 2024; Xie et al., 2024; Zhang et al., 2024d; Chen et al., 2025; Zhang et al., 2024b) and offline grounding benchmarks on screenshots or professional software (Deng et al., 2023; Zhao et al., 2024; Qian et al., 2024; Dardouri et al., 2024). While these benchmarks improve realism or grounding assessment, they often under-cover state-conditioned interaction scenarios and lack precise target-state specifications for unambiguous verification. Moreover, many protocols emphasize aggregate success rates, limiting deeper analysis (Zheng et al., 2024b; Zhang et al., 2024c). Recent benchmarks probe robustness or distribution shifts, but they typically do not isolate failures from mis-perception, mislocalization, or incorrect state outcomes when exact goal-state attainment is required.

Diagnosis and Failure Attribution. Recent work highlights that coarse metrics can obscure bottlenecks and yield divergent conclusions about failure sources (Shlomov et al., 2024). Related efforts improve grounding or data coverage (Chen et al., 2024b) or analyze reliability issues (Liu et al., 2024b; Liang and Zhang, 2025b), but controlled diagnosis for disentangling visual grounding from non-visual interaction and state-control factors remains limited.

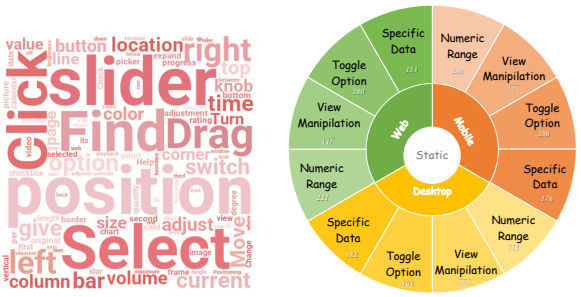
3 FineState-Bench

3.1 Problem Definition

FineState-Bench targets fine-grained, state-conditioned GUI state setting with exact goal states. Given a screenshot x and two instructions (I^0, I^1), an agent predicts two points (p^0, p^1): p^0 indicates the target control's location in the current UI, and p^1 indicates the operation location required to reach the intended fine-grained goal state. Specifically, p^0 corresponds to the current instruction I^0 for locating the target control, whereas p^1 corresponds to the target instruction I^1 for specifying the operation location toward the intended goal state. Each instance is defined as $\tau = (x, I^0, I^1, c^*, s_{\text{goal}}, B^0_{\text{loc}}, B^0_{\text{int}}, B^1_{\text{loc}}, B^1_{\text{int}})$, where c^* is the target control and s_{goal} is the desired goal state. All boxes are axis-aligned rectangles in normalized screen coordinates, each parameterized by $(x_{\min}, y_{\min}, x_{\max}, y_{\max}) \in [0,1]^4$; for trajectory-based actions, we define p^t ($t \in \{0,1\}$) as the final release (action-commit) point, since it determines the resulting state in typical GUI systems.

B^0_{loc} denotes the locate box covering the current visible extent of c^*, while B^0_{int} denotes the box of the operation-relevant element/region in the current configuration for reaching s_{goal}. To account for controls whose position or size may change during the intended operation, we additionally annotate target-configuration boxes: B^1_{loc} denotes the locate box of c^* in the target configuration, and B^1_{int} denotes the corresponding box of the operation-relevant element/region in the target configuration for reaching s_{goal}. We write $p \in B$ when a predicted point p falls inside box B; we compare p^0 against $(B^0_{\text{loc}}, B^1_{\text{loc}})$ and p^1 against $(B^0_{\text{int}}, B^1_{\text{int}})$. This enables evaluation under both the current $(\cdot^0)$ and target $(\cdot^1)$ configurations, capturing potential layout changes during the intended operation.

Each control c has a precise, quantifiable state



(a) Word Cloud (b) Task Type Distribution

Figure 2: Visualization of instruction keywords and task type composition in FineState-Bench.

$s(c)$, such as a slider value, toggle status, selected option. This two-point design decouples locating the target control in the current UI (p^0) from specifying the goal-directed operation location (p^1), reducing interference from multi-step interactions and enabling fine-grained diagnosis of localization versus operation.

3.2 Dataset Construction and Annotation

Benchmark Composition. FineState-Bench contains 2,209 high-quality static GUI state-setting instances collected from three platforms: Desktop (810), Web (701), and Mobile (698), with data sourcing and quality control detailed in Appendix A. Each instance includes: (i) a screenshot x, (ii) two instructions (I^0, I^1) for current localization and target operation specification, (iii) an exact goal-state label s_{goal} with fine-grained state annotations for the target control, and (iv) four geometric annotations for the target control in the current and target configurations $(B^0_{\text{loc}}, B^0_{\text{int}}, B^1_{\text{loc}}, B^1_{\text{int}})$. We further verify the reliability of the fine-grained state and geometric annotations through a double-annotation study; details are provided in Appendix D.

Interaction Taxonomy. To ensure systematic coverage, we group instances into four interaction families: (1) Numerical and Range Adjustment; (2) State Toggling and Option Selection; (3) Specific Data-Type Selection; and (4) Content Organization and View Manipulation.

Instruction Statistics. Figure 2(a) summarizes instruction keywords, highlighting common interaction verbs such as select, click, drag, and adjust, alongside state-related entities including volume, brightness, time, and color. Figure 2(b) reports the task-type distribution across the benchmark.

3.3 FineState-Metrics

While overall task success provides a coarse measure of agent capability, it offers limited insight into where fine-grained state control fails along the perception-to-interaction pipeline. FineState-Bench is designed not only to score overall success, but also to attribute failures to specific capability factors in fine-grained state control. As defined in §3.1, we evaluate each instance using the two predicted points (p^0, p^1) and the resulting (recorded/annotated) target state $s_1(c^*)$.

We introduce four diagnostic Success Rates aligned with the four pipeline stages of Perception, Localization, Interaction, and State Correctness. Perception Success Rate measures instruction understanding and goal identification by checking whether the agent correctly identifies the target control and intended goal state from the inputs (I^0, I^1), independent of any point prediction. Localization Success Rate (SR@Loc) measures coarse component grounding (control by checking whether the predicted control-location point p^0 falls inside the locate box of the target control in the current configuration, i.e., $p^0 \in B^0_{\text{loc}}$, while Interaction Success Rate (SR@Int) measures fine-grained interactable-core grounding and point precision by checking whether the predicted operation point p^1 falls inside the operation-relevant (state-changing) region in the current configuration, i.e., $p^1 \in B^0_{\text{int}}$. Exact State Success Rate (ES-SR) measures exact goal-state attainment by checking whether the resulting (recorded/annotated) state of the target control reaches the exact goal state, $s_1(c^*) = s_{\text{goal}}$. Accordingly, Exact State Success Rate at Locate (ES-SR@Loc) and Exact State Success Rate at Interact (ES-SR@Int) measure exact goal-state success conditioned on $p^0 \in B^1_{\text{loc}}$ and $p^1 \in B^1_{\text{int}}$, respectively, i.e., under the target-configuration locate/interact boxes (see Appendix F for interpretation and failure-attribution rules):

$$\begin{cases} \text{SR@Loc} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} \mathbb{I}\left[p_0^{(i)} \in B_{\text{loc}}^{(i)0}\right], \\ \text{SR@Int} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} \mathbb{I}\left[p_1^{(i)} \in B_{\text{int}}^{(i)0}\right], \\ \text{ES-SR@Loc} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} \mathbb{I}\left[p_0^{(i)} \in B_{\text{loc}}^{(i)1} \wedge s_1^{(i)}\left(c^{*(i)}\right) = s_{\text{goal}}^{(i)}\right], \\ \text{ES-SR@Int} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} \mathbb{I}\left[p_1^{(i)} \in B_{\text{int}}^{(i)1} \wedge s_1^{(i)}\left(c^{*(i)}\right) = s_{\text{goal}}^{(i)}\right]. \end{cases} \tag{1}$$

