🎯 REFRAMING GUI GROUNDING AS INTENT-LEVEL OBJECT DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Currently, Vision-Language Model (VLM)-based GUI agents formulate grounding as coordinate prediction, forcing the model to learn a mapping from spatial locations to discrete coordinate tokens without explicit spatial supervision. To address this limitation, we revisit GUI grounding from a spatial detection perspective and find that existing object detectors propose GUI-element-like regions but rarely rank the intended one first, i.e., they lack high-level intent understanding. We therefore propose **INTENTLOC**, an intent-level object detection model that uses high-level semantic queries to equip a detector with intent understanding ability and introduces **Crosshair Attention** to progressively locate small GUI targets. Under the same backbone and training data, INTENTLOC-GUI outperforms pure coordinate prediction baselines by 10.6 points, averaged with equal weight over four GUI grounding benchmarks and two model sizes. We further introduce **Atypical-GUI-Bench**, a manually curated benchmark for evaluating the robustness and generalization of GUI grounding models under atypical in-the-wild visual observations. Additionally, when integrated into an agentic framework with a VLM planner, our approach improves success on end-to-end GUI tasks. Code, all data (training data and benchmark), and models will be publicly available.

1 INTRODUCTION

GUI agents are intelligent systems that autonomously interact with graphical user interfaces, performing human-like operations to accomplish various tasks on electronic devices. At the core of GUI agents lies *GUI grounding*, which maps a screenshot and a natural language instruction to precise interaction locations. Current VLM-based GUI agents formulate grounding as a coordinate prediction problem by directly outputting numerical (x, y) values, introducing three fundamental deficiencies: (1) the model is required to implicitly learn a mapping from 2D spatial locations to discrete coordinate tokens, which is inherently difficult and error-prone; (2) directly supervised fine-tuning VLMs on numerical tokens without explicit spatial supervision leads to optimization difficulty and inefficient learning (Jiang et al., 2026); (3) the predicted coordinates are tied to the input resolution, so the same target maps to different token sequences as the screenshot is rescaled. To address these deficiencies, we explore an alternative approach to GUI grounding, where the model identifies the target region in the image instead of directly predicting coordinates.

This raises natural questions: Can GUI grounding be viewed as a form of visual object detection and localization? And how can GUI grounding leverage object detection models' strong spatial localization capability?

To answer these questions, we systematically investigate the performance and potential of GroundingDINO (Liu et al., 2024a), the most widely-adopted open-vocabulary object detector, for GUI grounding (Section 3). Fine-tuning it on GUI data shows that GroundingDINO can effectively localize a broad set of interactive elements, indicating that object detection models have the potential to serve as a GUI grounding model. However, semantic ranking and understanding of user intent remain limited—the model often fails to assign the highest confidence to the intended element.

To address the conventional object detection model's limitations revealed by this pre-analysis, we propose **Intent-Level Object Detection (ILOD)**, a detection-based paradigm of solving the grounding problem. ILOD localizes the visual target implied by a high-level user intent rather than matching an explicitly named object category. In conventional object detection, the query specifies what

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

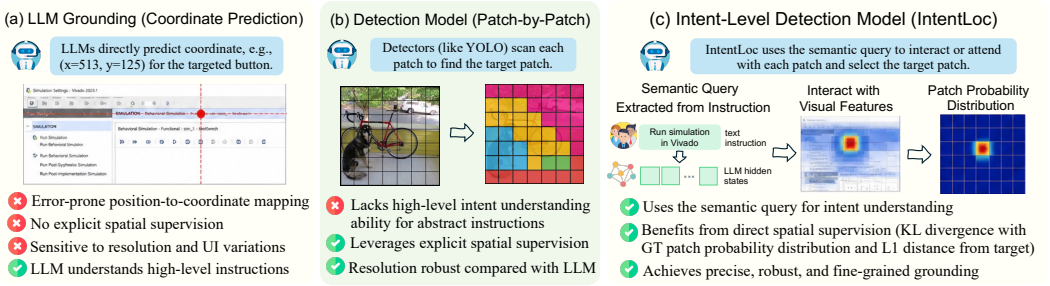


Figure 1: **(a)** LLM-based coordinate prediction method suffers from error-prone position-to-coordinate mapping and other limitations, while **(b)** detection models provide strong spatial modeling but lack high-level intent understanding. **(c)** INTENTLOC bridges the two paradigms by using semantic query to interact with visual features, enabling precise and robust grounding.

the object is (e.g., "a running girl"); ILOD instead takes a query that specifies what the user intends to do (e.g., "run simulation in the Vivado software", "grasp the mug on the table"). The ILOD paradigm applies in many concrete settings: GUI grounding, embodied pointing (Yuan et al., 2026b), and instruction-driven visual search in natural scenes (Wu & Xie, 2024).

Under the ILOD paradigm, we present a new model architecture, **INTENTLOC**. In our INTENTLOC model, we leverage semantic features from the VLM to enable the detector to understand user intent. Figure 1 compares the localization mechanisms of LLM-based methods, classic detection models, and our INTENTLOC, highlighting their respective strengths and limitations. We also introduce **Crosshair Attention**, which predicts a geometrically supervised 2D reference point at each decoder layer and progressively focuses decoder attention on the relevant local region. Crosshair Attention is designed to suppress distant distractors and facilitate localization when a small GUI target occupies only a few visual patches. We present the details of our method in Section 4.

We evaluate the grounding capability of our INTENTLOC-GUI model on four GUI grounding benchmarks, where it achieves the best performance on all four benchmarks among Qwen2.5-VL-based models of comparable scale. We further introduce **Atypical-GUI-Bench**, a benchmark for evaluating the robustness and generalization of GUI grounding models under atypical in-the-wild visual observations, where INTENTLOC-GUI-3B outperforms finetuned Qwen2.5-VL-3B and GUI-Actor-3B (Wu et al., 2025a) by 16.9 and 12.4 points, respectively. In addition, we develop an agentic framework that combines state-of-the-art proprietary VLMs as the planner and our INTENTLOC-GUI model as the executor, which achieves better performance than competing grounding models and end-to-end GUI agent models. We also validate INTENTLOC on natural image and embodied pointing task in **Appendix A**, where it also achieves strong performance.

Overall, our contributions can be summarized as follows: **(1)** We present a systematic study of why GroundingDINO fails on GUI grounding, concluding that it has spatial localization potential but lacks intent understanding. We therefore propose **Intent-Level Object Detection (ILOD)**, a detection-based paradigm of solving the visual grounding problem. **(2)** Under the ILOD paradigm, we present **INTENTLOC**, a novel detection model architecture with enhanced semantic understanding capability. The key innovation is **Crosshair Attention**, which derives an internal 2D reference point (i.e., the crosshair) from the evolving semantic query at each decoder layer. With direct geometric supervision, these crosshairs progressively shift to the target region, enhancing fine-grained GUI localization. **(3)** We introduce **Atypical-GUI-Bench**, a manually curated and carefully designed benchmark for evaluating the robustness and generalization of GUI grounding models under naturally occurring non-standard visual inputs.

2 RELATED WORK

2.1 GUI AGENT

Early computer-use agents operated in structured web environments through HTML or accessibility trees (Shi et al., 2017; Yao et al., 2022; Deng et al., 2023). With the emergence of modern VLMs, recent agents increasingly interact directly with screenshot pixels, integrating perception, reasoning, and action across platforms (Hong et al., 2024; You et al., 2024; Cheng et al., 2024; Lin et al., 2025;

Table 1: **Comparison with prior GUI grounding methods and spatially guided DETR detectors.** Unlike GUI-Actor and MolmoPoint, which score image features against a grounding token via dot product, Direct Cross-Attention Readout uses the decoder's final cross-attention logits as the output. △: via additional generated tokens rather than decoder layers.

Method	Intent Query from VLM	Iterative Query Refinement	Coarse-to-Fine Localization	Location-Guided Attention	Explicit Location Supervision	Dual Visual Feature Fusion	Direct Cross-Attention Readout
GUI-Actor (Wu et al., 2025a)	✓	✗	✗	✗	✗	✗	✗
MolmoPoint (Clark et al., 2026)	✓	△	△	△	△	✗	✗
SMCA (Gao et al., 2021)	✗	✓	✗	✓	✗	✗	✗
DAB-DETR (Liu et al., 2022)	✗	✓	✓	✓	✓	✗	✗
Grounding DINO (Liu et al., 2024a)	✗	✓	✓	✓	✓	✗	✗
IntentLoc (Ours)	✓	✓	✓	✓	✓	✓	✓

Xu et al., 2025; Qin et al., 2025; Zhou et al., 2025; Liu et al., 2024b; Yan et al., 2025; Ye et al., 2025; Li et al., 2026; Gu et al., 2025). Evaluation has likewise progressed from static grounding to realistic interactive environments involving complex, multi-step tasks (Rawles et al., 2023; Li et al., 2025b; Xie et al., 2024; Rawles et al., 2025; Kong et al., 2026).

A growing line of work (Jiang et al., 2025; Li et al., 2025a) identifies GUI grounding as a central bottleneck and proposes specialized designs to improve localization accuracy, such as predicting spatial locations directly through grounding tokens, as explored in GUI-Actor (Wu et al., 2025a) and MolmoPoint (Clark et al., 2026). Our method differs from them in three aspects: (1) instead of scoring visual features against a grounding token in a single step, it refines the grounding query layer by layer in an encoder–decoder detector; (2) unlike GUI-Actor, whose Actor token is added to the vocabulary and supervised by a next-token prediction loss, it requires only spatial supervision; and (3) it introduces directly supervised layer-wise crosshairs for precise localization of small targets. Table 1 summarizes these differences and also contrasts our design with spatially guided DETR detectors; **Appendix B** explains the criteria.

2.2 Object Detection: From Closed-Set to Open-Set, and Then to Intent-Level

Object detection has evolved from closed-set category prediction toward increasingly flexible language-conditioned localization. Classical two-stage and one-stage detectors localize objects from fixed label sets (Girshick et al., 2016; Girshick, 2015; Ren et al., 2017; Redmon et al., 2016; Liu et al., 2016; Lin et al., 2017). DETR recasts detection as set prediction with learned object queries and a Transformer encoder–decoder, inspiring improvements in convergence, multi-scale localization, and query design (Carion et al., 2020; Meng et al., 2021; Li et al., 2022a; Zhang et al., 2023). **Appendix B** relates our work to task-driven detection, LLM-driven dense decoders, and spatially guided DETR variants such as SMCA (Gao et al., 2021), Deformable DETR (Zhu et al., 2021) and DAB-DETR (Liu et al., 2022). Open-vocabulary detectors further replace fixed category prediction with text-conditioned representations, allowing targets to be specified through object names or referring phrases (Radford et al., 2021; Li et al., 2022b; Zhong et al., 2022; Liu et al., 2024a). Recent VLM-based approaches extend this progression to intent-level detection, but still cast localization as coordinate-token generation under next-token prediction (Peng et al., 2024; Jiang et al., 2026). LocateAnything (Wang et al., 2026a) introduces parallel box decoding, emitting all coordinate tokens of a geometric unit in a single forward pass. In contrast, IntentLoc adopts a query-based detection architecture and predicts a spatial patch distribution rather than coordinate tokens.

3 Pre-analysis: Why GroundingDINO Fails on GUI Grounding

Before introducing our model, we examine whether a strong open-vocabulary detector can support GUI grounding by fine-tuning GroundingDINO (Liu et al., 2024a) with single-target supervision. We compare whole intent-level instructions with extracted target spans across four training scales (500 to 1M examples) on a held-out validation split, ScreenSpot, ScreenSpot-v2, and ScreenSpot-Pro (Cheng et al., 2024; Wu et al., 2025b; Li et al., 2025b). Since the detector emits boxes rather than points, we report Hit@k, counting an example correct when the center of any of its top-k boxes falls inside the ground-truth box. We also apply a training-free OCR reranker (JaidedAI, 2020), which rescores proposals by combining their detection confidence with the text similarity between their OCR content and the target phrase, to distinguish failures of candidate localization from failures of semantic ranking. Figure 2 summarizes the results; the instruction taxonomy, complete experimental setup, and detailed analysis are provided in **Appendices C and D**.

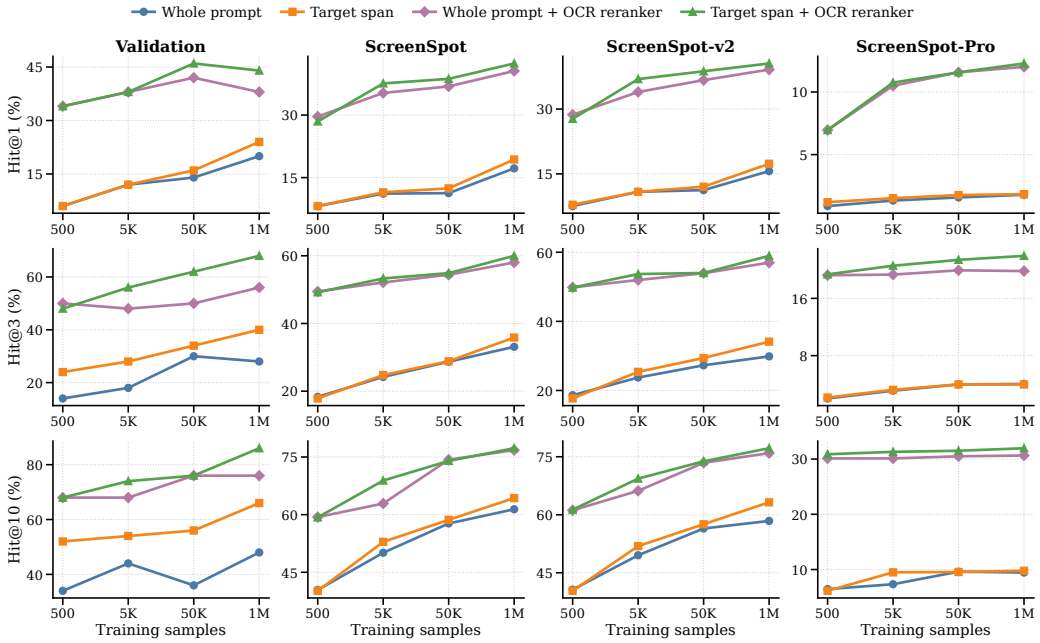


Figure 2: **Scaling GroundingDINO for GUI grounding.** Columns show the validation set, ScreenSpot-v1/v2/Pro, while rows report Hit@1, Hit@3, and Hit@10. Curves compare whole-prompt and target-span supervision, each with and without OCR reranking.

Pre-analysis Takeaways. The results reveal three main observations: **(1) Semantic ranking is the primary bottleneck.** Hit@1 remains far below Hit@3 and Hit@10, while OCR reranking substantially improves performance, indicating that GroundingDINO often localizes plausible GUI elements but fails to rank the intended target first. We further validate and visualize this finding in Appendix D. **(2) Scaling does not ensure generalization to challenging GUIs.** More training data improves performance on i.i.d. screenshots of the validation set, but gains on more professional and complicated ScreenSpot-Pro GUIs remain limited and non-monotonic. **(3) Target-span supervision provides only modest gains.** Removing action-oriented context makes the prompt more detector-friendly, but still does not substantially address GroundingDINO's weak semantic understanding.

4 Method: IntentLoc

In this section, we introduce our model IntentLoc, with the model architecture illustrated in Figure 3. It extracts an intent-level semantic query from a VLM backbone, constructs visual memory by fusing raw and contextualized visual features, and localizes the target with an L-layer Crosshair Attention decoder and Direct Cross-Attention Readout (DCAR).

Visual Memory and Semantic Query. Given a screenshot and an instruction, the vision encoder (Dosovitskiy et al., 2021) produces N raw visual tokens $\{\mathbf{e}_j\}_{j=1}^{N}$. After these tokens interact with the instruction in the VLM, we obtain their VLM hidden states $\{\mathbf{h}_j^v\}_{j=1}^{N}$, which are the contextualized visual features. The raw features retain local appearance, whereas contextualized features encode instruction-aware semantics. We fuse them element-wise with a learned scalar gate ρ:

$$\tilde{\mathbf{e}}_j = \rho\,\mathbf{e}_j + (1-\rho)\,\mathbf{h}_j^v \tag{1}$$

A standard pre-norm self-attention encoder (Vaswani et al., 2017) transforms the fused tokens into visual memory $\mathbf{Z} = \{\mathbf{z}_j\}_{j=1}^{N}$. We also add one special placeholder token at the end of the input sequence. We extract the VLM hidden state of the special placeholder token, which serves as the intent-level semantic embedding of the instruction due to the autoregressive nature of VLMs. We use this semantic embedding as the initialized decoder query $\mathbf{q}^{(0)}$.

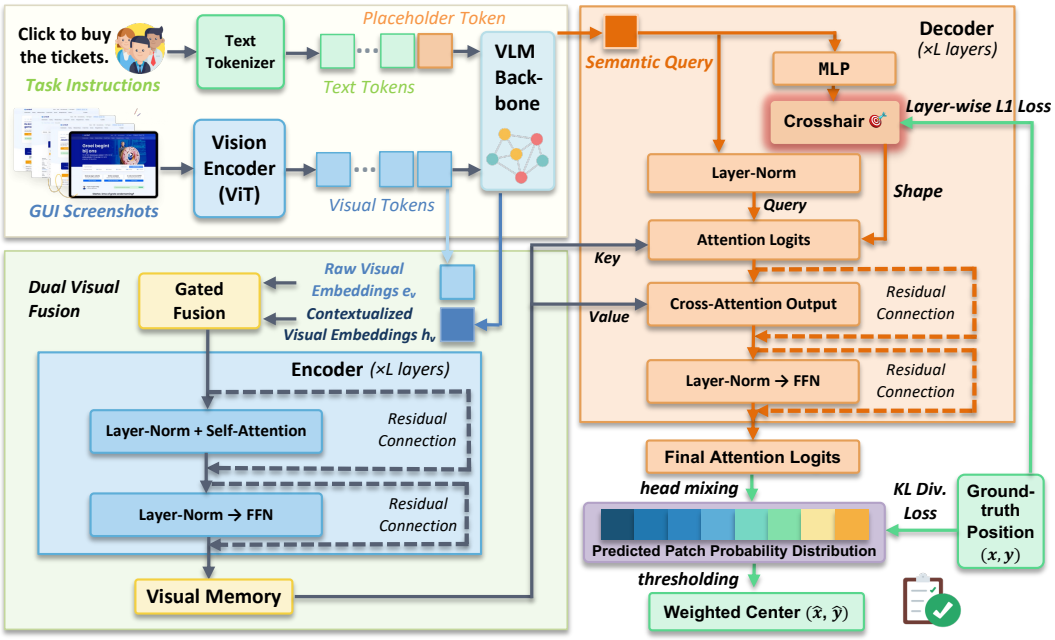


Figure 3: **Overview of INTENTLOC Architecture. Top left:** Extract the semantic query and visual features from the VLM backbone. **Bottom left:** The encoder aggregates the visual features. **Right:** The decoder introduces layer-wise crosshairs to guide cross-attention and localizes the target.

Crosshair Attention Decoder. The semantic query can describe the target element but provides no explicit location hypothesis. This is particularly limiting in high-resolution GUIs, where a small icon may occupy only a few patches and compete with thousands of visually similar distractors under global cross-attention. Therefore, we introduce Crosshair Attention to progressively locate the small target. We use l to index the L decoder layers and k to index the K attention heads. Layer l consumes the query $\mathbf{q}^{(l-1)}$ and derives a crosshair $\mathbf{r}^{(l)}$:

$$\bar{\mathbf{q}}^{(l-1)} = \mathrm{LN}\left(\mathbf{q}^{(l-1)}\right), \qquad \mathbf{r}^{(l)} = \sigma\left(f_l\left(\bar{\mathbf{q}}^{(l-1)}\right)\right), \tag{2}$$

where LN means layer normalization, f_l is a two-layer MLP, and the sigmoid σ constrains $\mathbf{r}^{(l)}$ to the normalized screen. We zero-initialize the final layer of f_l, so each crosshair initially lies at the screen center. For each head k, we then perform cross attention where the semantic query $\bar{\mathbf{q}}^{(l-1)}$ turns into $\mathbf{Q}$ and the encoded visual memory $\mathbf{z}_j$ turns into $\mathbf{K}$ and $\mathbf{V}$ in the attention formulation. Unlike vanilla cross-attention, we add a crosshair-centered spatial prior to the attention logit $a_j^{(l,k)}$:

$$a_j^{(l,k)} = \frac{\left\langle \mathbf{W}_Q^{(l,k)} \bar{\mathbf{q}}^{(l-1)}, \mathbf{W}_K^{(l,k)} \mathbf{z}_j \right\rangle}{\sqrt{d_a}} - \frac{\|\mathbf{c}_j - \mathbf{r}^{(l)}\|_2^2}{2s_l^2}, \tag{3}$$

where $\mathbf{c}_j \in [0,1]^2$ denotes the normalized center of patch j, d_a is the per-head dimension, and s_l is a learned spatial scale. The second term is an isotropic Gaussian log-kernel as in SMCA (Gao et al., 2021): it is zero at the crosshair and increasingly suppresses distant patches, while s_l controls its spread. Then, the decoder aggregates visual values with these attention logits $a_j^{(l,k)}$ to finish the cross-attention process:

$$\mathbf{o}^{(l)} = \mathbf{W}_O^{(l)} \, \mathrm{Concat}_{k=1}^{K} \left[\sum_{j=1}^{N} \mathrm{softmax}_j\left(a_j^{(l,k)}\right) \mathbf{W}_V^{(l,k)} \mathbf{z}_j \right]. \tag{4}$$

Finally, the decoder updates the query by cross-attention and Feed-Forward Network (FFN) residual connection, which is consistent with GroundingDINO (Liu et al., 2024a).

$$\begin{aligned} \tilde{\mathbf{q}}^{(l)} &= \mathbf{q}^{(l-1)} + \mathbf{o}^{(l)}, \\ \mathbf{q}^{(l)} &= \tilde{\mathbf{q}}^{(l)} + \mathrm{FFN}^{(l)}\left(\mathrm{LN}\left(\tilde{\mathbf{q}}^{(l)}\right)\right). \end{aligned} \tag{5}$$

Therefore, through Crosshair Attention, target-nearby patches retain high attention scores, whereas distant distractors are smoothly suppressed. The next layer derives a new crosshair from this refined query. Crosshair Attention thus realizes a coarse-to-fine "hypothesize, attend, and refine" process: the first layer forms a location hypothesis, and subsequent layers update it using visual evidence.

Layer-Wise Direct Geometric Supervision. INTENTLOC uses layer-wise direct geometric signal to prevent the crosshair from becoming an unconstrained latent variable. Let $\mathcal{S}$ be the set of patches intersecting the ground-truth box. We assign equal target mass to these valid patches to compute the ground-truth patch distribution y:

$$y_j = \begin{cases} |\mathcal{S}|^{-1}, & j \in \mathcal{S}, \\ 0, & j \notin \mathcal{S}, \end{cases} \qquad \sum_{j=1}^{N} y_j = 1. \tag{6}$$

The geometric target is the center of this regional distribution, $\mathbf{r}^* = \sum_{j=1}^{N} y_j \mathbf{c}_j$. We directly supervise every intermediate crosshair by L1 loss:

$$\mathcal{L}_{\text{crosshair}} = \frac{1}{L} \sum_{l=1}^{L} \text{L1}\left(\mathbf{r}^{(l)}, \mathbf{r}^*\right). \tag{7}$$

Direct Cross-Attention Readout. DCAR directly converts the final layer's attention logits into the output distribution. Let $\boldsymbol{\omega} \in \mathbb{R}^K$ be global trainable head-mixing parameters, shared across inputs and initialized to zero. The predicted probability of patch j is

$$\hat{y}_j = \text{softmax}_j \left(\sum_{k=1}^{K} [\text{softmax}(\boldsymbol{\omega})]_k \, a_j^{(L,k)} \right). \tag{8}$$

The normalized weights $\text{softmax}(\boldsymbol{\omega})$ determine the contribution of each attention head. DCAR combines the input-conditioned final-layer logits across heads and normalizes them over patches, producing the patch distribution without a separate scoring head.

Training and Inference. We compute the KL divergence between the predicted patch distribution $\hat{y}$ and the ground-truth patch distribution y as the patch loss. The overall loss is the weighted sum of the patch loss and crosshair loss, with $\lambda_{\text{crosshair}} = 0.2$:

$$\mathcal{L}_{\text{patch}} = \text{KL}(\mathbf{y} \,\|\, \hat{\mathbf{y}}), \qquad \mathcal{L} = \mathcal{L}_{\text{patch}} + \lambda_{\text{crosshair}} \mathcal{L}_{\text{crosshair}}. \tag{9}$$

Inference uses the same decoder forward pass but does not require ground-truth labels. The final attention logits produce $\hat{\mathbf{y}}$, while the crosshairs remain internal spatial guides. Then, as in GUI-Actor, we threshold $\hat{\mathbf{y}}$, group active patches into connected components, and return the weighted centroid of the highest-scoring component. Hyperparameters, training, and inference details are provided in **Appendix E**.

5 ATYPICAL-GUI-BENCH

Existing GUI grounding benchmarks predominantly use standard screenshots or controlled transformations, which do not fully reflect the heterogeneous visual observations encountered in practice. We therefore construct **Atypical-GUI-Bench**, a new benchmark for evaluating whether GUI grounding models generalize beyond standard screenshot distributions and remain robust under naturally occurring, non-canonical visual inputs. Unlike robustness sets produced by manually scaling, cropping, resizing, or simulating zoom on standard screenshots, Atypical-GUI-Bench preserves images in the forms in which they were collected in the wild. Its diverse observation categories test generalization to atypical visual formats and robustness across realistic distribution shifts. The data collection and annotation details and representative images are provided in **Appendix F**.

Sub-categories. Atypical-GUI-Bench contains 1,000 examples organized into six categories that capture naturally occurring observation shift: **(1) Partial UI view** (67 examples): a dialog, toolbar, plugin panel, application window, or other local interface region with incomplete global screen context. **(2) Long screenshot** (250 examples): a semantically continuous page or list whose length substantially exceeds a normal viewport. **(3) Multi-window or pane** (90 examples): multiple windows,

Table 2: **Overall accuracy on GUI grounding benchmarks.** The GroundingDINO-Finetuned and Qwen-Finetuned baselines are trained on the same 1.25M GUI-Actor data as INTENTLOC-GUI-3B and INTENTLOC-GUI-7B (GUI-Actor Data). Bold marks the best result at each model scale, while arrows give our improvement over the underlined Qwen-Finetuned baselines.

Model	Backbone	ScreenSpot-Pro	OSWorld-G	MMBench-GUI-L2	UI-I2E-Bench
Proprietary Model					
GPT-4o (OpenAI, 2024)	–	0.8	–	2.9	15.2
Claude Computer Use (Anthropic, 2024)	–	17.1	–	–	–
Operator (OpenAI, 2025)	–	36.6	40.6	–	–
Gemini-2.5-Pro (Comanici et al., 2025)	–	11.4	45.2	–	–
Claude 3.7 Sonnet (Anthropic, 2025a)	–	27.7	–	4.7	–
Open-Source Model (About 3B Parameter Size)					
GroundingDINO-Finetuned (GUI-Actor Data)	–	3.3	4.3	13.1	11.9
GroundingDINO-Finetuned + OCR-Reranker	–	11.6	10.1	13.1	16.7
OS-Atlas-4B (Wu et al., 2025b)	InternVL2-4B	3.7	–	–	44.3
Aria-UI (Yang et al., 2025)	Aria	11.3	–	–	–
Qwen2.5-VL-3B (Bai et al., 2025a)	–	16.1	27.3	32.9	41.7
Qwen2.5-VL-3B-Finetuned (GUI-Actor Data)	Qwen2.5-VL-3B	31.1	34.8	50.5	59.2
GUI-Actor-3B (Wu et al., 2025a)	Qwen2.5-VL-3B	42.2	40.3	73.4	63.7
INTENTLOC-GUI-3B (GUI-Actor Data)	Qwen2.5-VL-3B	**44.7** (13.6↑)	**45.7** (10.9↑)	**75.1** (24.6↑)	**67.1** (7.9↑)
Open-Source Model (7B Parameter Size)					
Qwen2.5-VL-7B (Bai et al., 2025a)	–	26.8	31.4	33.9	53.8
JEDI-7B (Xie et al., 2025)	Qwen2.5-VL-7B	39.5	54.1	75.1	–
Qwen2.5-VL-7B-Finetuned (GUI-Actor Data)	Qwen2.5-VL-7B	42.4	47.5	73.0	64.9
GUI-Actor-7B (Wu et al., 2025a)	Qwen2.5-VL-7B	44.6	42.2	76.5	68.2
INTENTLOC-GUI-7B (GUI-Actor Data)	Qwen2.5-VL-7B	49.2 (6.8↑)	51.8 (4.3↑)	81.2 (8.2↑)	73.6 (8.7↑)
UI-TARS-1.5-7B (ByteDance Seed, 2025)	Qwen2.5-VL-7B	49.6	52.8	64.3	73.2
GTA1-7B (Yang et al., 2026)	Qwen2.5-VL-7B	50.1	55.1	78.5	78.2
UI-Venus-Ground-7B (Gu et al., 2025)	Qwen2.5-VL-7B	50.8	54.6	79.9	–
UI-Ins-7B (Chen et al., 2026)	Qwen2.5-VL-7B	52.2	–	83.1	81.1
GUI-Owl-7B (Ye et al., 2025)	Qwen2.5-VL-7B	54.9	55.9	80.5	–
INTENTLOC-GUI-7B (GUI-Actor Data + 0.25M Data)	Qwen2.5-VL-7B	**58.1**	**56.4**	**85.7**	**84.4**

overlays, dialogs, or semantically distinct panes that can coexist within one continuous workspace. **(4) Photographed screen** (40 examples): a physical display captured by a camera, exhibiting cues such as device bezels, perspective distortion, reflections, moiré patterns, or environmental background. **(5) