IGG: A Benchmark for Interactive GUI Grounding under Visibility Constraints

Kyeongseon Kim[*,1], **Jiyeon Son**[*,1], **Tae-Hyun Oh**[1]
[1]KAIST
Correspondence: ella94.ai@gmail.com

Abstract

GUI grounding benchmarks ask whether an agent can localize a target that is already visible in a static screenshot. Real interfaces often violate this assumption: a target may be off-screen, appear only after a state change or hover, remain occluded, or become actionable after a delay. This leaves the ability to recover target visibility unevaluated. To measure this reveal step, we introduce ***Interactive GUI Grounding (IGG)***[1], a benchmark in which an agent must first make the target visible through interaction and then click it. IGG contains 601 tasks across 16 controllable GUI environments and seven visibility-constraint sub-types, with a compact action space of scrolling, moving, dragging, waiting, and clicking. In our human study, participants achieve 94.6% accuracy, while GPT-5.5 reaches 79.4% and the strongest evaluated open-source VLM reaches only 31.6%. To identify the source of this gap, we conduct two controlled interventions. Telling Qwen3-VL which interaction to perform raises its accuracy only from 30.8% to 33.9%, whereas performing the interaction for the model raises it to 81.9%. The bottleneck is executing the reveal interaction, not recognizing the target once it is exposed. We further show that this capability is learnable, demonstrating that visibility recovery can be strengthened consistently across model families.

1 Introduction

GUI agents increasingly rely on vision-language models (VLMs) to map natural-language instructions to screen locations (Wu et al., 2025; Mittal et al., 2026; Koh et al., 2024; Kumbhar et al., 2026; Gao et al., 2024; Zheng et al., 2024). Existing grounding benchmarks evaluate this mapping within a single screenshot and typically assume that the target is already visible (Cheng et al., 2024; Li et al., 2025). However, routine interfaces often violate this assumption (Figure 2): a target may lie outside the viewport (Burigat et al., 2012), sit in a collapsed panel, be covered by a popup, be distinguishable from its neighbors only on hover, or activate only after a delay. In each case, no static grounding model, however accurate, can localize the target in the initial screenshot. The agent must first select and execute an interaction that reveals it, and only then recognize and click it. Static grounding benchmarks leave this reveal step unmeasured, and navigation benchmarks fold it into end-to-end task success, where it cannot be separated from planning. In light of this gap, this paper asks whether current GUI agents can make a target visible before grounding it and, when they cannot, at which stage they fail. To fill this gap, we introduce ***Interactive GUI Grounding (IGG)*** to measure this missing step. An IGG episode decomposes success into four stages: *selecting a reveal action*, *executing it at the correct location or with the correct parameters*, *recognizing the target once it appears*, and *localizing the final click*. We call the first two stages *visibility recovery*. The remaining stages, recognizing the target once it appears and localizing the click, coincide with static grounding (Section 3.1). Thus, like static grounding, IGG scores a single final click. Unlike static grounding, it additionally requires the agent to recover the target's visibility beforehand. Because interaction serves only to expose the target, IGG isolates visibility recovery without introducing the broader planning demands of full GUI navigation.

IGG contains 601 tasks across 16 controllable GUI environments modeled on familiar applications. Tasks fall into seven sub-types under three constraint types, *Viewport*, *Multi-State*, and *Temporal/Manipulation*, and are solved with five actions: SCROLL, MOVE, DRAG, WAIT, and CLICK. De-

[*]Equal contribution.
[1]The benchmark and code are available at https://igg-bench.github.io/.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 61475–61498
October 24-29, 2026 ©2026 Association for Computational Linguistics

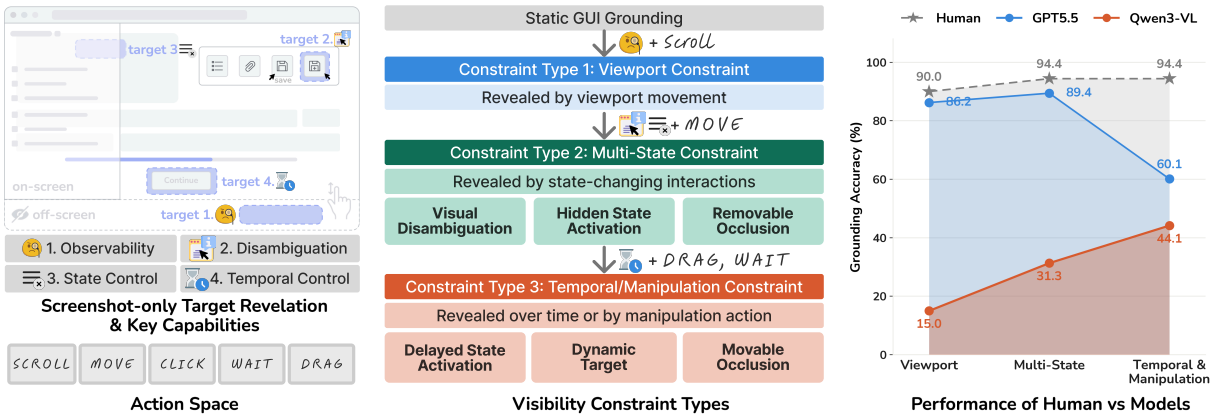


Figure 1: **Overview of Interactive GUI Grounding (IGG). Left**: representative visibility constraints require observability, disambiguation, state control, or temporal control before the target can be clicked. **Center**: these capabilities and a compact action space define *Viewport Constraint*, *Multi-State Constraint*, and *Temporal/Manipulation Constraint*. **Right**: human solvability exceeds model grounding accuracy on all three constraint types, and the gap is largest on viewport constraints for Qwen3-VL and on temporal and manipulation constraints for GPT-5.5.

terministic initial states and stable target identities make evaluation replayable at click time, while the environments preserve the interaction patterns that make visibility recovery difficult: hover reveals, popups, delayed activation, carousels, and movable occlusions.

Our experiments identify interaction execution as the main bottleneck in IGG. Telling a model which interaction to perform provides little improvement, whereas performing the reveal action for it recovers most of its accuracy. SFT on successful interaction trajectories followed by online GRPO further improves visibility recovery across Qwen3-VL, InternVL3.5, and GUI-Owl-1.5.

Our contributions are:

- We formulate interactive GUI grounding as visibility recovery followed by target recognition and localization, separating it from both static grounding and end-to-end navigation.
- We introduce IGG, a benchmark of 601 tasks, 16 controllable GUI environments, and seven visibility-constraint sub-types, together with a click-time evaluator and human-verified ground-truth interaction trajectories.
- We provide controlled interventions, heuristic analyses, cross-viewport and paired mock-real evaluations, and training studies that identify where current agents fail and show consistent improvements across three VLM architectures.

2 Related Work

GUI grounding. GUI grounding maps a natural-language instruction to the corresponding interface element (Cheng et al., 2024; Tang et al., 2025). ScreenSpot (Cheng et al., 2024), ScreenSpot-Pro (Li et al., 2025), and OSWorld-G (Xie et al., 2025) evaluate localization within a screenshot where the target is visible. Recent methods improve this setting through rich screenshot supervision (Gao et al., 2024), diffusion-based prediction (Kumbhar et al., 2026), UI decomposition (Xie et al., 2025), and multi-turn refinement (Mittal et al., 2026). Universal grounding extends localization to agent settings (Gou et al., 2025), but still predicts a target from the currently observed state. IGG instead evaluates the preceding visibility-recovery step: the agent must change the interface before the target can be recognized and localized.

GUI navigation Mind2Web (Deng et al., 2023), WebArena (Zhou et al., 2024), OSWorld (Xie et al., 2024) evaluate multi-step task completion on web and desktop interfaces, and agent-side improvements such as interaction-history compression (Zhu et al., 2025) are measured against their end-to-end success. Because only end-to-end success is scored, planning, exploration, and grounding failures collapse into a single signal. IGG instead scores one narrower capability, visibility recovery followed by a single evaluated click (Section 3.1). Just as static grounding was separated from navigation and became a standard evaluation axis (Cheng et al.,

Table 1: Minimal action space $\mathcal{A}$ for visibility recovery.

Action	Role
SCROLL$(\text{dir}, \Delta_{\text{px}})$	Scroll to reveal off-screen regions.
MOVE$(x_{\text{px}}, y_{\text{px}})$	Move the pointer to trigger hover states or expose disambiguating cues.
DRAG(startX, startY, endX, endY)	Drag an element between the specified start and end coordinates.
WAIT(Δt_{ms})	Pause for a specified duration to handle delayed or asynchronous UI updates.
CLICK$(x_{\text{px}}, y_{\text{px}})$	Click the specified location; grounding is evaluated at this step.

2024), IGG isolates a step that navigation benchmarks contain only implicitly, and it complements rather than replaces them.

3 The IGG Benchmark

Our goal is to measure visibility recovery in isolation and reproducibly. To this end, we formalize the task (Section 3.1), categorize the interactions it requires (Section 3.2), describe the environments and how tasks were built (Section 3.3), specify the evaluation protocol (Section 4.1), and validate the benchmark with human studies (Section 4.5).

3.1 Problem formulation

An IGG task is specified by an initial interface state in which the target is not visible or not actionable, a goal-only instruction q that identifies the target but not the procedure for exposing it, a target element with a stable identifier, and an evaluator that scores a CLICK as successful if and only if the clicked element's identifier matches the target's. Each task also carries one human-verified ground-truth trace, which is withheld from the agent and used only for efficiency metrics. At step t, the agent receives the screenshot s_t, the instruction q, and its action history $h_{t-1} = (a_1, \ldots, a_{t-1})$, selects an action from

$$\mathcal{A} = \{\text{SCROLL}, \text{MOVE}, \text{DRAG}, \text{WAIT}, \text{CLICK}\},$$

and receives the next screenshot s_{t+1} (Table 1). IGG uses screenshot-only observations and restricts interaction to these five reveal-and-click actions. The episode ends with success on the first click that hits the target and with failure when the action budget is exhausted. A click elsewhere applies any defined state transition, and the episode continues.

3.2 Visibility constraint taxonomy

We categorize tasks by the interaction required to recover target visibility (Figures 1 and 2).

Constraint Type 1: Viewport Constraint. The target is outside the current viewport but reachable within the same UI state.

- **CT1-1. Scroll-based Exploration:** The target is off-screen and must be revealed by scrolling within the current state.

Constraint Type 2: Multi-State Constraint. The target, or the visual cue needed to identify it, is hidden by the current UI state and must be revealed through a state-changing interaction.

- **CT2-1. Visual Disambiguation:** The target is visually similar to other elements and can only be distinguished through hover-based cues such as tooltips or labels, requiring a MOVE before the final CLICK.
- **CT2-2. Hidden State Activation:** The target is contained in a collapsed or inactive UI region, such as a dropdown, tab, or side panel, that must be opened before clicking.
- **CT2-3. Removable Occlusion:** A blocking element, such as a popup or modal, hides the target and must be dismissed before the target can be clicked.

Constraint Type 3: Temporal & Manipulation Constraint. Reaching the target requires temporal control or spatial manipulation.

- **CT3-1. Delayed State Activation:** The target is initially inactive and becomes interactable only after a temporal delay or asynchronous event, requiring a WAIT before clicking.
- **CT3-2. Dynamic Target Interaction:** The target resides in an automatically transitioning element, such as a carousel or ticker, and must be reached by waiting for the correct state or by using navigation controls.
- **CT3-3. Movable Occlusion:** A UI element occludes the target due to its position or size, requiring a DRAG to reposition or resize the occluding element.