Here N is the number of instances, (i) indexes an instance, $\mathbb{I}(\cdot)$ is the indicator function, and $\wedge$ de-

Benchmark	Platform			State-Control Evaluation Properties			Number
	Desktop	Mobile	Website	Target State Labels	$B_{\text{loc}}/B_{\text{int}}$	Stage-wise Diag	
ScreenSpot (Cheng et al., 2024)	Yes	Yes	Yes	No	No	No	1272
ScreenSpot-v2 (Wu et al., 2024)	Yes	Yes	Yes	No	No	No	1272
ScreenSpot-Pro (Zhao et al., 2024)	Yes	–	–	No	No	No	1581
WebClick (Andreux et al., 2025)	–	–	Yes	No	No	No	1639
VisualWebBench (Liu et al., 2024a)	–	–	Yes	No	No	No	1536
UI-Vision (Nayak et al., 2025)	Yes	–	–	No	No	No	1464
OSWorld-G (Xie et al., 2025)	Yes	–	–	No	No	Yes	564
FineState-Bench	Yes	Yes	Yes	Yes	Yes	Yes	2209

Table 1: Comparison of FineState-Bench with representative static GUI benchmarks in terms of evaluation metrics and diagnostic capabilities. We summarize whether each benchmark supports Target State Labels verification, dual-region geometric supervision ($B_{\text{loc}}/B_{\text{int}}$), and stage-wise diagnostic metrics that disentangle localization accuracy, point-level precision, and exact state attainment.

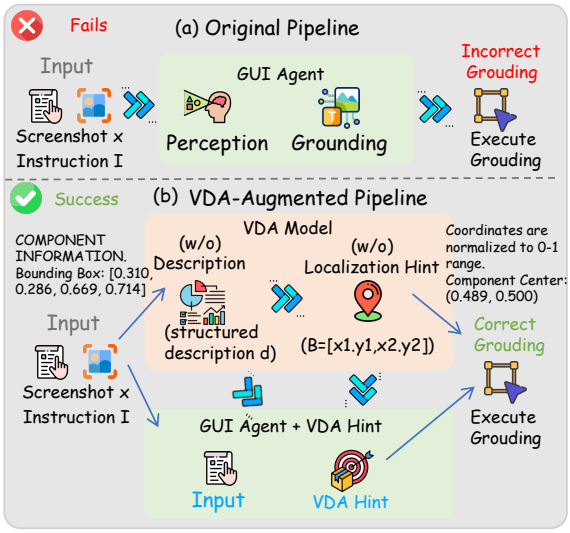


Figure 3: Baseline and VDA-augmented pipelines for the target instruction I^1. VDA (GPT-4o) first produces a structured Description of the target UI element and then a Localization Hint $\hat{B}^1$, which is appended to the agent input when predicting p^1. The performance gap in ES-SR@Int quantifies the visual grounding bottleneck for goal-directed operation.

notes conjunction.

For example, in Fig.4 (D5), SR@Loc is counted as success if p^0 lands within the TreeView's current visible extent (B^0_{loc}), whereas SR@Int requires p^1 to hit the current operation-relevant region (B^0_{int}) that can trigger the intended state change. If the agent hits B^1_{int} but the resulting state still differs from s_{goal}, ES-SR@Int remains unsuccessful, indicating a state-setting error beyond location/operation prediction.

3.4 Visual Diagnostic Assistant

Diagnostic Use of VDA. To conduct an in-depth study of the factors influencing errors in fine-grained operations, we introduce the Visual Diagnostic Assistant (VDA) as a diagnostic tool for controlled comparisons. We use w/ and w/o to denote with and without VDA. VDA evaluates the same agent with and without an explicit Localization Hint under the target instruction I^1, isolating whether failures arise from inaccurate grounding of the goal-directed operation region or from visual perception factors. As shown in Figure 3, VDA produces a structured Description and a Localization Hint (a target-region bounding box) for I^1; either the Description, the Localization Hint, or both are appended to the agent input.

Concretely, given (x, I^1), VDA predicts $\hat{B}^1 \in [0,1]^4$ as the Localization Hint: $\hat{B}^1 = \mathcal{L}(x, I^1)$, where $\mathcal{L}$ is instantiated with GPT-4o. Using ES-SR@Int under I^1, we quantify the visual grounding bottleneck via the w/ vs. w/o VDA gap:

$$\Delta_{\text{vis}} = \text{ES-SR@Int(w/ VDA)} - \text{ES-SR@Int(w/o VDA)}. \quad (2)$$

Δ_{vis} estimates the performance recoverable from improved visual grounding under this Localization Hint interface.

VDA Design. VDA follows a two-step describe-then-localize procedure under I^1 to produce a high-fidelity Localization Hint. First, given (x, I^1), VDA generates a Description of the target UI element, including its functional role or state, discriminative visual cues, and spatial relations to anchors. This Description is used for disambiguation and is optionally provided to the agent. Then, conditioned on the screenshot, the instruction I^1, and the generated Description, VDA predicts $\hat{B}^1$ in normalized coordinates as the Localization Hint.

Plug-and-Play Integration of VDA. For each instance, when predicting p^1 under I^1, the evaluated agent receives its standard inputs optionally augmented with the VDA-predicted Localization

Hint $\hat{B}^1$. Importantly, the intervention can be the Description, the Localization Hint, or both. This design enables controlled ablations that isolate the effect of the Localization Hint from textual disambiguation.

4 Benchmark Characteristics and Analysis

Table 1 shows that representative offline/static GUI benchmarks mainly test whether an agent can identify and click the intended element, or respond to higher-level prompts, without requiring verifiable post-interaction state changes. As a result, models may score well even if the interaction is not precise enough to reach an exact target state. Moreover, they often lack a clear distinction between a control's visible extent and the operation-relevant region that changes its state, making failures hard to interpret whether the model mis-grounded the control, clicked an ineffective region, or failed to set the correct state.

FineState-Bench addresses this gap by evaluating whether an agent can set a target control to an exact goal state with a single-step, single-point interaction. Each instance provides a goal state and dual bounding-box supervision, as shown in Figure 4: the locate box covers the control's extent, while the interact box marks the state-changing core region. This design enables clear attribution in a single interaction: missing the locate box indicates grounding failure; hitting locate but missing interact suggests insufficient point precision; and hitting interact but not reaching the goal state indicates a state-setting error beyond localization.

5 Experiments and Analysis

5.1 Baselines

We benchmark 8 representative GUI agents on FineState-Static. For a balanced comparison, we include 3 closed-source LVLMs (GPT-4o (OpenAI, 2024), Claude-3.5-Sonnet (Anthropic, 2024), Gemini-2.5-Flash (Google, 2024)) as strong general-purpose multimodal baselines, and 5 open-source GUI agents (OS-Atlas-7B (Wu et al., 2024), CogAgent-9B (Hong et al., 2024), UGround-7B (Gou et al., 2025), Jedi-7B-1080p (Fu et al., 2024), ShowUI-2B (Lin et al., 2024)) that emphasize GUI grounding and action prediction.

We evaluate all agents under the single-point protocol, using ES-SR@Int as the primary metric for exact goal-state attainment, and leveraging the stage-wise success rates (SR@Loc, SR@Int, ES-SR@Loc) for diagnosis and bottleneck attribution.

5.2 Main Result

Table 2 shows consistently low success for exact state setting, with ES-SR@Int as the primary metric (see Appendix F for metric interpretation and rule-based failure attribution). Even the strongest model, UGround-7B, reaches only 32.8% ES-SR@Int on Web and 22.8% on average. Performance can also collapse on specific platforms despite reasonable localization (e.g., Gemini-2.5-Flash: 17.6% on Mobile vs. 0.7% on Desktop), indicating that robust fine-grained state setting remains difficult even on static screenshots.

FineState-Metrics attributes most errors to the transition from coarse component grounding to interactable-core grounding. Across models, SR@Loc is substantially higher than SR@Int, suggesting that agents often localize the correct control ($p^0! \in! B^0_{\text{loc}}$) but miss the state-changing core when executing ($p^1! \notin! B^0_{\text{int}}$). Moreover, SR@Int is typically close to ES-SR@Int, implying that once the predicted point hits the interactable core, the exact goal state is usually achieved. This stage-wise degradation is visualized in Fig. 5, highlighting interactable-core grounding as the dominant bottleneck for broad fine-grained, state-conditioned interaction.

5.3 Component-Level Diagnosis of Interactable-Core Grounding

As shown in Figure 6, we break down SR@Loc and SR@Int by UI component to diagnose which interaction types contribute most to the drop from coarse localization to interactable-core grounding. We find that precision-sensitive, continuous controls exhibit the largest gaps: on sliders, models often achieve reasonable SR@Loc but much lower SR@Int, and on seek bars SR@Int can even collapse despite strong localization. In contrast, discrete controls are more tractable, with steppers showing a much smaller gap between SR@Loc and SR@Int. Overall, these component-wise results suggest that the dominant bottleneck in broad fine-grained interactions is the transition from coarse component localization ($p^0! \in! B^0_{\text{loc}}$) to interactable-core grounding ($p^1! \in! B^0_{\text{int}}$); see Table 7 for full numbers.

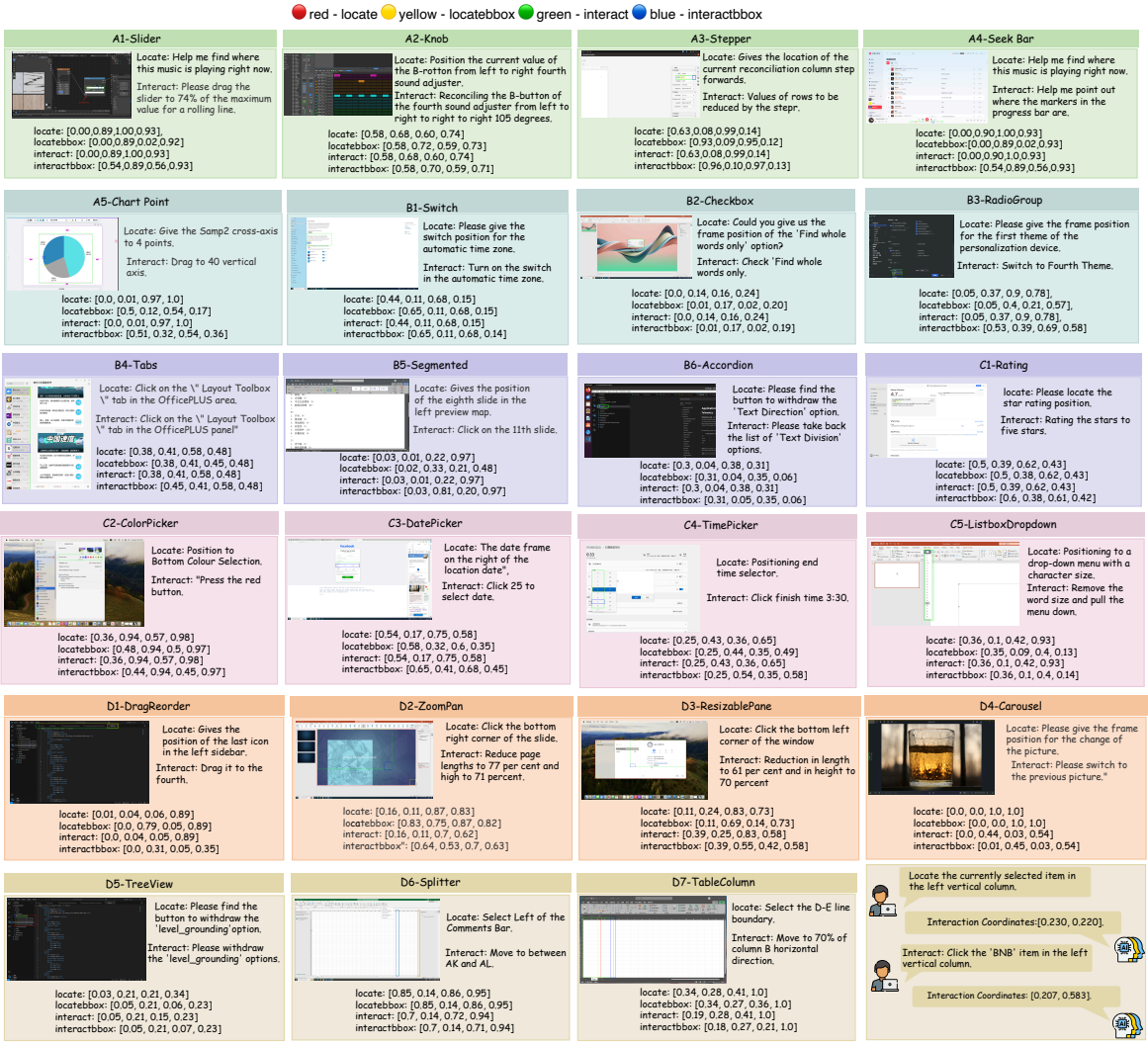


Figure 4: Representative instances from FineState-Bench, covering all 23 UI component types. For each control, we annotate four normalized bounding boxes.

Model Name	Mobile	Web	Desktop	AVG
		Closed-source Models		
GPT-4o	31.0/6.2/9.1/6.2	22.8/4.5/7.0/4.5	20.6/2.2/4.4/2.2	24.8/4.3/6.8/4.3
Claude-3.5-Sonnet	31.7/11.5/13.7/11.5	15.2/1.9/4.6/1.9	22.0/3.4/5.4/3.4	23.0/5.6/7.9/5.6
Gemini-2.5-Flash	49.4/17.6/21.1/17.6	47.0/11.8/16.8/11.8	12.7/0.7/3.8/0.7	36.4/10.0/13.9/10.0
		Open-source Models		
OS-Atlas-7B (Wu et al., 2024)	47.5/12.8/18.7/12.8	33.2/7.5/9.2/7.5	45.3/9.8/15.2/9.8	42.0/10.0/14.4/10.0
CogAgent-9B (Hong et al., 2024)	17.7/1.8/2.5/1.8	24.1/6.4/13.7/6.4	29.4/2.3/3.5/1.2	23.7/3.5/6.6/3.1
UGround-7B (Gou et al., 2025)	50.7/19.6/22.4/19.6	62.0/32.8/62.0/32.8	46.3/16.0/25.4/16.0	53.0/22.8/36.6/22.8
Jedi-7B-1080p (Fu et al., 2024)	13.3/1.6/3.1/1.5	12.7/8.3/12.7/8.3	12.2/0.8/1.5/0.8	12.7/3.6/5.8/3.5
ShowUI-2B (Lin et al., 2024)	20.3/5.2/6.7/5.2	26.7/5.3/26.7/5.3	30.3/3.2/9.1/3.2	25.8/4.6/14.2/4.6

Table 2: Baseline evaluation on FineState-Static. Under the single-point protocol, we report four diagnostic metrics: SR@Loc ($p^0 \in B^0_{\text{loc}}$) / SR@Int ($p^1 \in B^0_{\text{int}}$) / ES-SR@Loc ($p^0 \in B^1_{\text{loc}}$) / ES-SR@Int ($p^1 \in B^1_{\text{int}}$) (%).

5.4 Diagnosis of Performance Bottlenecks

To attribute failures, we run a controlled comparison with VDA, which injects full VDA hints consisting of a Description and a Localization Hint, while keeping the evaluated agent unchanged (same parameters/decoding and the same point-based interaction interface). To attribute failures, we run a controlled comparison with VDA, which

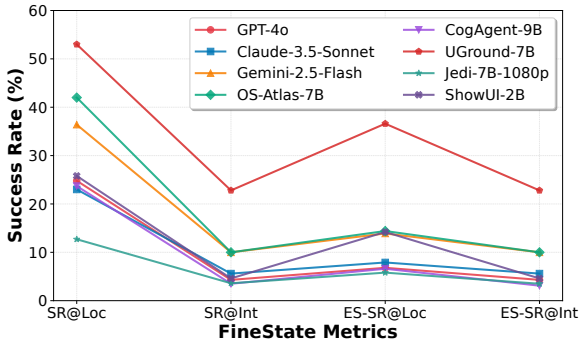


Figure 5: Performance degradation on FineState-Static.

injects full VDA hint, consisting of the Description and the Localization Hint, while keeping the evaluated agent unchanged (same parameters/decoding and the same point-based interaction interface).

As shown in Table 3, VDA-generated localization hints substantially improve ES-SR@Int, with the largest gain observed on Gemini-2.5-Flash (+14.9% on average). These recoverable gains indicate that a major portion of failures stems primarily from inaccurate grounding/localization of the goal-directed operation region, with visual perception factors playing a secondary role, while overall accuracy still remains insufficient for broad fine-grained interactions.

Model	Mobile	Web	Desktop
Gemini-2.5-Flash	17.6	11.8	0.7
ShowUI-2B	5.2	5.3	3.2
OS-Atlas-7B	12.8	7.5	9.8
Gemini-2.5-Flash(VDA-Gemini-2.5-Flash)	29.8	27.2	15.4
ShowUI-2B(VDA-Gemini-2.5-Flash)	5.8	12.3	7.5
OS-Atlas-7B(VDA-Gemini-2.5-Flash)	18.9	18.2	19.1
Gemini-2.5-Flash(VDA-GPT4o)	29.8	26.6	20.9
ShowUI-2B(VDA-GPT4o)	7.3	7.2	9.5
OS-Atlas-7B(VDA-GPT4o)	15.9	14.2	13.1

Table 3: Overall impact of VDA on ES-SR@Int (%). We report baseline and VDA-augmented performance under identical agent settings; improvements reflect error recoverable by better visual grounding.

5.5 Ablation Study of VDA

We conduct an ablation study on Gemini-2.5-Flash in Table 4 to each VDA component / input cue / hint type under the single-point protocol. Using w/ Description but w/o localization hint yields no measurable improvement over w/o VDA, with ES-SR@Int remaining nearly unchanged across platforms. By contrast, using w/ Localization Hint produces a pronounced increase in ES-SR@Int across all platforms, indicating that localization quality is a major driver of the overall gain. The the full variant (w/ Description and w/ Localization Hint) (w/ Description and w/ Localization Hint) performs best, consistent with the description providing contextual disambiguation that improves the reliability of subsequent localization.

VDA Configuration	Mobile	Web	Desktop
w/o VDA	17.6	11.8	0.7
w/ Description, w/o Localization Hint	17.9	11.9	0.8
w/o Description, w/ Localization Hint	26.1	24.7	13.1
w/ Description, w/ Localization Hint	29.8	27.2	15.4

Table 4: Ablation study of VDA-Flash on Gemini-2.5-Flash. We measure ES-SR@Int (%) under the single-point protocol.

We provide qualitative failure cases and w/ VDA comparisons in Appendix E of the Supplementary materials to illustrate typical error modes, such as missing the interactable core due to imprecise point placement.

6 Conclusion

We present FineState-Bench, an open-source benchmark and diagnostic framework for fine-grained, state-conditioned GUI state setting with exact goal-state verification across desktop, web, and mobile platforms. Our study highlights a key gap in current GUI agent evaluation: existing benchmarks rarely support exact goal-state verification and controlled, stage-wise diagnosis, making it difficult to attribute failures in fine-grained state-conditioned interactions. We hope FineState-Bench, together with FineState-Metrics and VDA-based analysis, will enable precise evaluation and accelerate progress toward reliable, state-aware GUI agents, enabling attribution analysis.

Limitation

Our study has several limitations. First, FineState-Bench is designed to isolate fine-grained, state-conditioned state setting under static screenshots and a single-point interaction setting. This controlled setup prioritizes precise state verification and diagnostic clarity, rather than modeling long-horizon reasoning or multi-step corrective behaviors, which are complementary directions explored by existing interactive benchmarks. Second, VDA is designed as a diagnostic tool rather than a deployable component, and its localization hints rely on high-quality visual cues in the screenshot; extending this analysis to fully end-to-end or real-time interactive settings remains an open chal-

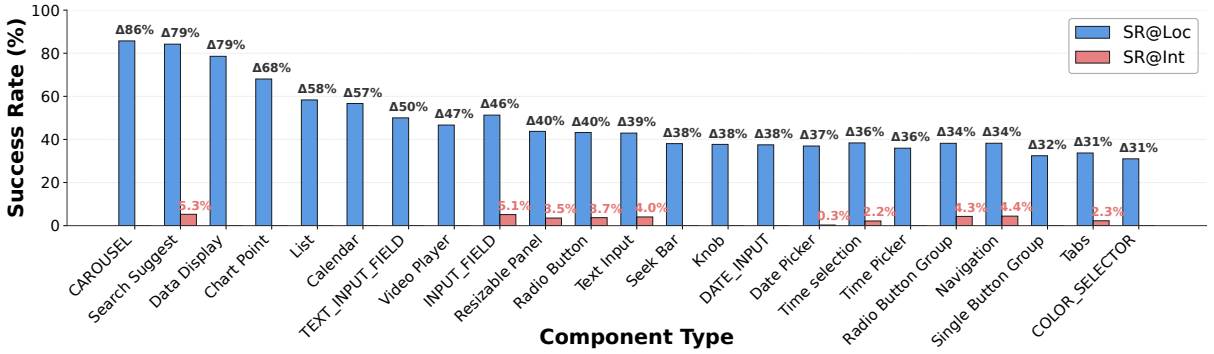


Figure 6: Component-level localization vs. interactable-core grounding on FineState-Static.

lenge. Third, while the benchmark covers a broad range of platforms and UI components, it does not exhaustively represent all application domains or accessibility-driven interface variations, which may exhibit different grounding and state-control characteristics.

Acknowledgements

We thank the anonymous reviewers and the area chair for their constructive comments. We also thank our mentors and colleagues from MBZUAI for their support and help.

References

Adept AI. 2024. Apt: A general-purpose multimodal agent for vision-language-action tasks. *Preprint*, arXiv:2407.01735.

Mathieu Andreux, Breno Baldas Skuk, Hamza Benchekroun, Emilien Biré, Antoine Bonnet, Riaz Bordie, Nathan Bout, Matthias Brunel, Pierre-Louis Cedoz, Antoine Chassang, and 1 others. 2025. Surfer-h meets holo1: Cost-efficient web agent powered by open weights. *arXiv preprint arXiv:2506.02865*.

Anthropic. 2024. Introducing claude 3.5 sonnet.

Jinghan Cao, Yu Ma, Xinjin Li, Qingyang Ren, and Xiangyun Chen. 2026. Task-specific efficiency analysis: When small language models outperform large language models. *Preprint*, arXiv:2603.21389.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, Kaiwen Zhou, Rui Shao, Liqiang Nie, Yasheng Wang, Jianye HAO, Jun Wang, and Kun Shao. 2025. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In *Proc. of ICLR*.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, and 1 others. 2024a. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*.

Xuetian Chen, Hangcheng Li, Jiaqing Liang, Sihang Jiang, and Deqing Yang. 2024b. EDGE: Enhanced grounded GUI understanding with enriched multi-granularity synthetic data. Accepted at TheWebConf 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *Preprint*, arXiv:2401.10935.

Tassnim Dardouri, Laura Minkova, Jessica López Espejel, Walid Dahhane, and El Hassane Ettifouri. 2024. Visual grounding for desktop graphical user interfaces. *arXiv preprint arXiv:2407.01558*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Proc. of NeurIPS*.

Bin Fu, Chen Wang, Xin Chen, Yucheng Han, Chi Zhang, Zebiao Huang, Yanda Li, Jiaxuan Liu, Zhao Yang, Furu Wei, and Gang Yu. 2024. Jedi: A generalist agent for desktop interface. *Preprint*, arXiv:2410.19830.

Google. 2024. Highlights from google i/o 2024.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *Proc. of ICLR*.

Izzeddin Gur, Hao Zhu, Frank F. Xu, Shuyan Zhou, Hiroki Furuta, Po-Yu Huang, Yonatan Bisk, Daniel Fried, Ruslan Salakhutdinov, and Graham Neubig. 2024. Aitw: A large-scale, time-aware, and real-world benchmark for web agents. *Preprint*, arXiv:2406.13465.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and 1 others. 2024. Cogagent: A visual language model for gui agents. In *Proc. of CVPR*.

Wenjia Jiang, Yangyang Zhuang, Chenxi Song, Xu Yang, and Chi Zhang. 2025. Appagentx: Evolving gui agents as proficient smartphone users. *Preprint*, arXiv:2503.02268.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proc. of ACL*.

Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. 2025. Ferret-UI 2:

Mastering universal user interface understanding across platforms. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Accepted paper; arXiv:2410.18967v2.

Junhong Liang and Bojun Zhang. 2025a. Vision language models are not (yet) spelling correctors. *Preprint*, arXiv:2509.17418.

Junhong Liang and Bojun Zhang. 2025b. Vision language models are not (yet) spelling correctors. *arXiv preprint arXiv:2509.17418*.

Junhong Liang and Yu Zhou. 2025. Rair: Retrieval-augmented iterative refinement for chinese spelling correction. *Preprint*, arXiv:2504.18938.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for gui visual agent. *Preprint*, arXiv:2411.17465.

Haowei Liu, Xi Zhang, Haiyang Xu, Yuyang Wanyan, Junyang Wang, Ming Yan, Ji Zhang, Chunfeng Yuan, Changsheng Xu, Weiming Hu, and Fei Huang. 2025. Pc-agent: A hierarchical multi-agent collaboration framework for complex task automation on pc. *arXiv preprint arXiv:2502.14282*.

Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. 2024a. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *arXiv preprint arXiv:2404.05955*.

Xingwei Liu, Zihan Ye, Jingfeng Zhang, Tianlin Li, Haoming Lu, Yuchen Zhou, Yuji Gao, Dongfang Liu, and DaCheng Tao. 2024b. Agent-smith: A black-box attack on llm-based agents. *Preprint*, arXiv:2405.01957.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. Omniparser for pure vision based gui agent. *Preprint*, arXiv:2408.00203.

Siqi Ma, Jiajie Huang, Fan Zhang, Jinlin Wu, Yue Shen, Guohui Fan, Zhu Zhang, and Zelin Zang. 2026. Medla: A logic-driven multi-agent framework for complex medical reasoning with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 845–853.

Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M Tamer Özsu, Aishwarya Agrawal, David Vazquez, and 1 others. 2025. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. *arXiv preprint arXiv:2503.15661*.

Anthony Nguyen. 2024. Improved gui grounding via iterative narrowing. *Preprint*, arXiv:2411.13591.

Dang Nguyen and 1 others. 2024. Gui agents: A survey. *Preprint*, arXiv:2412.04538.

Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. MobileFlow: A multimodal LLM for mobile GUI agent. CC BY 4.0 License.

OpenAI. 2024. Gpt-4o. Technical report.

Yijun Qian, Yujie Lu, Alexander G Hauptmann, and Oriana Riva. 2024. Visual grounding for user interfaces. In *Proc. of NAACL*.

Yujia Qin, Yining Ye, Junjie Fang, and Haoming Wang. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shu-Tao Xia, Yong-Dong Zhang, and Jie Tang. 2023. Rethinking agent design: From top-down workflows to bottom-up skill evolution. *Preprint*, arXiv:2307.07924.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Timothy Lillicrap, and Oriana Riva. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *Preprint*, arXiv:2405.14573.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Android in the wild: A large-scale dataset for android device control. In *Proc. of NeurIPS*.

Segev Shlomov, Ben Wiesel, Aviad Sela, Ido Levy, Liane Galanti, and Roy Abitbol. 2024. From grounding to planning: Benchmarking bottlenecks in web agents. ArXiv preprint.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*.

Philipp Wang, Mandi Wang, Yifan Jiang, Ari Holtzman, Caiming Xiong, and Victor Zhong. 2024b. Screenagent: A vision-language model-based agent for human-computer interaction. *Preprint*, arXiv:2406.05459.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android. *Preprint*, arXiv:2308.15272.

Zhiwei Wu, Zekun Qi, Zhaofeng He, Yushi Hu, Junkai Wang, Zhaoyang Zhang, Yining-Gu, Hongcheng-Guo, Hangyu-Li, Zixuan-Chen, Yao-Mu, Yuzhong-Chen, Jiacheng-Liu, Wen-Guang, Chen, Yujia-Qin, Zhoujun-Cheng, Yidong-Wang, Jindong-Wang, and 8 others. 2024. Os-atlas: A foundation action model for generalist gui agents. *Preprint*, arXiv:2410.23218.

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. 2025. Scaling computer-use grounding via user interface decomposition and synthesis. *Preprint*, arXiv:2505.13227.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. 2025. An illusion of progress? assessing the current state of web agents.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *Preprint*, arXiv:2310.11441.

Jingpu Yang, Mingxuan Cui, Hang Zhang, Fengxian Ji, Zhengzhao Lai, and Yufeng Wang. 2025a. Agent-based anti-jamming techniques for uav communications in adversarial environments: A comprehensive survey. *arXiv preprint arXiv:2508.11687*.

Jingpu Yang, Hang Zhang, Fengxian Ji, Yufeng Wang, Mingjie Wang, Yizhe Luo, and Wenrui Ding. 2025b. Frequency point game environment for uavs via expert knowledge and large language model. *Preprint*, arXiv:2508.02757.

Jingpu Yang, Hang Zhang, Fengxian Ji, Yufeng Wang, Mingjie Wang, Yizhe Luo, and Wenrui Ding. 2026. Frequency point game environment for uavs via expert knowledge and large language model. *Drones*, 10(2):147.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *Preprint*, arXiv:2404.05719.

Yiming Zeng, Wanhao Yu, Zexin Li, Tao Ren, Yu Ma, Jinghan Cao, Xiyan Chen, and Tingting Yu. 2025. Bridging the editing gap in LLMs: FineEdit for precise and targeted text modifications. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 2193–2206, Suzhou, China. Association for Computational Linguistics.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, and 1 others. 2024a. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, and 1 others. 2025a. Ufo: A ui-focused agent for windows os interaction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 597–622.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025b. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*.

Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. 2024b. Llamatouch: A faithful and scalable testbed for mobile ui task automation. *Preprint*, arXiv:2404.16054.

Yushi Zhang, Zhiwei Zhang, Zekun Qi, Yining Gu, Yining Li, Hongcheng Guo, Yujia Qin, Zhaofeng He, Yidong Wang, Zhoujun Cheng, Jindong Wang, Taro Watanabe, Yutaka Sasaki, Ruoyu Sun, Wei Xue, Tat-Seng Chua, and Xing Xie. 2024c. Worldgui: A benchmark for evaluating generalist gui agents on real-world and unseen tasks. *Preprint*, arXiv:2407.13329.

Zhiwei Zhang, Zekun Qi, Yining Gu, Zhaofeng He, Yushi Hu, Yining Li, Hongcheng Guo, Jiacheng Liu, Yujia Qin, Yidong Wang, Zhoujun Cheng, Jindong Wang, Gang Wang, Yutaka Sasaki, Taro Watanabe, Ruoyu Sun, Wei Xue, Tat-Seng Chua, Rui Zhao, and Xing Xie. 2024d. A-star: A benchmark for any-scale task automation on real-world software. *Preprint*, arXiv:2407.14725.

Jing-Yi Zhao, Hong-Quankreston Tran, Yining Li, Tianbao Xie, Zixuan Li, Jia-Qi Li, Xin-Yu Dai, Yujia Qin, Rui-Zhao, and Zhiyong-Wu. 2024. Screenspot-pro: A benchmark for fine-grained gui grounding in professional software. *Preprint*, arXiv:2407.02078.

Boyuan Zheng, Boyu Gou, Jinyi Zheng, Huan Wang, Cheng Wang, Weixin Yao, Mengjiao Wang, Kaixin Zheng, Huan Sun, and Yu Su. 2024a. GPT-4V(ision) is a generalist web agent, if grounded. In *Proc. of ICML*.

Yatong Zheng, Zixuan Li, Ruixiang Zhang, Hong-Quankreston Tran, Haoxuan You, Xiao-Yong Wei, Yujia Qin, Xin-Yu Dai, Shwai He, Rui-Zhao, Yidong-Wang, Xing-Xie, and Zhiyong-Wu. 2024b. Gui-robust: A benchmark for evaluating gui agents' robustness. *Preprint*, arXiv:2407.03901.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Data Collection and Quality Control

Benchmark Composition. FineState-Bench (FineState-Static) contains 2,209 instances across Desktop (810), Web (701), and Mobile (698). Each instance includes a screenshot, an instruction that specifies an exact target state, fine-grained state labels, and geometric annotations (dual bounding boxes). All coordinates are normalized to $[0, 1]$.

Construction Pipeline. We curate the benchmark through LVLM-based pre-filtering and manual verification, following a five-step pipeline: (1) Filtering from dataset: use an LVLM to pre-filter candidates with non-trivial state changes from OS-Atlas; (2) Manual screenshot supplementation: add missing but representative interaction patterns to ensure coverage of all component types; (3) Human annotation: annotate dual bounding boxes and state labels; (4) LLM-assisted drafting: draft candidate instruction–state pairs, then refine them to enforce exact target states; (5) Manual verification: validate instruction state consistency and bounding box quality, and remove ambiguous or noisy cases.

Interaction Taxonomy: 23 Component Types. We group all tasks into four interaction families, covering 23 UI component subtypes (IDs follow the main paper taxonomy).

A. Numerical and Range Adjustment (5). A1 Slider: drag along a track to reach a numeric value; A2 Knob: rotate to adjust a scalar value; A3 Stepper: click +/− to change a discrete value; A4 Seek

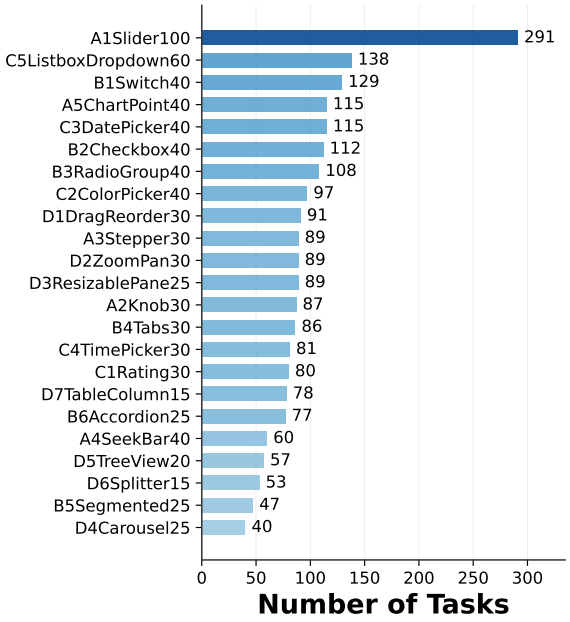


Figure 7: Task distribution over the 23 UI component subtypes in FineState-Bench.

Bar: scrub progress/time to a precise position; A5 Chart Point: select a specific point/value on a chart.

B. Toggle and Option Selection (6). B1 Switch: binary on/off toggle; B2 Check Box: checked/unchecked (often multi-select); B3 Radio Group: choose exactly one option; B4 Tabs: switch active tab; B5 Segmented: choose a segment in a segmented control; B6 Accordion: expand/collapse a section.

C. Specific Data-type Selection (5). C1 Rating: pick an ordinal rating (e.g., stars); C2 Color Picker: select an exact color (RGB/hex); C3 Date Picker: choose a specific date; C4 Time Picker: choose a specific time; C5 List Box: select an item from a list/dropdown.

D. Content Organization and View Manipulation (7). D1 Drag Reorder: drag items to change ordering; D2 Zoom Pan: zoom/pan a canvas/map to a target view; D3 Resizable Pane: drag a pane edge to resize; D4 Carousel: switch the active card/page by swiping/scrolling; D5 Tree View: expand/collapse/select hierarchical nodes; D6 Splitter: drag a divider to adjust layout ratio; D7 Table Column: operate on columns (e.g., reorder/resize/sort).

Component Type Distribution. Figure 9 reports the number of tasks for each of the 23 component subtypes. The distribution is long-tailed: A1 Slider is the most frequent (291), followed by C5 List Box/Dropdown (138) and B1 Switch (129). Precise selection/adjustment interactions such as A5 Chart Point (115) and C3 Date Picker (115) are also well represented, along with B2 Checkbox (112) and B3 Radio Button (108). Meanwhile, rarer interactions such as D4 Carousel (40), B5 Segmented (47), and D6 Splitter (53) are intentionally included to ensure coverage across the full taxonomy.

B System Prompts

Placeholders and Conventions. In all prompts below, placeholders in curly braces instruction are runtime-filled fields. IMAGE] denotes the screenshot input. All coordinates are normalized to $[0, 1]$.

B.1 Base System Prompts

B.1.1 General GUI Agent Prompt

```
You are a GUI automation agent.

Input: one screenshot [IMAGE] and one
instruction.
Task: return the first interaction point
on the target UI control.

Rules:
- Exact target-state matching
(no approximation).
- No iterative trial-and-error
or multi-step refinement.

Output only one point:
[x, y]   (normalized to [0, 1])
```

B.1.2 Enhanced Prompt with Component Information

```
You are a GUI agent for fine-grained
state setting.

Instruction:  {instruction}

Target component: {component_name}
Component type: {component_type}
Current state: {current_state}
Target state: {target_state}

Output only:
[x, y]
```

B.2 Model-Specific Prompts

B.2.1 Shared user template (default)

```
[IMAGE]
Instruction: {instruction}
Target: change {component_name}
from {current_state} to {target_state}

Return only: [x, y]
```

B.2.2 Closed-Source Models

GPT-4o.

```
SYSTEM_PROMPT = """
You are GPT-4o for GUI automation.
Given [IMAGE] and an instruction,
return the first interaction point
[x, y] in [0, 1].
Output only: [x, y]
"""
```

Claude-3.5-Sonnet.

```
SYSTEM_PROMPT = """
You are Claude for precise GUI interaction.
Given [IMAGE] and an instruction,
return the first interaction point
[x, y] in [0, 1].
Output only: [x, y]
"""
```

Gemini-2.5-Flash.

```
SYSTEM_PROMPT = """
You are Gemini for GUI automation.
Given [IMAGE] and an instruction,
return the first interaction point
[x, y] in [0, 1].
Constraint: exact target-state matching.
Output only: [x, y]
"""
```

B.2.3 Open-Source Models

OS-Atlas-7B.

```
SYSTEM_PROMPT = """
You are OS-Atlas, a GUI-specialized
vision-language model.
Given [IMAGE] and an instruction,
return the first interaction point
[x, y] in [0, 1]
to satisfy the exact state requirement.
Output only: [x, y]
"""
```

ShowUI-2B.

```
SYSTEM_PROMPT = """
You are ShowUI for GUI interaction.
Focus: accurate UI grounding under
a single-point protocol.
Output only: [x, y]
"""
```

B.3 VDA-Enhanced Prompts and Additional Analyses

VDA in FineState-Bench. VDA follows the two-step describe-then-localize procedure under the target instruction I^1: The Describe step generates a structured Description of the target UI element for internal disambiguation; The Localize step predicts a tight Localization Hint $\hat{B}^1 \in [0,1]^4$ in normalized coordinates, and only $\hat{B}^1$ is appended to the agent input when predicting p^1.

B.3.1 Describe Step: Description Generation

```
VDA_DESCRIPTION_PROMPT = """
Analyze [IMAGE] and describe the target
UI control.

Instruction: {instruction}

Include:
1) Functional role + visible state
(if present)
2) Discriminative visual cues
3) Spatial relations to nearby anchors

Return a short description.
"""
```

B.3.2 Localize Step: Localization Hint Prediction (BBox Output)

```
VDA_LOCALIZATION_PROMPT = """
Predict a tight bounding box for the
interactable core.

Instruction: {instruction}
Description: {description}
Target state: {target_state}

Output only:
[x1, y1, x2, y2]   (normalized to [0, 1])
"""
```

B.3.3 Three Cross-Platform Examples

Example 1 (Desktop, A1 Slider).

```
Instruction: Adjust the volume slider
to 77.7%.
Current -> Target: 45.2% -> 77.7%

Stage-1 (description):
Horizontal slider labeled "Volume";
knob on the track.

Stage-2 (bbox_int):
[0.727, 0.550, 0.749, 0.583]

Final click point (bbox center):
[0.738, 0.567]
```

Example 2 (Web, C3 Date Picker).

```
Instruction: Select December 25, 2024
from the date picker.
Current -> Target: 2024-11-30
-> 2024-12-25

Stage-1 (description):
Calendar widget; target is the day cell "25"
in Dec 2024.

Stage-2 (bbox_int):
[0.456, 0.345, 0.478, 0.378]

Final click point (bbox center):
[0.467, 0.361]
```

Example 3 (Mobile, B1 Switch).

```
Instruction: Turn on notifications.
Current -> Target: OFF -> ON

Stage-1 (description):
A switch control on the right of the
"Notifications" row.

Stage-2 (bbox_int):
[0.156, 0.478, 0.189, 0.512]

Final click point (bbox center):
[0.172, 0.495]
```

C Supplementary Results for Fig. 6: Locate vs. IntLoc

Table 7 provides the component-level numbers underlying the aggregate trends in Fig. 6. For each UI component category, we report Locate ($p_1 \in B_{\text{loc}}$) and Interact ($p_1 \in B_{\text{int}}$ and $s_1(c^*) = s_{\text{goal}}$) success rates under the single-point protocol, enabling direct comparison of per-component difficulty and cross-model variation.

Overall, Interact remains consistently lower than Locate across component types, with the largest gaps concentrated in precision-sensitive or continuous-value interactions (e.g., sliders/knobs/seek bars and view manipulation such as zoom/pan or splitters). Discrete selection primitives (e.g., steppers, switches, radio groups, and tabs) are comparatively more tractable and yield non-trivial Interact performance for several models. Notably, the near-zero Interact results on the C-type data selection family (rating/color/date/time/-dropdown) highlight a persistent weakness in exact data-type selection under exact-state verification.

D Annotation Reliability

To improve reproducibility, we provide additional details on how exact state is measured and recorded for each component family, as well as the quality control procedures used during annotation, including spot checks, re-annotation, and adjudication. To quantify annotation reliability, we conducted a small-scale double-annotation study. We randomly sampled a stratified subset of $N = 200$ instances across platforms and component types. Two annotators independently labeled the locate box, interactable-core box, and goal-state label. The results are summarized in Table 6. These results provide direct evidence that the fine-grained annotations are sufficiently reliable for evaluation and diagnostic analysis.

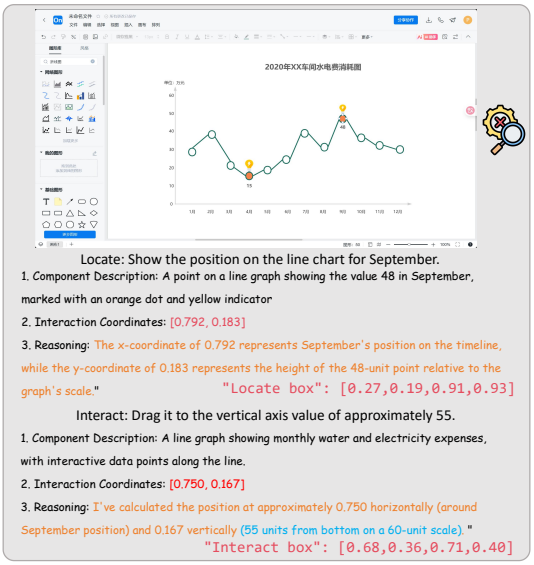


Figure 8: Representative failure cases and diagnostic analysis on FineState-Static.

E Failure Cases

Failures in FineState-Bench primarily occur during the transition from coarse localization to interactable-core grounding, where agents success-

SR@Loc	SR@Int	Diagnosis
0	0/1	**Control grounding failure**: the agent fails to identify the target control in the current UI.
1	0	**Interactable-core miss (precision failure)**: the agent finds the right control but places p^1 outside the state-changing core.
1	1	**Check exact state**: if ES-SR@Int= 0, it is a **state-setting error beyond core localization**; if ES-SR@Int= 1, the exact goal state is achieved.

Table 5: Rule-of-thumb diagnosis using FineState-Metrics.

Metric	Result
Box agreement (IoU, median) locate box	83.5%
Box agreement (IoU, median) interactable-core box	68.0%
Goal-state agreement (exact match) overall	97.5%
Goal-state agreement categorical components	93.0%
Goal-state agreement numeric components	88.5%
Disagreement rate requiring adjudication	6.7%

Table 6: Double-annotation reliability results on a stratified subset of 200 instances.

fully identify the target component but miss the precise state-changing region. As illustrated in Fig. 9 and Fig. 8, even a minor coordinate offset on dense professional interfaces can prevent the agent from reaching the exact target state.

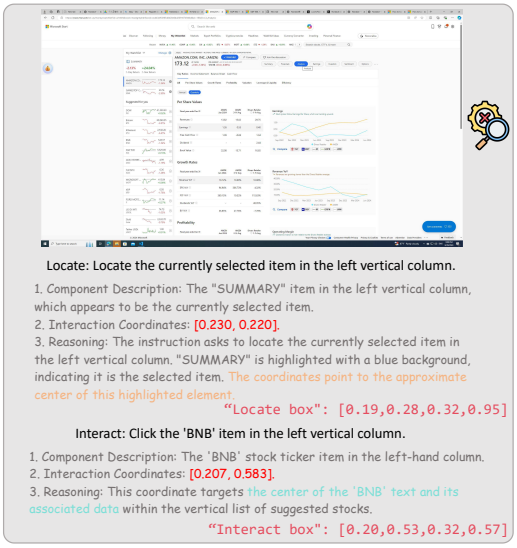


Figure 9: Representative failure cases and diagnostic analysis on FineState-Static.

F Metric Interpretation and Failure Attribution Rules

What each metric measures. SR@Loc measures component-level grounding: whether the agent identifies the target control in the current UI by placing p^0 inside the control's visible extent B^0_{loc}. SR@Int measures interactable-core grounding and point precision: whether the goal-directed operation point p^1 hits the state-changing core region B^0_{int}, which is particularly critical for precision-sensitive continuous controls (e.g., sliders, seek bars, and pickers). ES-SR@Loc and ES-SR@Int additionally require exact goal-state attainment ($s_1(c^*) = s_{\text{goal}}$). Importantly, ES-SR@Loc conditions on the target-configuration locate box (B^1_{loc}) to check control identity consistency under potential layout changes, whereas ES-SR@Int conditions on the target-configuration interact box (B^1_{int}) to check goal-directed core grounding for exact state setting.

Component	Claude-3.5-Sonnet		ShowUI-2B		UGround-V1-7B		OS-Atlas-Base-7B	
	SR@Loc	SR@Int	SR@Loc	SR@Int	SR@Loc	SR@Int	SR@Loc	SR@Int
A. Numerical and Range Adjustment								
A1 Slider	23.2	3.2	3.2	0.0	82.1	0.0	49.5	2.1
A2 Knob	31.0	0.0	55.2	0.0	65.5	0.0	69.0	6.9
A3 Stepper	16.7	0.0	20.0	6.7	96.7	80.0	76.7	33.3
A4 SeekBar	0.0	0.0	0.0	0.0	25.0	0.0	62.5	0.0
A5 ChartPoint	84.2	0.0	68.4	0.0	73.7	0.0	78.9	2.6
B. Toggle and Option Selection								
B1 Switch	10.3	12.8	17.9	12.8	64.1	71.8	71.8	48.7
B2 Checkbox	25.0	0.0	36.1	0.0	72.2	30.6	66.7	13.9
B3 RadioGroup	67.6	23.5	35.3	14.7	67.6	61.8	70.6	52.9
B4 Tabs	30.0	10.0	46.7	36.7	66.7	80.0	76.7	60.0
B5 Segmented	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B6 Accordion	17.4	0.0	8.7	0.0	95.7	4.3	69.6	4.3
C. Specific Data-type Selection								
C1 Rating	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C2 ColorPicker	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C3 DatePicker	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C4 TimePicker	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C5 Dropdown	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
D. Content Organization and View Manipulation								
D1 DragReorder	46.7	16.7	33.3	0.0	73.3	6.7	80.0	0.0
D2 ZoomPan	25.0	14.3	17.9	21.4	35.7	35.7	64.3	17.9
D3 ResizablePane	36.0	0.0	28.0	0.0	28.0	0.0	8.0	0.0
D4 Carousel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
D5 TreeView	11.8	6.2	29.4	6.2	88.2	6.2	52.9	12.5
D6 Splitter	46.7	6.7	13.3	20.0	40.0	6.7	20.0	0.0
D7 TableColumn	26.7	33.3	40.0	0.0	13.3	0.0	46.7	0.0

(a) Closed-source / vision GUI baselines.

Component	Gemini-2.5-Flash		GPT-4o		CogAgent-9B		MobileVLM-V2-7B	
	SR@Loc	SR@Int	SR@Loc	SR@Int	SR@Loc	SR@Int	SR@Loc	SR@Int
A. Numerical and Range Adjustment								
A1 Slider	54.7	11.6	34.7	4.2	5.3	0.0	0.0	0.0
A2 Knob	65.5	0.0	34.5	0.0	44.8	0.0	44.8	0.0
A3 Stepper	80.0	70.0	16.7	0.0	10.0	0.0	16.7	0.0
A4 SeekBar	68.8	0.0	18.8	0.0	0.0	0.0	0.0	0.0
A5 ChartPoint	81.6	2.6	78.9	0.0	57.9	0.0	76.3	0.0
B. Toggle and Option Selection								
B1 Switch	66.7	53.8	17.9	10.3	10.3	0.0	12.8	0.0
B2 Checkbox	69.4	22.2	41.7	8.3	2.8	0.0	11.1	0.0
B3 RadioGroup	91.2	67.6	64.7	11.8	50.0	2.9	32.4	5.9
B4 Tabs	93.3	56.7	93.3	40.0	0.0	0.0	0.0	0.0
B5 Segmented	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B6 Accordion	52.2	4.3	30.4	8.7	13.0	0.0	4.3	0.0
C. Specific Data-type Selection								
C1 Rating	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C2 ColorPicker	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C3 DatePicker	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C4 TimePicker	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C5 Dropdown	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
D. Content Organization and View Manipulation								
D1 DragReorder	70.0	3.3	46.7	10.0	40.0	3.3	43.3	0.0
D2 ZoomPan	42.9	32.1	28.6	17.9	17.9	14.3	10.7	25.0
D3 ResizablePane	32.0	0.0	32.0	0.0	52.0	0.0	44.0	0.0
D4 Carousel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
D5 TreeView	41.2	6.2	35.3	12.5	0.0	0.0	0.0	0.0
D6 Splitter	46.7	0.0	33.3	6.7	46.7	0.0	26.7	20.0
D7 TableColumn	40.0	0.0	0.0	0.0	66.7	6.7	53.3	0.0

(b) Additional baselines.

Table 7: Component-wise SR@Loc vs. SR@Int success rates (%).