3.3 Environments and task construction

We want environments that are deterministic enough to score a click programmatically, yet rich enough in interaction to make visibility recovery difficult. IGG comprises 601 tasks across 16 controllable environments modeled on widely used

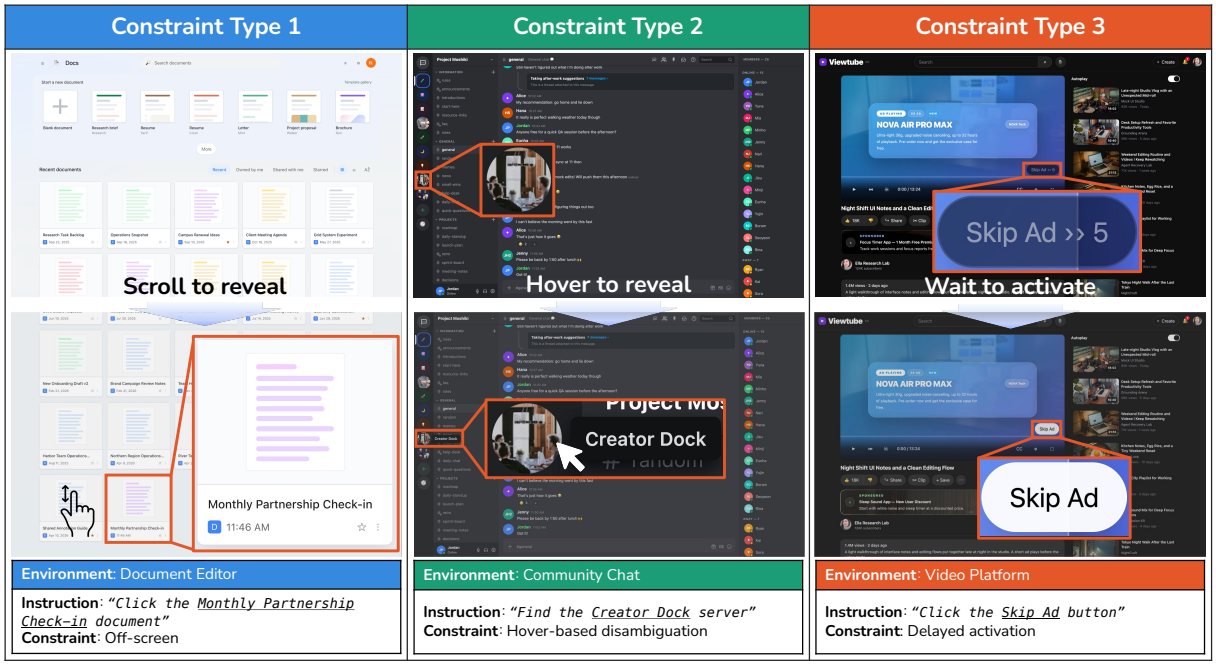


Figure 2: Examples of the three IGG constraint types: viewport, multi-state, and temporal or manipulation constraints. Additional examples appear in Appendix E.2.

applications such as Gmail, Google Calendar, VS Code, Figma, Spotify, and YouTube (full list in Appendix A).[2] Each environment is a deterministic web application: every task starts from a fixed initial state, state transitions are reproducible, and every element carries a stable identifier, so the evaluator decides success at click time without heuristic matching. At the same time, the environments implement the interaction patterns that visibility recovery requires, including hover reveals, collapsible panels, popups, delayed activation, carousels, and draggable occluders.

Tasks are built in three steps. A multimodal agent first drafts a goal-only instruction and a reference trace for each target. Annotators then rewrite instructions that leak the procedure ("Click 'Reply' on Minji's message" rather than "Hover over the message and click 'Reply' ", as in Figure A10) and add a natural disambiguating cue, such as an author, title, or surrounding content, when several elements share a label (Appendix C.3). Finally, annotators replay every trace to confirm that the target is unique and reachable. Figure 3 shows the resulting distribution over sub-types and environments.

[2]Real applications are named as canonical references only, and no affiliation with or endorsement by their owners is implied.

4 Evaluation and Experiments

4.1 Evaluation setup

IGG combines task specifications, 16 controllable GUI environments, and a Playwright-driven Chromium evaluator at a 1440×900 viewport. On each `CLICK`, the evaluator compares the clicked element ID with the target ID; a match ends the episode, while a mismatch returns the updated screenshot. Models receive only screenshots, the goal-only instruction, and action history. The DOM, accessibility tree, target ID, and ground-truth interaction trajectory remain hidden. Each baseline is evaluated once per task after an environment reset; main runs use an eight-action budget and greedy decoding unless noted. Table D provides the prompts, decoding settings, and action formats.

Metrics. We report the following metrics:

- **Grounding Accuracy (GA) ↑ :** the fraction of tasks in which the agent `clicks` the target element within the action budget.
- **Action Cost (AC) ↓ :** mean number of actions executed until success or budget exhaustion.
- **Reference Efficiency Ratio (RER) ↑ :** mean of $|g_i|/|a_i|$ over successful tasks, where g_i is the annotated ground-truth trajectory and a_i is the agent trajectory; values above 1 indicate trajec-

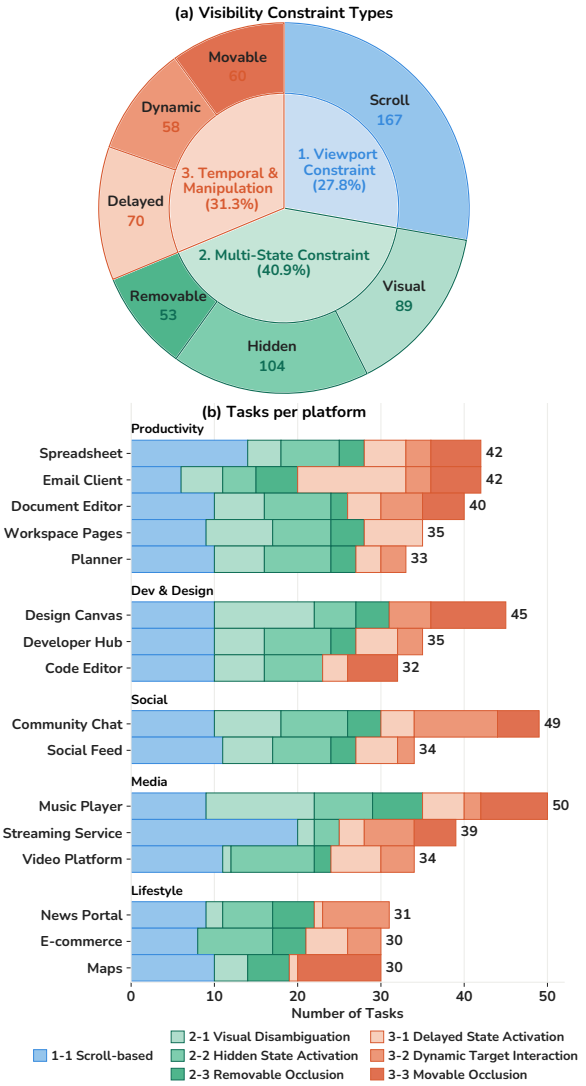


Figure 3: Distribution of IGG tasks. (a) Task distribution across the three constraint types and their sub-types. (b) Number of tasks per platform, broken down by sub-type.

tories shorter than the annotated ground-truth trajectory.

- **Total Time (TT) ↓ :** mean elapsed wall-clock time per task.

GA measures success; AC and RER measure interaction efficiency. TT is reported descriptively: local and API-backed models use different serving stacks, so their runtimes are not hardware-normalized.

4.2 Main results

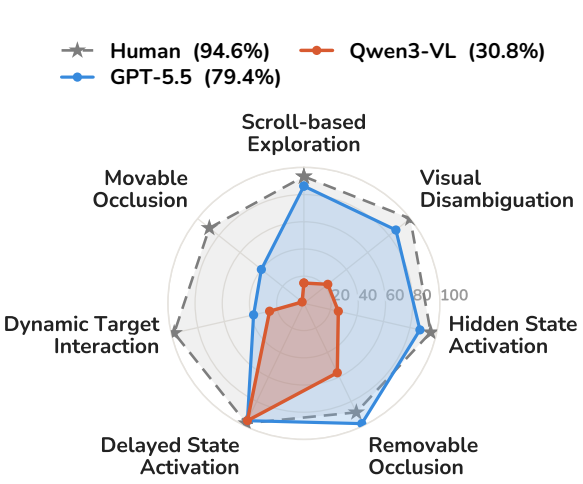
We compare Qwen2.5-VL-7B (Bai et al., 2025b), Qwen3-VL-8B (Bai et al., 2025a), and GUI-Owl-1.5-8B (Xu et al., 2026) with the commercial reference model GPT-5.5.

Table 2: Results by constraint type and sub-type.

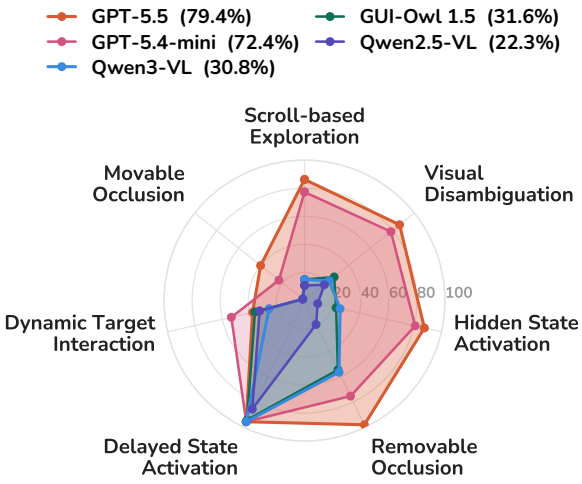
Model	GA (%) ↑	AC ↓	RER ↑	TT (ms) ↓
Constraint Type 1: Viewport Constraint				
Qwen2.5-VL-7B	10.8	7.37	0.70	**14718**
Qwen3-VL-8B	15.0	7.26	**0.73**	15246
GUI-Owl-1.5-8B	15.0	7.27	0.62	21985
GPT-5.5	**86.2**	**4.29**	0.65	56355
Constraint Type 2: Multi-State Constraint				
Qwen2.5-VL-7B	14.6	7.20	1.16	**8773**
Qwen3-VL-8B	31.3	6.21	1.13	10862
GUI-Owl-1.5-8B	31.3	6.14	**1.16**	14682
GPT-5.5	**89.4**	**3.28**	1.08	49103
Constraint Type 2-1: Visual Disambiguation				
Qwen2.5-VL-7B	18.0	6.96	1.52	**8048**
Qwen3-VL-8B	22.5	6.51	**1.73**	11287
GUI-Owl-1.5-8B	27.0	6.22	1.71	13350
GPT-5.5	**86.5**	**2.91**	1.40	44982
Constraint Type 2-2: Hidden State Activation				
Qwen2.5-VL-7B	9.6	7.44	0.93	**9181**
Qwen3-VL-8B	26.0	6.54	**0.97**	11251
GUI-Owl-1.5-8B	23.1	6.67	0.87	17408
GPT-5.5	**87.5**	**3.73**	0.91	56739
Constraint Type 2-3: Removable Occlusion				
Qwen2.5-VL-7B	18.9	7.13	0.81	**9193**
Qwen3-VL-8B	56.6	5.08	0.89	9384
GUI-Owl-1.5-8B	54.7	4.96	**0.95**	11569
GPT-5.5	**98.1**	**3.02**	0.89	41038
Constraint Type 3: Temporal & Manipulation Constraint				
Qwen2.5-VL-7B	42.6	6.02	0.52	24467
Qwen3-VL-8B	44.1	6.08	0.63	**10907**
GUI-Owl-1.5-8B	46.8	5.78	0.73	11864
GPT-5.5	**60.1**	**4.82**	**0.97**	134642
Constraint Type 3-1: Delayed State Activation				
Qwen2.5-VL-7B	85.7	4.57	0.50	19820
Qwen3-VL-8B	**95.7**	3.74	0.63	**6626**
GUI-Owl-1.5-8B	94.3	3.53	0.67	7412
GPT-5.5	**95.7**	**2.50**	**0.97**	20603
Constraint Type 3-2: Dynamic Target Interaction				
Qwen2.5-VL-7B	32.8	**5.81**	0.58	45702
Qwen3-VL-8B	25.9	7.02	0.60	**12631**
GUI-Owl-1.5-8B	36.2	6.33	0.83	13558
GPT-5.5	**37.9**	6.47	**0.84**	134949
Constraint Type 3-3: Movable Occlusion				
Qwen2.5-VL-7B	1.7	7.90	1.00	**9362**
Qwen3-VL-8B	1.7	7.90	1.00	14235
GUI-Owl-1.5-8B	1.7	7.88	**2.00**	15421
GPT-5.5	**40.0**	**5.93**	1.10	267392
Overall				
Qwen2.5-VL-7B	22.3	6.88	0.72	15334
Qwen3-VL-8B	30.8	6.46	0.85	**12094**
GUI-Owl-1.5-8B	31.6	6.34	0.89	15830
GPT-5.5	**79.4**	**4.04**	**0.92**	77876

Table 2 reports grounding accuracy (GA) and efficiency by constraint type and sub-type; Table 3 aggregates GA by application. Figure 4 compares humans and model baselines, and Appendix A10 provides extended per-application metrics.

Figure 5 shows the corresponding accuracy-efficiency trade-off. On the validation comparison, GPT-5.5 is substantially more accurate, but averages 66.5 seconds per task compared with 14.2 seconds for the trained Qwen3-VL and requires paid API inference. Evaluating GPT-5.5 over the



(a) Humans, GPT-5.5, and open-source VLMs.



(b) Additional model baselines.

Figure 4: **IGG exposes a large interaction gap.** Human results summarize the solvability study, whereas model results span the complete benchmark.

Table 3: Grounding Accuracy (GA) (%) by application.

App	Qwen2.5-VL	Qwen3-VL	GUI-Owl-1.5	GPT-5.5
Social Feed	23.5	29.4	38.2	**61.8**
Planner	15.2	21.2	24.2	**81.8**
Developer Hub	11.4	25.7	25.7	**57.1**
Code Editor	25.0	21.9	28.1	**87.5**
Community Chat	22.4	26.5	34.7	**61.2**
Document Editor	32.5	37.5	42.5	**85.0**
Design Canvas	8.9	8.9	13.3	**82.2**
Spreadsheet	16.7	33.3	23.8	**83.3**
Email Client	47.6	50.0	54.8	**88.1**
News Portal	9.7	29.0	29.0	**71.0**
Workspace Pages	25.7	34.3	22.9	**85.7**
Streaming Service	7.7	20.5	17.9	**87.2**
E-commerce	16.7	43.3	33.3	**86.7**
Music Player	20.0	36.0	32.0	**86.0**
Maps	16.7	26.7	26.7	**70.0**
Video Platform	55.9	50.0	58.8	**94.1**
Overall (601)	22.3	30.8	31.6	**79.4**

Table 4: Grounding accuracy (%) when a model solves the complete task, is told the required interaction, or receives the interface after the reveal interaction has been performed.

Condition	Qwen3-VL	GPT-5.4 Mini (Low)
Full task	30.8	72.4
Interaction specified	33.9	–
Reveal performed	81.9	90.4

complete benchmark cost approximately $74.9.

4.3 Where does interactive grounding fail?

We intervene at two points in the four-stage decomposition from Section 3.1. In the first condition, the model is told which reveal interaction is required but must determine its location and parameters. In the second, we perform the ground-truth reveal trajectory and ask the model only for the final click.

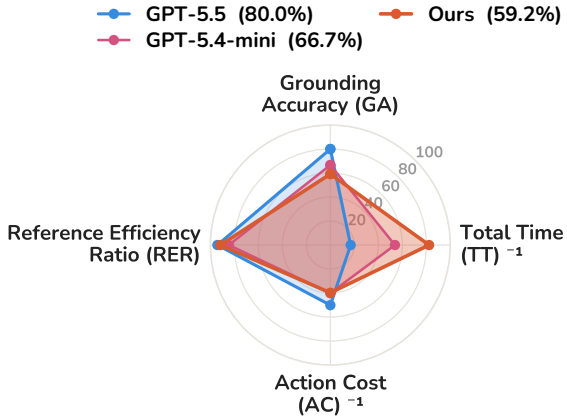


Figure 5: **Accuracy-efficiency trade-off on the validation set.** GPT-5.5 offers substantially higher accuracy, while the trained Qwen3-VL is faster and runs without paid API inference.

Naming the interaction changes Qwen3-VL by only 3.1 points, whereas completing it raises accuracy by 51.1 points. Providing the correct reveal-action category is therefore insufficient. Most remaining errors arise from choosing its location or parameters and executing the reveal correctly, before target recognition and final-click localization are evaluated.

Target-agnostic heuristics further separate easy and hard reveal patterns (Appendix F.3). Fixed scrolling, hovering, and waiting reveal 68.8%, 61.4%, and 87.1% of their matched tasks, whereas popup dismissal and occluder dragging reach 17.6% and 16.9%. The difficult cases require identifying which element controls visibility and how to manipulate it.

Failed Qwen3-VL trajectories show the same pattern. When the ground-truth trajectory requires DRAG, SCROLL, or MOVE, failed runs omit that action in 59/59, 207/247, and 71/96 cases, respectively (Appendix F.2). This motivates first expanding the policy's action support with SFT, then improving execution with online feedback.

4.4 Learning missing interaction priors

We ask whether training can supply reveal actions that base VLMs rarely use. DRAG provides the clearest case: Qwen2.5-VL, Qwen3-VL, and GUI-Owl-1.5 emit no drag actions on the held-out tasks.

Setup. After excluding two tasks whose ground-truth trajectories could not be replayed reliably, we use 479 tasks for training and reserve 120 for evaluation. The held-out tasks contribute neither SFT examples nor GRPO rollouts.

Supervised training on successful trajectories (Hu et al., 2022). We fine-tune the model on human-verified successful trajectories. Let x_t denote the instruction, screenshot, and interaction history at step t, and let a_t be the next ground-truth action. We optimize a weighted next-action negative log-likelihood (Pomerleau, 1991):

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathrm{E}_{(x_t,a_t)\sim\mathcal{D}}\left[w(a_t)\log \pi_\theta(a_t \mid x_t)\right].$$

For standard behavior cloning, $w(a_t) = 1$. In the frequency-balanced variant (Cui et al., 2019), we increase $w(a_t)$ for rare reveal actions such as MOVE, WAIT, and DRAG; details are in Appendix G. The model still receives goal-only instructions. Only the training distribution changes.

Refining interactions with online feedback (Shao et al., 2024). We then apply online GRPO under the same eight-action interaction budget as main evaluation. The reward is defined over browser-executed trajectories rather than text strings:

$$R(\tau) = R_{\mathrm{out}}(\tau) + R_{\mathrm{proc}}(\tau) - \lambda_{\mathrm{step}}|\tau| - R_{\mathrm{err}}(\tau).$$

R_{out} rewards click-time target success, R_{proc} gives bounded credit for visibility-changing progress, and R_{err} penalizes wrong clicks, empty clicks, invalid actions, and repeated ineffective actions. Exact weights and optimization hyperparameters are provided in Appendix G.

The two stages serve different roles. SFT broadens action support, after which GRPO selects among those actions using online feedback. If a seed policy never samples DRAG, GRPO observes no successful drag rollout to reinforce. Frequency-balanced SFT brings rare reveal actions into the policy's sampled support, enabling GRPO to optimize their selection and execution.

Table 5: Human task solvability on the 21-task suite (15 participants, 315 trials).

Sub-type	Trials	Success (%) ↑	Time (s) ↓
CT1-1	45	93.3	23.3
CT2-1	45	100.0	15.9
CT2-2	45	95.6	20.0
CT2-3	45	88.9	21.0
CT3-1	45	97.8	14.2
CT3-2	45	97.8	22.9
CT3-3	45	88.9	21.1
Overall	315	94.6	19.8

Table 6: Human fidelity ratings averaged over 15 participants on a five-point Likert scale: visual realism (VR), interaction realism (IR), and task executability (TE).

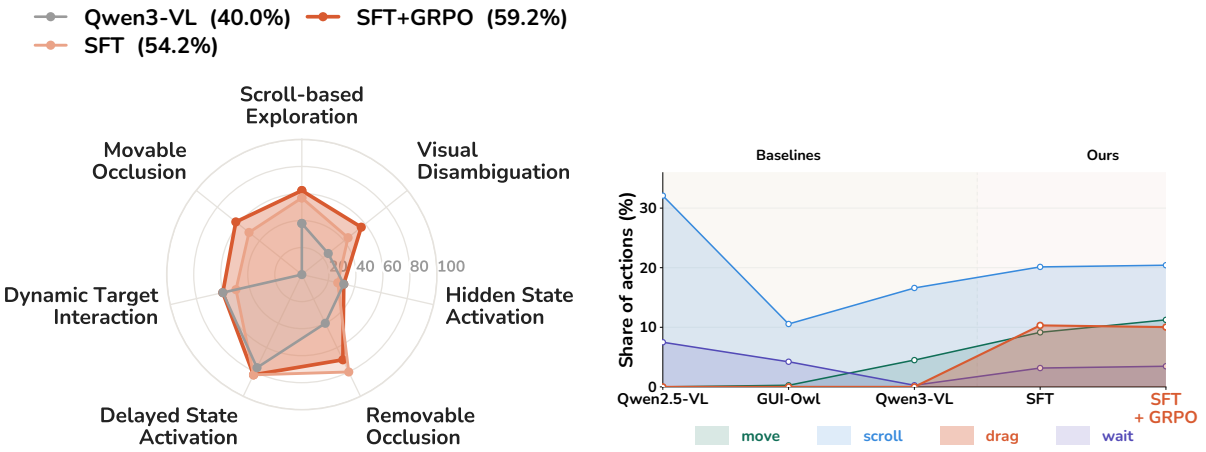
Environment	VR ↑	IR ↑	TE ↑
Music Player	4.93	4.80	5.00
Video Platform	4.73	4.87	4.80
E-commerce	4.67	4.80	4.80
Developer Hub	4.67	4.53	4.80
Code Editor	4.67	4.80	4.87
News Portal	3.93	4.60	4.60
Maps	3.93	4.07	4.40
Average	4.50	4.64	4.75

Results. The two-stage procedure raises Qwen3-VL from 40.0% to 59.2% GA (Figure 6). On Movable Occlusion, frequency-balanced SFT emits DRAG in 7/8 required-drag tasks and solves 4/8; GRPO raises this to 5/8. Initialization also matters on Dynamic Target Interaction: GRPO reaches 60.0% from the frequency-balanced seed but 30.0% from the standard SFT seed. The same procedure improves InternVL3.5 from 28.3% to 47.5% and GUI-Owl-1.5 from 35.0% to 52.5% (Table A8), showing consistent gains across three architectures.

4.5 Human solvability and environment fidelity

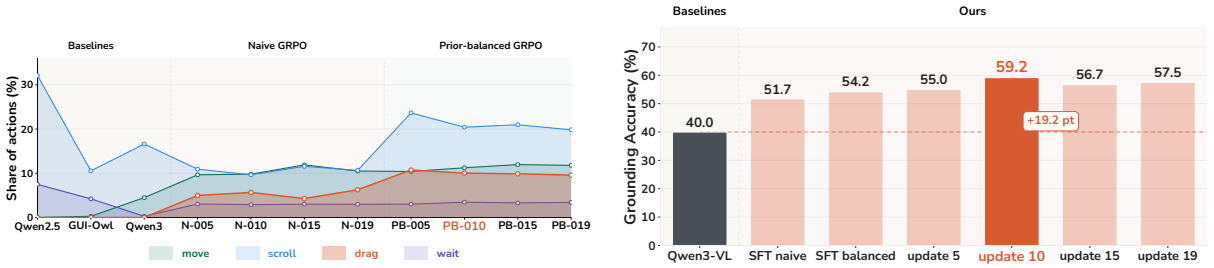
A failure on IGG is informative only if the task is solvable and the environment is faithful to the interface it models. Before comparing models, we therefore verify both with the same 15 participants (protocol details in Appendix H).

Task solvability. Participants solve a fixed suite of 21 tasks, three from each sub-type, using the



(a) Per-sub-type gains from baseline to SFT and SFT+GRPO. (b) Action-prior shift across baselines, SFT, and SFT+GRPO.

Figure 6: **SFT expands action support, and GRPO selects among those actions.** The best trained model improves Qwen3-VL from 40.0% to 59.2% GA on the 120-task validation split. Baseline VLMs solve some validation tasks without emitting every action in the ground-truth trajectory, but emit no drag actions. SFT introduces these actions into the policy's support, after which GRPO improves task success.



(a) Action share across GRPO variants.

(b) Validation GA across GRPO checkpoints.

Figure 7: **GRPO ablation.** GRPO is effective only when initialized from the frequency-balanced SFT seed: it reaches 59.2% GA, while GRPO from the standard SFT seed does not improve over its seed model. The frequency-balanced seed gives the policy access to rare reveal actions, after which online reward improves their selection and execution.

same instructions and interfaces given to the models, without the constraint label, target ID, target coordinates, or reveal procedure. A trial ends on the correct target click or after eight click-or-drag attempts. Scrolling and hovering are recorded but do not count, so the limit is comparable to, though not identical with, the eight-action model budget. Participants succeed in 298 of 315 trials (94.6%) with a mean completion time of 19.8 seconds (Table 5). Together with the replay verification of every reference trace, this indicates that the model failures reported below do not stem from unsolvable tasks or ambiguous instructions.

Environment fidelity. The same participants compare seven representative environments with their real-interface counterparts and rate visual realism, interaction realism, and task executability on five-point scales, giving mean scores of 4.50, 4.64, and 4.75 (Table 6). Appendix C.2 complements these ratings with visual-similarity metrics and a preliminary paired mock–real evaluation on 16 tasks, one per environment, in which model ordering is preserved across the two settings and the Qwen3-VL model trained in Section 4.4 leads in both.

5 Conclusion

We introduced IGG to make visibility recovery a measurable component of GUI grounding. Our analyses show that knowing which interaction is required is not enough; current models most often fail in deciding where and how to carry it out. Training on successful interaction trajectories followed by online feedback improves this capability consistently across model families. IGG therefore provides a focused setting for evaluating and learning the interactions that must precede a grounding click.

6 Limitations

IGG uses deterministic mockups for controlled and reproducible evaluation. Our fidelity analyses, including paired mock-real evaluation, suggest that these environments capture relevant visual and interaction patterns, but they do not replace evaluation on live interfaces. Interactive grounding can support accessibility and UI testing, but the same capability may also be misused for automated interaction. We therefore release only controlled mockups that are not connected to external services.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2026-25518317 (50%), Development of AI memory mechanism that reflects human cognitive principles; No. RS-2024-00457882 (50%), National AI Research Lab Project), InnoCORE program of the Ministry of Science and ICT(N10250156), and Samsung Electronics Co., Ltd (Project Code: IO260114-15161-01).

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, and 1 others. 2025a. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Stefano Burigat, Luca Chittaro, and Andrea Vianello. 2012. Dynamic visualization of large numbers of off-screen objects on mobile devices: an experimental comparison of wedge and overview+ detail. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 93–102.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. 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.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277.

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. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. 2023. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *Advances in Neural Information Processing Systems*, volume 36, pages 50742–50768.

Yuan Gao, Kunyu Shi, Pengkai Zhu, Edouard Belval, Oren Nuriel, Srikar Appalaraju, Shabnam Ghadar, Zhuowen Tu, Vijay Mahadevan, and Stefano Soatto. 2024. Enhancing vision-language pre-training with rich supervisions. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13480–13491. IEEE.

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 *The Thirteenth International Conference on Learning Representations*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905.

Shrinidhi Kumbhar, Haofu Liao, Srikar Appalaraju, and Kunwar Yashraj Singh. 2026. Towards gui agents: Vision-language diffusion models for gui grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27493–27502.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng

Chua. 2025. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8778–8786.

Himangi Mittal, Gaurav Mittal, Nelson Daniel Troncoso, and Yu Hu. 2026. See, point, refine: Multi-turn approach to gui grounding with visual feedback. *arXiv preprint arXiv:2604.13019*.

Dean A Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, and 1 others. 2025. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*.

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

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, and 1 others. 2025. Scaling computer-use grounding via user interface decomposition and synthesis. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*.

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.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, and 1 others. 2026. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Aman Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*.

Xiyue Zhu, Peng Tang, Haofu Liao, and Srikar Appalaraju. 2025. Turbocharging web automation: The impact of compressed history states. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3644–3651. Association for Computational Linguistics.

A Mockup Applications

IGG comprises 16 controllable applications spanning common GUI domains. Each environment combines deterministic state transitions and stable target identities with interaction patterns modeled on a familiar interface. Table A1 lists the references, and Figures A1-A5 show representative states.

Table A1: The 16 mockup environments and their reference interfaces.

Mockup	Reference
Social Feed	Twitter/X
Planner	Google Calendar
Developer Hub	GitHub
Code Editor	VS Code Web
Community Chat	Discord
Document Editor	Google Docs
Design Canvas	Figma
Spreadsheet	Google Sheets
Email Client	Gmail
News Portal	Google
Workspace Pages	Notion
Streaming Service	Netflix
E-commerce	Walmart
Music Player	Spotify
Maps	Google Maps
Video Platform	YouTube

B Design Choices

B.1 Viewport robustness

The main benchmark fixes the viewport at 1440×900 so that initial target visibility and constraint type remain constant across models. Responsive layouts can change whether a target is hidden and which interaction reveals it, so the viewport is part of the task specification.

To test whether the learned behavior depends on this layout, we regenerate the 601 tasks at five viewport sizes corresponding to desktop, mobile portrait, tablet, split-screen, and zoomed displays, and keep the 127 tasks that remain valid in every layout (i.e., the target is initially hidden and the reference trace replays successfully). The subset contains 103 training and 24 held-out tasks, so the study measures whether behavior learned at 1440×900 survives a layout change rather than held-out generalization. On this subset, the trained Qwen3-VL model outperforms its base model by 11.1-17.3 points at every viewport, including mobile portrait (Table A2).

Table A2: Grounding accuracy (%) at five viewport sizes.

Viewport	Base	Trained	Gain
1440×900	43.3	59.8	+16.5
390×844	43.3	60.6	+17.3
1024×768	42.5	55.9	+13.4
720×900	48.0	59.1	+11.1
1152×720	39.4	52.8	+13.4

C Additional Results and Analyses

C.1 Per-environment results

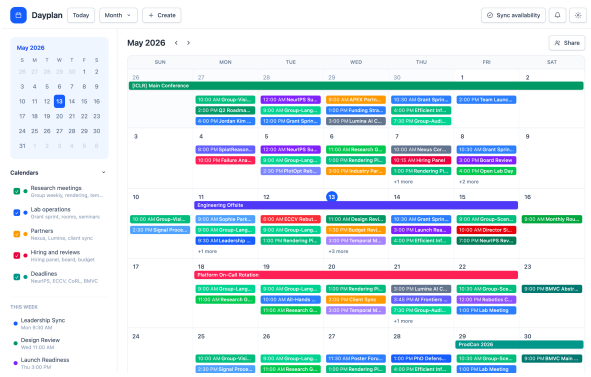
Table A10 breaks the main results down by environment. Open-source models trail GPT-5.5 in every environment. Design Canvas is the hardest environment for open-source models (8.9–13.3% GA) and Video Platform and Email Client the easiest, whereas GPT-5.5 is weakest on Developer Hub, Community Chat, and Social Feed.

C.2 Mockup fidelity

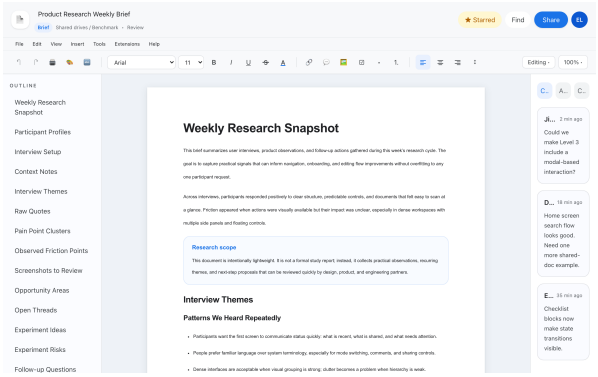
Each task starts from a resettable state, executes deterministic transitions, and uses a stable target ID for click-time evaluation. The environments implement the visibility-changing interactions measured by IGG, including hidden panels, hover reveals, popups, delayed elements, carousels, and movable occlusions. We evaluate their fidelity to real-interface counterparts using visual similarity, human ratings, and paired task execution.

Visual similarity. We measure visual similarity between each mockup and its corresponding real-application reference listed in Table A1. We compare each matched mock-real pair against non-matched pairs formed by pairing the mockup with the other 15 real-application references. We compute feature-space cosine similarities using DINO (Caron et al., 2021) and SigLIP (Zhai et al., 2023), and perceptual distance using DreamSim (Fu et al., 2023). As shown in Table A3, matched pairs are substantially more similar than non-matched pairs, with higher DINO similarity (0.7724 vs. 0.5442), higher SigLIP similarity (0.8094 vs. 0.6211), and lower DreamSim distance (0.2529 vs. 0.5229). These results indicate that the mockups preserve application-level visual structure beyond generic webpage appearance.

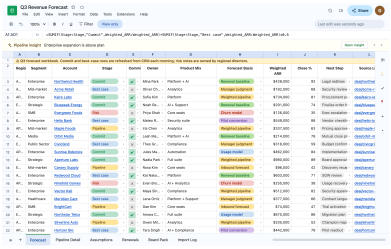
Human fidelity. Participants compare seven environments with their real-interface references and rate visual realism, interaction realism, and task ex-



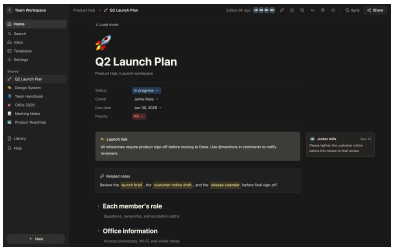
(a) Planner (Google Calendar)



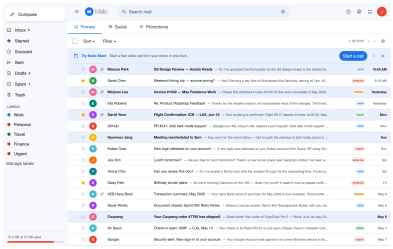
(b) Document Editor (Google Docs)



(c) Spreadsheet (Google Sheets)

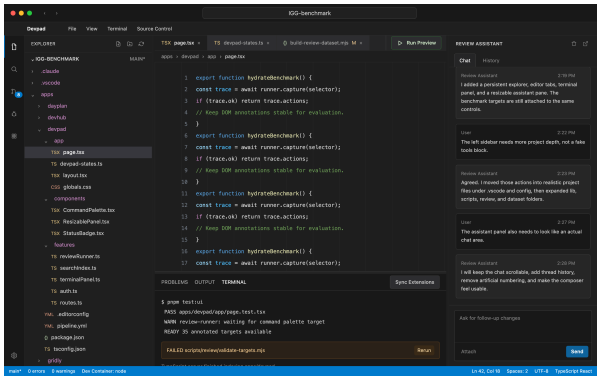


(d) Workspace Pages (Notion)

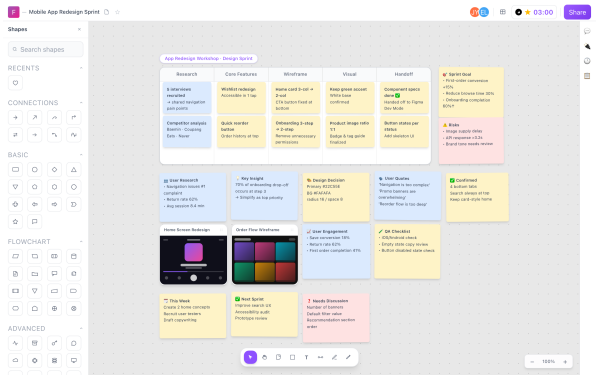


(e) Email Client (Gmail)

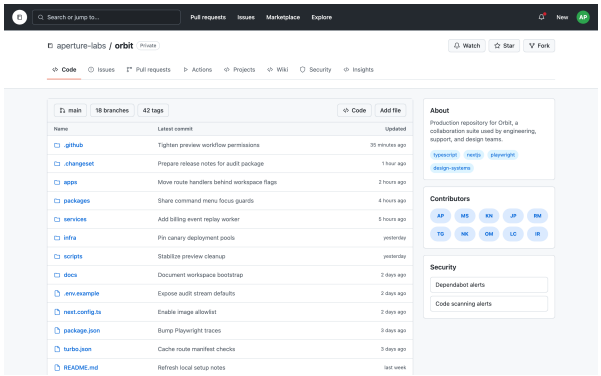
Figure A1: Representative states from productivity mockup applications.



(a) Code Editor (VS Code Web)

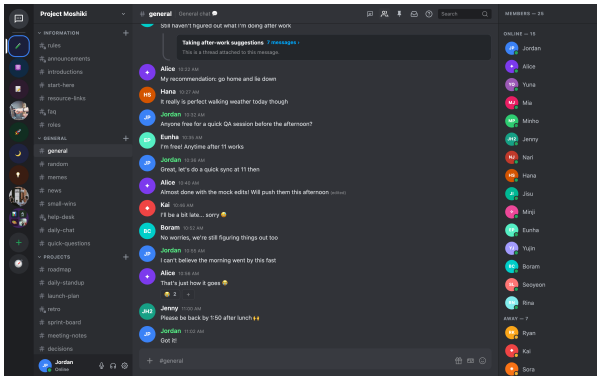


(b) Design Canvas (Figma)

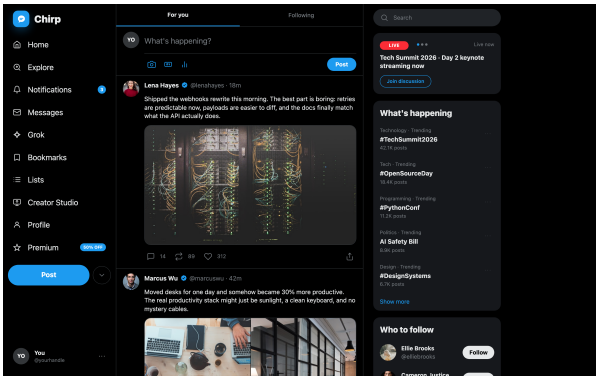


(c) Developer Hub (GitHub)

Figure A2: Representative states from development and design mockup applications.

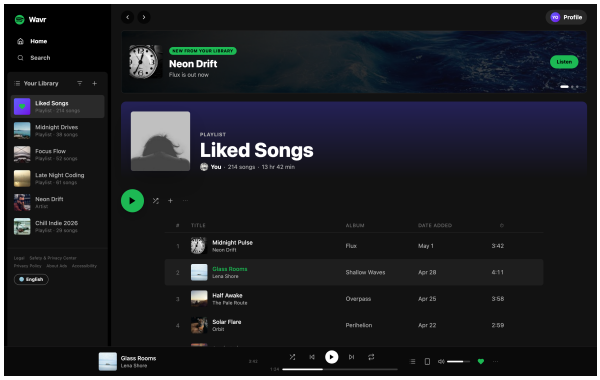


(a) Community Chat (Discord)

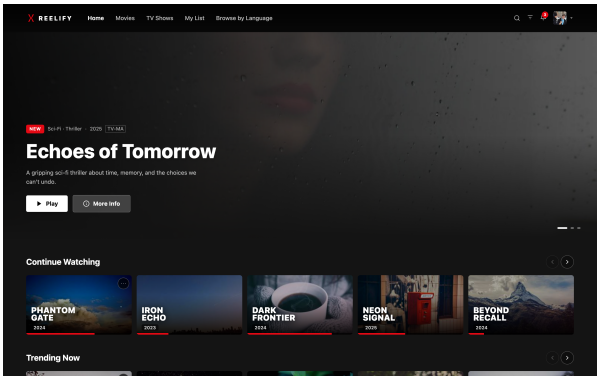


(b) Social Feed (Twitter/X)

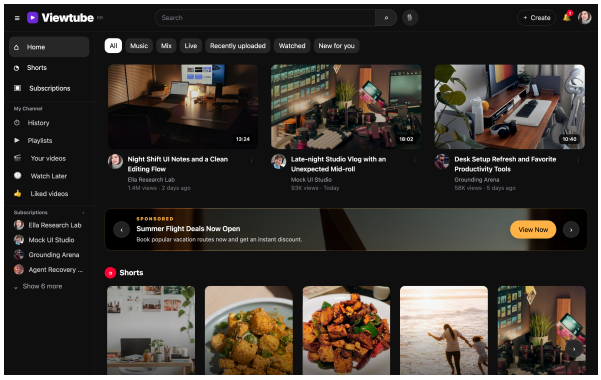
Figure A3: Representative states from social mockup applications.



(a) Music Player (Spotify)

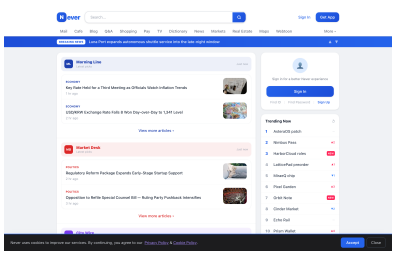


(b) Streaming Service (Netflix)

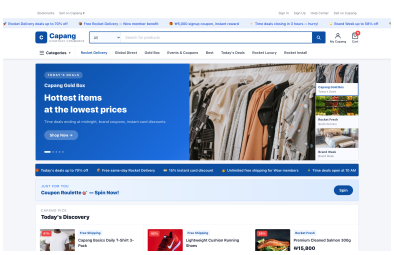


(c) Video Platform (YouTube)

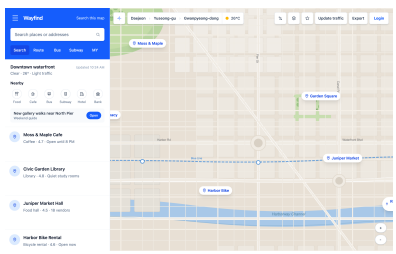
Figure A4: Representative states from media mockup applications.



(a) News Portal (Google)



(b) E-commerce (Walmart)



(c) Maps (Google Maps)

Figure A5: Representative states from lifestyle mockup applications.

Table A3: Visual similarity between mockups and real-application references. DINO and SigLIP are cosine similarities, where higher is better. DreamSim is a perceptual distance, where lower is better.

Pairing	Count	DINO ↑	SigLIP ↑	DreamSim ↓
Matched	16	0.7724	0.8094	0.2529
Non-matched	240	0.5442	0.6211	0.5229

Table A4: Task-level fidelity on 16 paired mock-real tasks. Succ. denotes the number of successful trials out of 16.

Model	Env.	Succ.	GA ↑	AC ↓	RER ↑
Qwen2.5-VL	Mock	3/16	18.8	7.25	0.45
	Real	3/16	18.8	7.69	0.42
Qwen3-VL	Mock	4/16	25.0	7.06	0.54
	Real	5/16	31.3	7.06	0.35
GUI-Owl-1.5	Mock	6/16	37.5	6.13	0.55
	Real	6/16	37.5	6.88	0.57
Qwen3 SFT+GRPO	Mock	9/16	56.3	5.06	0.72
	Real	9/16	56.3	5.00	0.70

ecutability. The three mean ratings are 4.50, 4.64, and 4.75 out of 5 (Table 6).

Task-level mock-real fidelity. We run four models on 16 paired mock and real-interface tasks. Model ordering is preserved: Qwen3-VL SFT+GRPO leads in both settings with 56.3% GA, followed by GUI-Owl-1.5, Qwen3-VL, and Qwen2.5-VL (Table A4). Across the four model families, mock and real GA are strongly consistent (Pearson $r = 0.98$; Spearman $\rho = 1.00$). The paired set contains one task per application. Figures A6 and A7 show representative outcomes.

C.3 Instruction design

Instructions name the target but not the reveal procedure. Annotators avoid action hints such as "scroll down," "hover over the icon," or "wait until the button appears," and add a natural cue, such as an author, document title, or surrounding content, when several elements share a label. For example, "Scroll down and click Kai" becomes "Find Kai in the direct message list," and "Drag the sticky note away and click QA Checklist" becomes "Click the QA Checklist note without deleting any notes." Multimodal LLMs propose candidate instructions and reference traces, and annotators correct the wording, repair traces where needed, and replay every task to verify that the target is unique and reachable.

Effect of procedural hints. As a diagnostic, we also evaluate instructions that state the intermediate procedure. Unlike the *reveal action given* condition of Section 4.3, which names only the action type, these hints describe the procedure. With such hints, Qwen3-VL rises from 30.8% to 56.9% GA over all 601 tasks, but hints do not solve precise manipulation: hinted Qwen3-VL solves 0 of 8 held-out movable-occlusion tasks, whereas SFT+GRPO solves 5 of 8 without any test-time hint. The main benchmark therefore retains goal-only instructions, and the training gains reflect an executable interaction prior rather than instruction following.

D Evaluation Protocol

The open-source baselines are `Qwen/Qwen2.5-VL-7B-Instruct`, `Qwen/Qwen3-VL-8B-Instruct`, and `mPLUG/GUI-Owl-1.5-8B-Instruct`, and the commercial baseline is `gpt-5.5`. Each model is evaluated once per task with greedy decoding from a reset environment.

At each turn, the agent receives the system prompt, task instruction, screenshot, step index, coordinate space, and action history. It returns one JSON action. Figure A8 shows the prompt, and the shared action schema is:

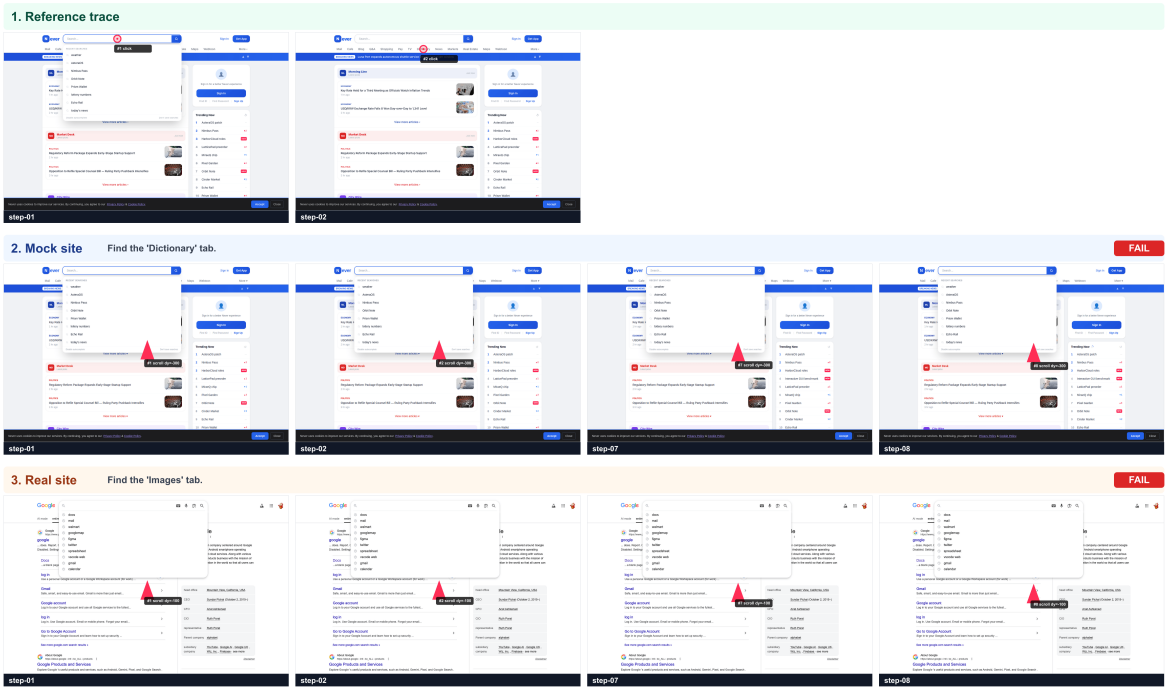
```
{"type": "scroll", "x": 720, "y": 480, "dy": 50}
{"type": "move", "x": 512, "y": 340}
{"type": "drag", "startX": 10, "startY": 20, "
    endX": 80, "endY": 100}
{"type": "wait", "ms": 1500}
{"type": "click", "x": 512, "y": 340}
```

A `click` succeeds only when the clicked element ID matches the annotated target ID. Otherwise the environment applies any defined state transition and returns the next screenshot. Navigation outside the task environment is disabled, and the episode ends on success or budget exhaustion. The `drag` schema (`startX`, `startY`, `endX`, `endY`) is identical in the prompt, the evaluator, and Table 1.

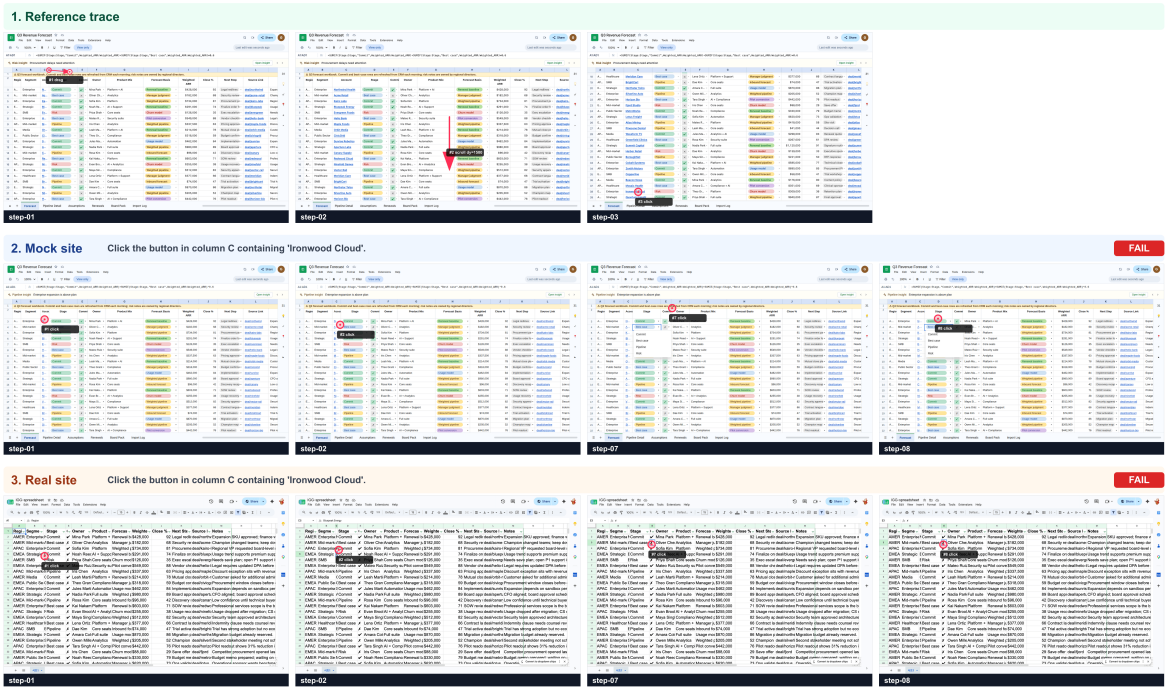
E Dataset Examples

E.1 Task specification format

Each task record contains a goal-only instruction, a target ID, an initial state, and one successful reference trace (Listing 1). The evaluator keeps the target ID hidden from the model and uses the trace only to compute interaction-efficiency metrics. Reference traces may refer to elements by stable selectors, which the replay tool resolves to the center

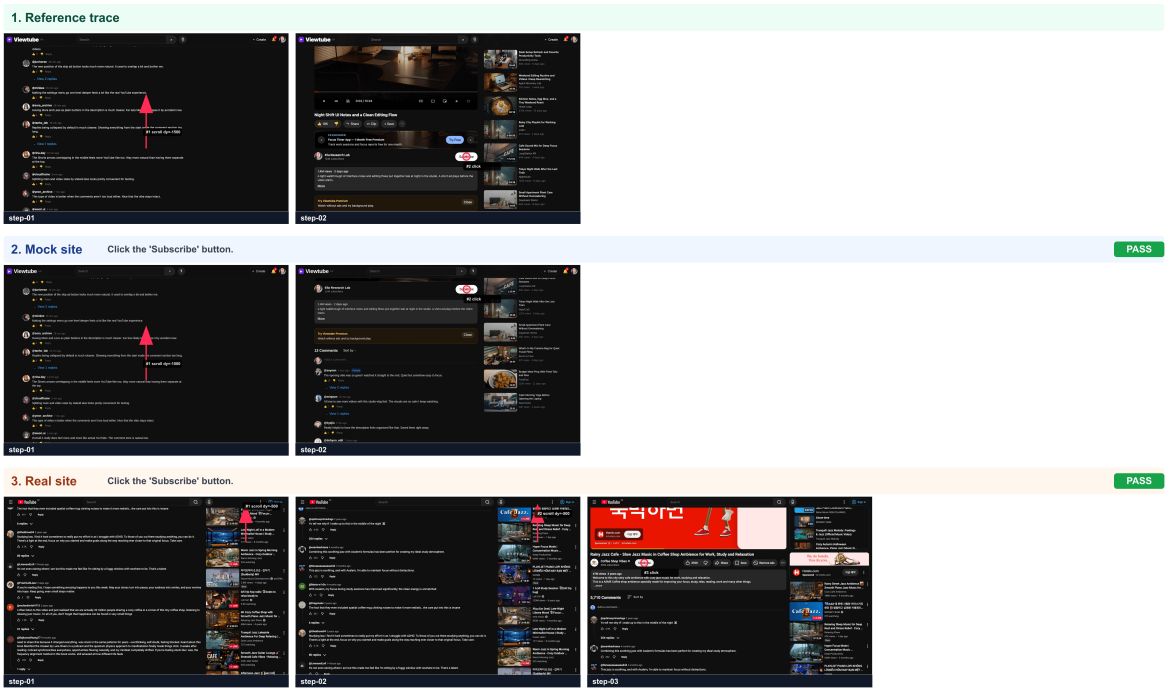


(a) Qwen2.5-VL

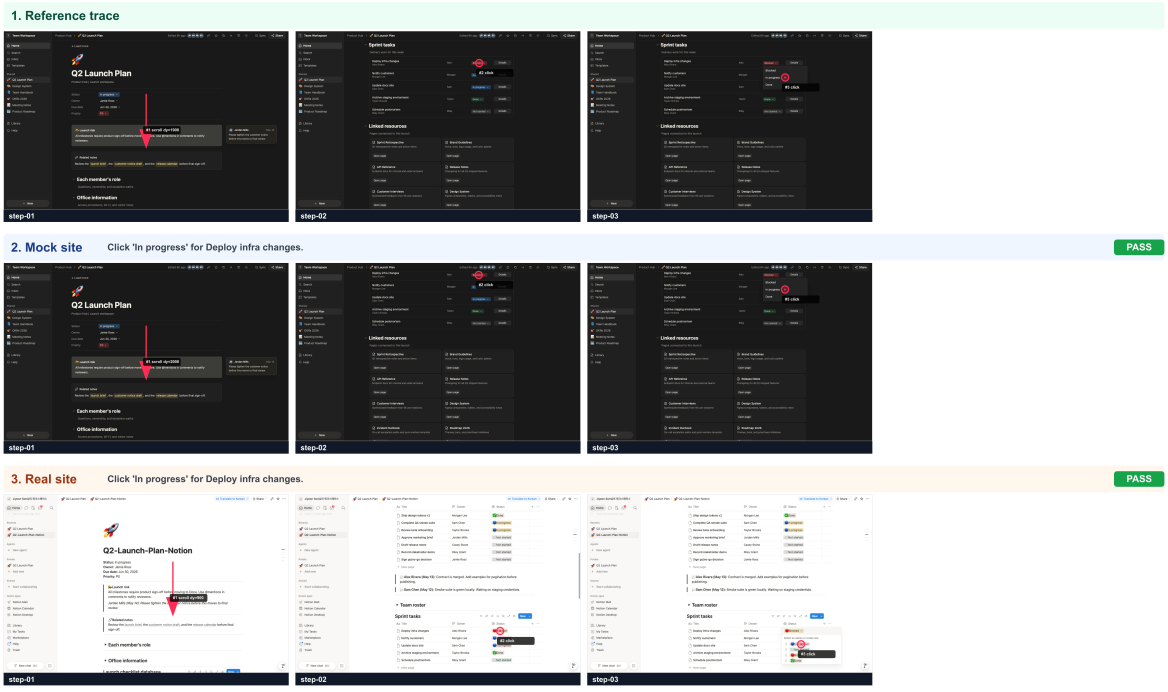


(b) Qwen3-VL

Figure A6: Paired examples in which the model fails in both the controlled and real-interface task.



(a) GUI-Owl-1.5



(b) Qwen3-VL SFT+GRPO

Figure A7: Paired examples in which the model succeeds in both the controlled and real-interface task.

```
You are an Interactive GUI Grounding agent.

Your ONLY goal is to locate the single UI element described in the instruction and click on it. The element may NOT be
visible in the current screenshot. In that case, take the minimum interaction needed to reveal it (scroll, hover, expand,
wait, drag away an occluder, etc.) and then click the element.

This is a grounding task, not a navigation task. Do NOT try to complete any higher-level workflow; just find the target
and click it.
  ...

Action space (emit exactly ONE action per turn):
  - Scroll a region: {"type": "scroll", "x": <int>, "y": <int>, "dy": <int>}
    dy > 0 scrolls down, dy < 0 scrolls up. Use (x, y) to pick which
    scroll container gets the event (e.g. a left sidebar vs the main feed).
  - Move the cursor: {"type": "move", "x": <int>, "y": <int>}
    Triggers hover effects, tooltips, and hover-reveal menus.
  - Drag: {"type": "drag", "startX": <int>, "startY": <int>, "endX": <int>, "endY": <int>}
    Use to nudge an occluding popup aside without fully closing it, or to move sliders/panels.
  - Wait: {"type": "wait", "ms": <int>}
    Use when the target is gated by a timer, async load, ad countdown, auto-rotating carousel, etc.
  - Click: {"type": "click", "x": <int>, "y": <int>}
    Grounding trigger. If (x, y) lands on the target element, the task ends immediately with success.
    If it lands on anything else, the task continues and the screen state may change.

Strategy:
  1. Read the instruction. Identify a unique visual description of the target.
  2. Scan the screenshot. If the target is visible, click its center.
  3. If not visible, reason about why: is it offscreen, in a collapsed panel,
    hidden behind a popup, revealed only on hover, or not loaded yet?
  4. Pick the single cheapest action that could make it visible or confirm it.
  5. Avoid exploratory clicks on random UI elements; each click is evaluated.

Output format (STRICT):
  Reply with exactly ONE line containing a single JSON object for the next action.
  No prose, no markdown, no code fences, no explanation.
  ALL coordinates are SEPARATE integer keys, NEVER a list and NEVER a nested object.
  ...
```

Figure A8: Excerpt of the system prompt used for IGG model evaluation. Ellipses indicate omitted formatting examples.

of the element's current bounding box, whereas models must emit screenshot coordinates.

Listing 1: Task specification example. The task targets "Kai" in the direct message list (Community Chat application, $CT_{1\text{-}1}$: Scroll-based Exploration). The ground-truth trajectory consists of a scroll action followed by a click on the revealed target.

```
{
  "id": "messaging-01-find-kai-direct-message-
      list",
  "sub_type": "1-1",
  "instruction": "Find 'Kai' in the direct
      message list.",
  "target_id": "messaging.dm.kai",
  "initial_state_id": "1",
  "ground-truth trajectory": [
    {"action": "SCROLL", "dy": 1000},
    {"action": "CLICK", "selector": "messaging.dm
        .kai"}
  ]
}
```

E.2 Sub-type examples

Figures A9–A15 show one representative task for each IGG sub-type. Each example pairs the initial state, where the target is not directly actionable, with the final state at the time of the successful target click.

F Failure Case Analysis

F.1 Representative cases

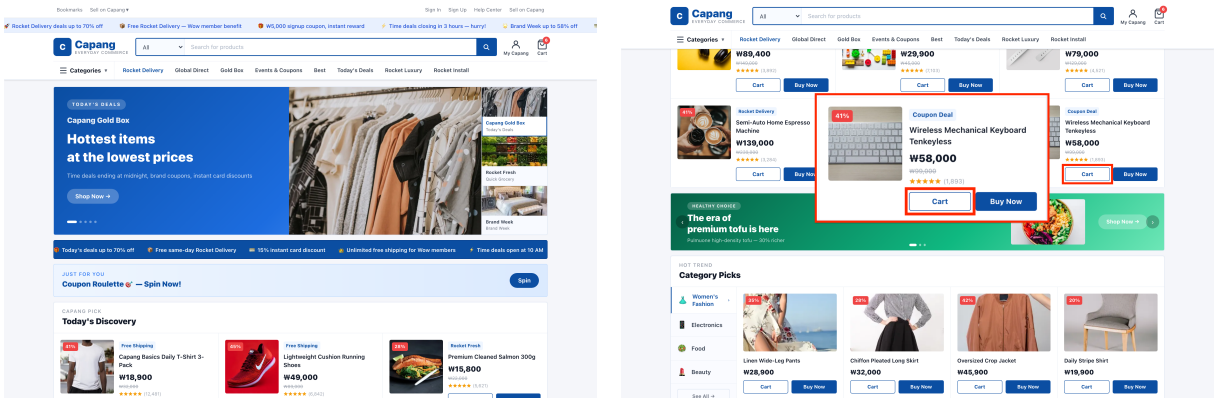
Across Qwen2.5-VL, Qwen3-VL, and GUI-Owl-1.5, shared failures often omit the action that would reveal the target. Table A5 summarizes three patterns, and Figure A16 shows representative traces.

F.2 Trace-derived action-prior diagnostic

The representative failure cases in Section F.1 show agents spending the budget without the needed reveal action. For each failed Qwen3-VL trajectory, we compare the model actions with the ground-truth trajectory and record whether each required reveal action appears at least once.

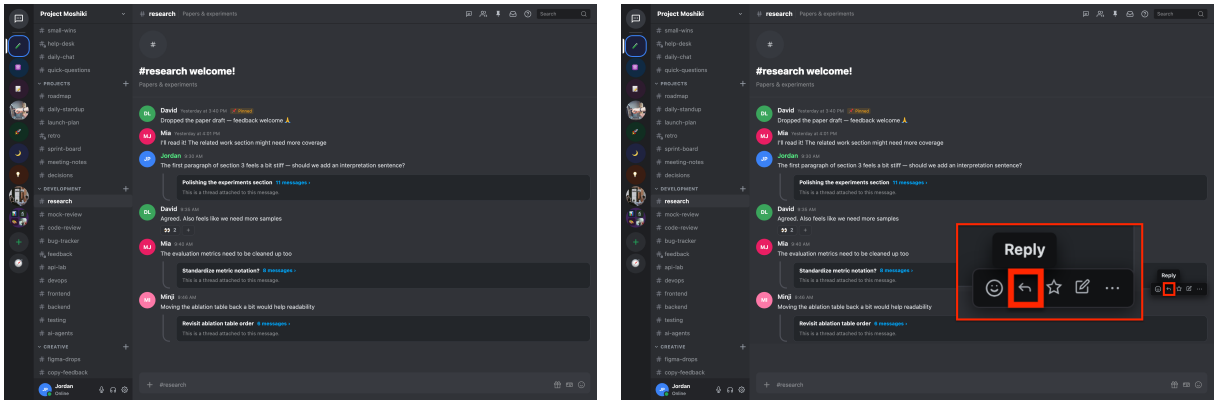
F.3 Target-agnostic reveal heuristics

To separate reveal patterns that a fixed rule can solve from those that require target-specific decisions, we evaluate five target-agnostic heuristics, each on the sub-type it is designed for: scrolling the page center in fixed steps, hovering over common icon regions in order, clicking common dismiss locations, waiting in one-second steps, and dragging visible occluders toward the screen edge. A heuristic succeeds when the target becomes visible,



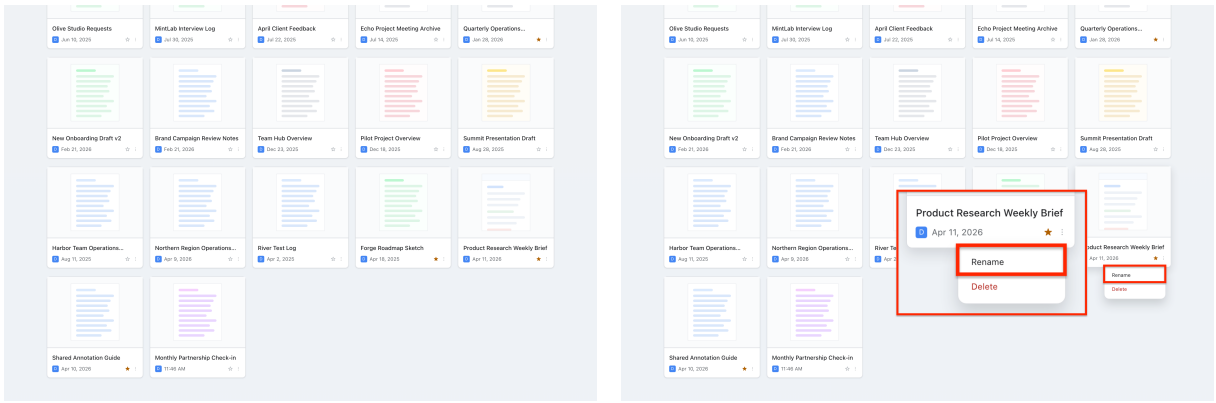
(a) Initial state (b) Final state at click time

Figure A9: **$CT_{1\text{-}1}$: Scroll-based Exploration** (E-commerce). *"Click 'Cart' for 'Wireless Mechanical Keyboard'."*



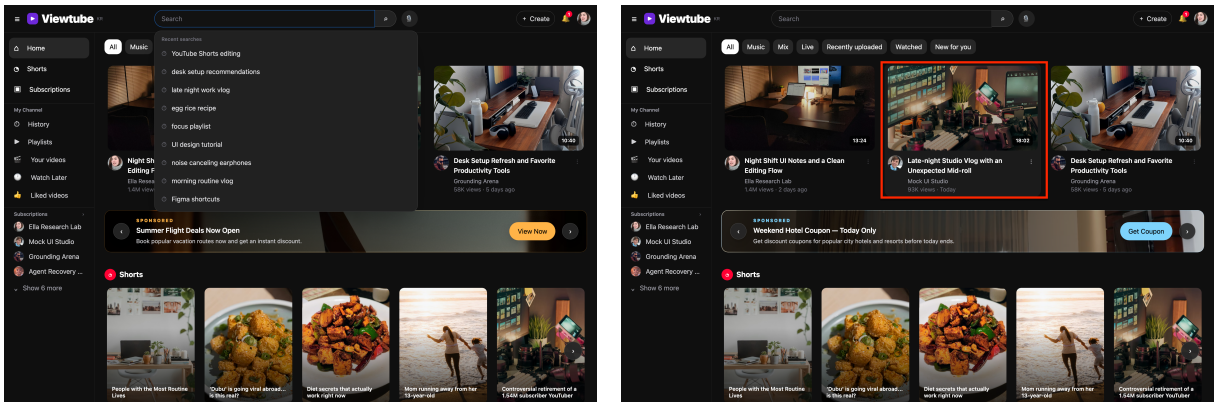
(a) Initial state (b) Final state at click time

Figure A10: **$CT_{2\text{-}1}$: Visual Disambiguation** (Community Chat). *"Click 'Reply' on Minji's message."*



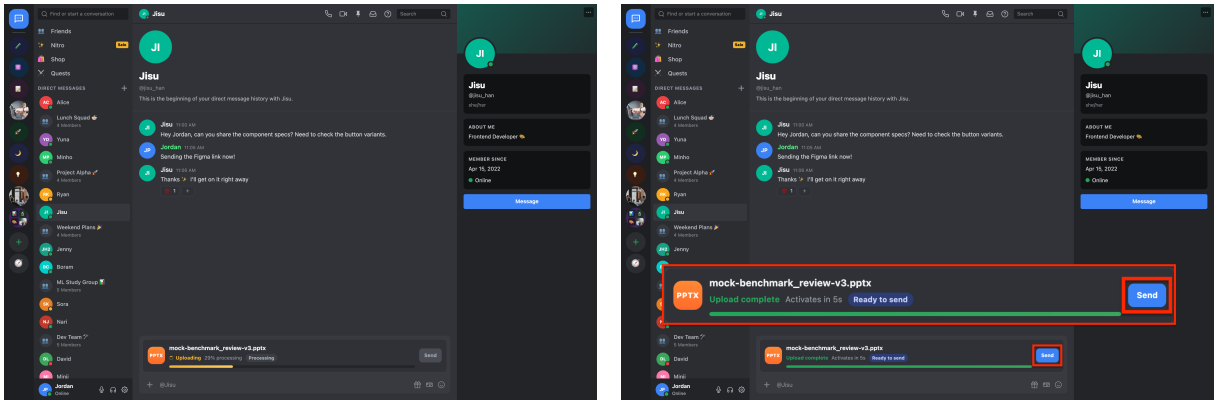
(a) Initial state (b) Final state at click time

Figure A11: **$CT_{2\text{-}2}$: Hidden State Activation** (Document Editor). *"Click 'Rename' on 'Product Research Weekly Brief'."*



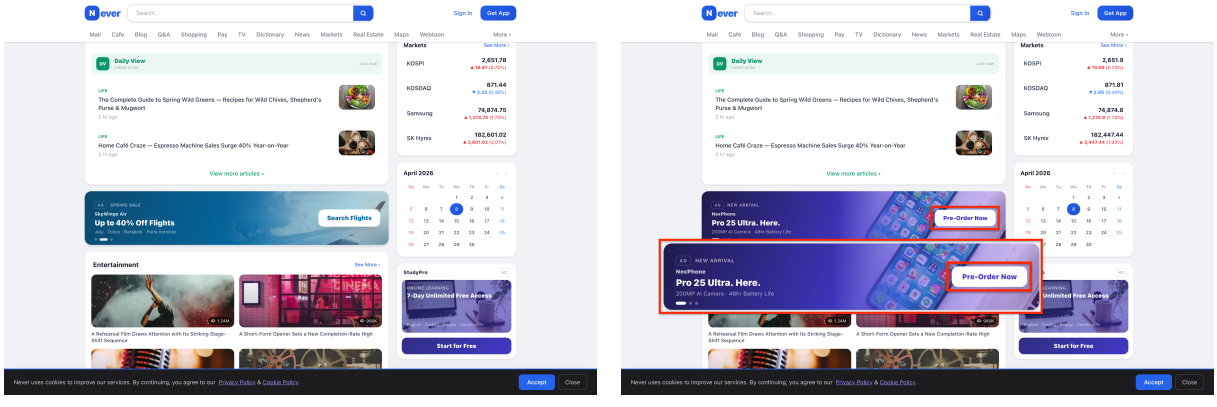
(a) Initial state (b) Final state at click time

Figure A12: **$CT_{2\text{-}3}$: Removable Occlusion** (Video Platform). *"Click the 'Late-night Studio Vlog' video card."*



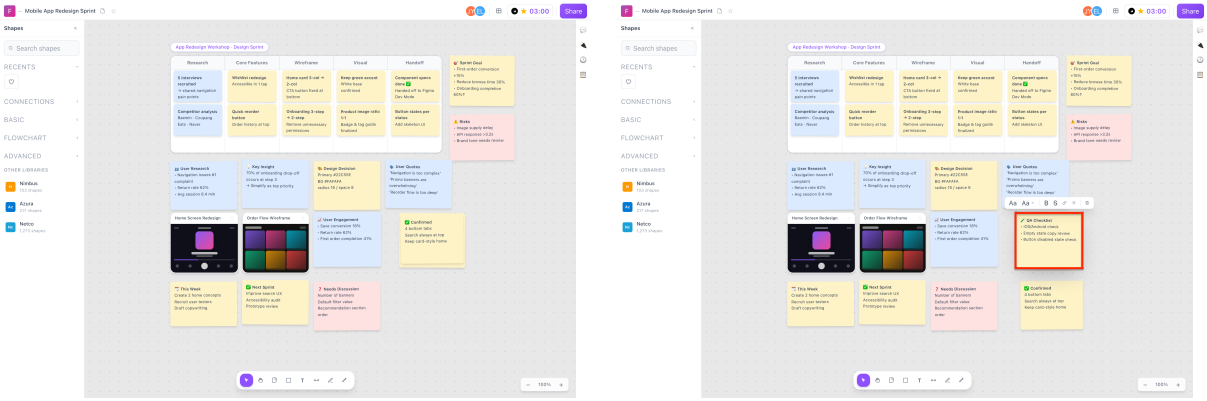
(a) Initial state (b) Final state at click time

Figure A13: **$CT_{3\text{-}1}$: Delayed State Activation** (Community Chat). *"Click 'Send' for the pptx upload."*



(a) Initial state (b) Final state at click time

Figure A14: **$CT_{3\text{-}2}$: Dynamic Target Interaction** (News Portal). *"Click 'Pre-Order Now' in the 'NexPhone' ad."*



(a) Initial state (b) Final state at click time

Figure A15: **$CT_{3\text{-}3}$: Movable Occlusion** (Design Canvas). *"Click the 'QA Checklist' note without deleting any."*

Table A5: Representative failure cases selected from tasks failed by Qwen2.5-VL-7B, Qwen3-VL-8B, and GUI-Owl-1.5-8B.

Failure mode	**Sub-type**	**Task**	**Observed trace pattern**
Spatial manipulation is not attempted	$CT_{3\text{-}3}$: Movable Occlusion	Design Canvas: find the *QA Checklist* note without changing zoom.	All three models exhausted the budget without using a drag operation. Qwen2.5-VL-7B and Qwen3-VL-8B produced eight scrolls; GUI-Owl-1.5-8B produced one wait followed by seven scrolls.
Dynamic target state is not tracked	$CT_{3\text{-}2}$: Dynamic Target Interaction	News Portal: find the *Glass Harbor begins a 24-hour pilot* headline in Breaking News.	The models interacted with the wrong state of the ticker. Qwen3-VL-8B clicked the ticker controls and ended on a different headline, while the other models scrolled or expanded unrelated page content.
Hover-based ambiguity is not resolved	$CT_{2\text{-}1}$: Visual Disambiguation	Community Chat: find the *Creator Dock* server.	The target server is identifiable only after hovering over ambiguous server icons. Qwen2.5-VL-7B repeatedly scrolled, while Qwen3-VL-8B and GUI-Owl-1.5-8B repeatedly clicked icons without first revealing their labels.

Table A6: Trace-derived action-prior diagnostic for Qwen3-VL failed trajectories. Counts are non-exclusive because one task can require multiple setup actions.

Required setup action	**Failed tasks**	**Omitted**	**Rate**	**Diagnostic interpretation**
DRAG	59	59	100.0%	Drag is absent from failed traces.
SCROLL	247	207	83.8%	Scroll is often replaced by clicks or local moves.
MOVE	96	71	74.0%	Hover and disambiguation moves are often skipped.
WAIT	3	1	33.3%	Few failed tasks require explicit waiting.

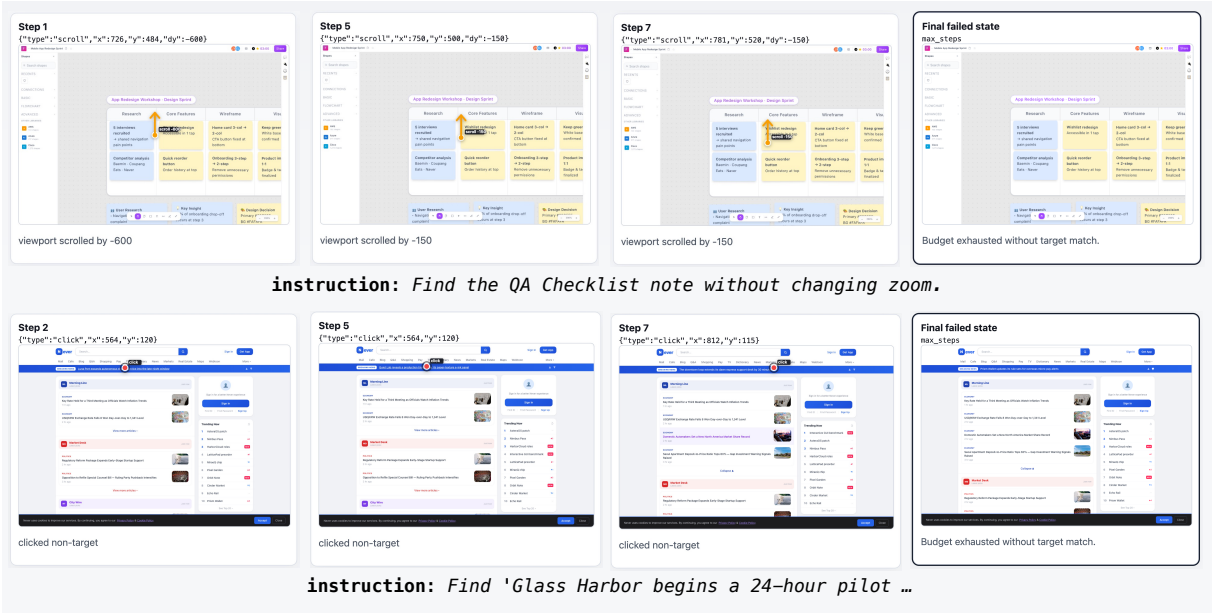


Figure A16: **Representative failed traces.** The figure visualizes example failed traces from two representative cases, showing how models spend the interaction budget without performing the visibility-changing action needed to reveal the target.

unobstructed, and actionable (Table A7).

Fixed rules often resolve scrolling, hovering, and waiting, but rarely resolve removable or movable occlusions. This agrees with the trace analysis in Section F.2, where DRAG is absent from every failed trajectory that requires it.

G Training Details

Our two-stage procedure uses SFT to introduce rare reveal actions and online GRPO to optimize their execution from browser feedback.

G.1 Training Setup

Two tasks whose ground-truth trajectories could not be replayed reliably during corpus construction are excluded before splitting the remaining 599 tasks into 479 training tasks and 120 held-out validation tasks. Every included ground-truth trajectory is human-corrected and verified by replay; validation tasks contribute neither SFT examples nor GRPO rollouts.

G.2 SFT

The ground-truth trajectories yield 1,141 action-level examples with normalized coordinates. We use LoRA rank 8, alpha 32, dropout 0.05, all-linear language-model target modules, a frozen vision tower and projector, bf16, learning rate 1×10^{-4}, effective batch size 8, maximum length 8192, maximum image pixels 1,382,400, and 500 steps with checkpoints every 100 steps. Frequency balancing oversamples CLICK/SCROLL/MOVE/WAIT/DRAG by 1/1/2/2/4, expanding the data to 1,460 examples and the drag examples from 55 to 220.

G.3 Online GRPO

GRPO starts from an SFT LoRA seed and collects online browser rollouts from the 479 training tasks under the same eight-action budget used in evaluation. We use group size 6, temperature 1.0, top-p 0.95 for rollout sampling, learning rate 2×10^{-6}, PPO-style clipping with $\epsilon = 0.2$, maximum gradient norm 1.0, and up to 20 update groups. We evaluate saved LoRA checkpoints with greedy decoding on the fixed 120-task validation split.

Online interaction reward. The reward is computed from browser-executed traces. Let S denote target-click success; T the action count; W non-empty wrong clicks; E empty clicks; I invalid actions; A repeated ineffective actions; and R repeated wrong-click locations. The bounded bonus B rewards reveal progress (Table A9). Successful traces with prior wrong clicks are capped at 1.2, shortcut successes at 0.8, and runtime or parsing failures are excluded from policy updates.

Table A7: Reveal success of target-agnostic interaction rules on their corresponding constraint sub-types.

Constraint sub-type	Interaction rule	Reveal success (%)
$CT_{1\text{-}1}$: Scroll-based Exploration	Scroll center by fixed steps	68.8
$CT_{2\text{-}1}$: Visual Disambiguation	Hover common icon areas	61.4
$CT_{2\text{-}3}$: Removable Occlusion	Click common dismiss areas	17.6
$CT_{3\text{-}1}$: Delayed State Activation	Wait in 1-s steps	87.1
$CT_{3\text{-}3}$: Movable Occlusion	Drag visible occluder aside	16.9

G.4 Cross-architecture results

We apply the same recipe, with the same data, budget, and hyperparameters, to InternVL3.5-8B and GUI-Owl-1.5-8B. All three architectures improve by a similar margin: Qwen3-VL from 40.0% to 59.2%, InternVL3.5 from 28.3% to 47.5%, and GUI-Owl-1.5 from 35.0% to 52.5% (Table A8). The models differ in coordinate tokenization and pretraining, so the gain is not specific to one model family.

Table A8: Grounding accuracy before and after SFT+GRPO for three model architectures.

Model	Base	SFT+GRPO
Qwen3-VL-8B	40.0	59.2
InternVL3.5-8B	28.3	47.5
GUI-Owl-1.5-8B	35.0	52.5

H Human Study Details

H.1 Study setup

This appendix provides implementation details for the two human studies summarized in Section 4.5. Fifteen participants completed two browser-based protocols connected to the same environments as model evaluation. In the solvability study, every participant completed a subtype-balanced suite of 21 tasks—three from each of the seven sub-types and at least one from every environment—yielding 315 trials. In the fidelity study, participants compared seven controlled environments with real-interface references and rated visual realism, task executability, and interaction realism on five-point scales.

H.2 Task-solvability interface

Participants receive one instruction and interact directly with the browser. The interface ends a trial when the participant clicks the target or reaches eight click or drag attempts. It records scroll sessions after an 800-ms debounce for interaction counts and records hover events without applying them to the attempt limit (Figure A17).

H.3 Fidelity-test interface

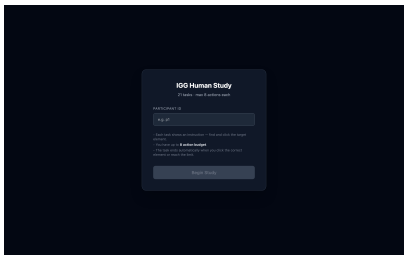
For each paired example, participants see the instruction, controlled environment, and real-interface reference, then rate visual realism, task executability, and interaction realism (Figure A18).

Table A9: GRPO reward components used in the diagnostic training experiments. Positive progress bonuses are bounded per rollout and are applied only when the browser state or target evidence changes in the intended direction.

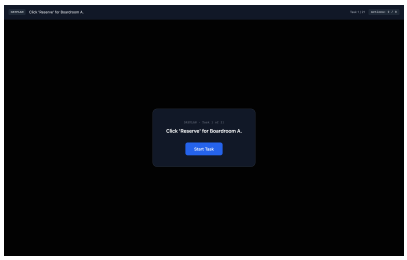
Component	**Symbol**	**Value / rule**
Target-click success	S	$+2.2$ if successful
Unsuccessful rollout	$\neg S$	-1.0 base term
Step cost	T	-0.03 per action
Wrong non-empty click	W	-0.65 per event
Empty click	E	-1.00 per event
Invalid action	I	-1.00 per event
Repeated ineffective action	A	-0.20 per event
Repeated wrong-click location	R	-0.25 per event
Hover / move progress	B	$+0.45$
Hidden-state progress	B	$+0.45$
Drag progress	B	$+0.75$; $0.7\times$ if no state change
Wait progress	B	$+0.35$; $0.5\times$ if no delayed change
Clean visible-target click	B	$+0.20$
Reward clipping	–	$[-4.0,\ 2.8]$

Table A10: Per-application GA, AC, RER, and TT for Qwen3-VL and GPT-5.5 on all 601 tasks.

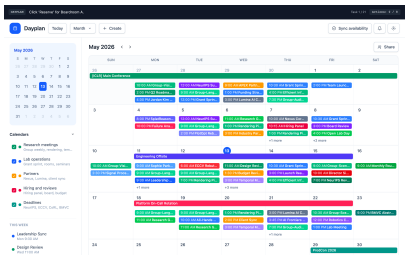
	Qwen3-VL				**GPT-5.5**			
App	**GA**	**AC**	**RER**	**TT**	**GA**	**AC**	**RER**	**TT**
Social Feed	29.4	6.47	0.93	17514	61.8	4.59	1.04	69757
Planner	21.2	6.88	0.79	13098	81.8	3.88	0.83	61011
Developer Hub	25.7	6.94	0.66	9763	57.1	5.57	0.69	138919
Code Editor	21.9	6.81	1.26	9575	87.5	3.88	0.97	59747
Community Chat	26.5	6.82	0.64	9357	61.2	4.82	1.03	108102
Document Editor	37.5	6.05	0.99	8451	85.0	3.48	1.04	63538
Design Canvas	8.9	7.44	1.25	13951	82.2	4.33	0.94	93199
Spreadsheet	33.3	6.24	0.84	8920	83.3	2.93	1.04	124023
Email Client	50.0	5.38	0.75	10493	88.1	3.38	0.97	63120
News Portal	29.0	6.55	0.80	19204	71.0	4.77	0.80	52984
Workspace Pages	34.3	6.40	0.86	8830	85.7	3.71	0.93	30489
Streaming Service	20.5	7.00	0.67	17224	87.2	4.05	0.71	52801
E-commerce	43.3	6.40	0.60	17047	86.7	4.40	0.72	49700
Music Player	36.0	5.80	1.15	12580	86.0	3.60	1.04	63597
Maps	26.7	6.47	1.25	11972	70.0	4.53	0.95	166930
Video Platform	50.0	5.91	0.63	8158	94.1	3.15	0.88	38354



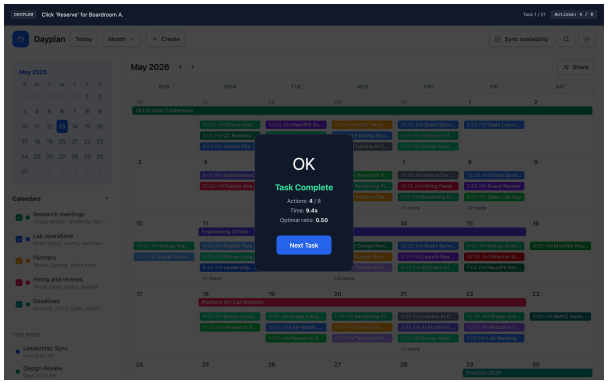
(a) Start page



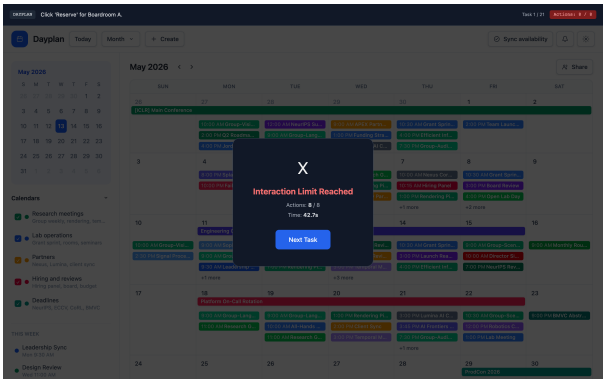
(b) Task instruction



(c) Task in progress

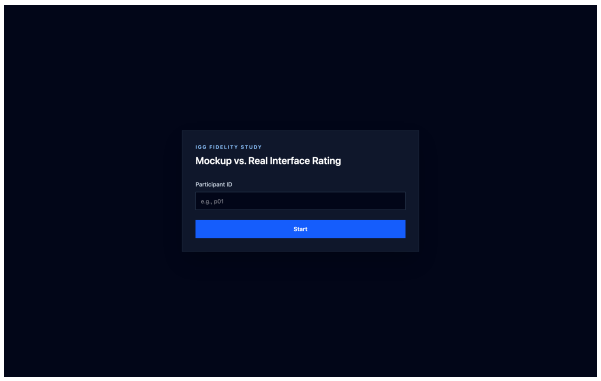


(d) Task complete (success)

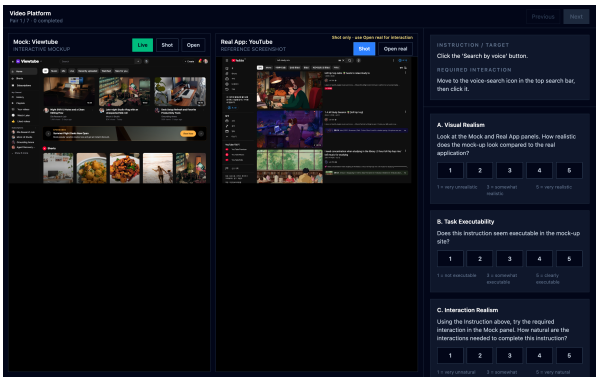


(e) Interaction limit reached (failure)

Figure A17: The web-based interface used in the human study. (a) Participants begin by entering an ID and reviewing the task format. (b) Each task is introduced with the instruction text. (c) Participants then interact with the live mockup; the interaction counter is shown in the top-right corner. (d) The task ends with success when the participant clicks the correct target, or (e) with failure when the eight click-or-drag attempt limit is reached.



(a) Start page



(b) Rating interface

Figure A18: The web-based interface used in the human fidelity test. (a) Participants enter the fidelity-study session. (b) For each application, participants compare the mock interface with the real-application reference and rate visual realism, task executability, and interaction realism.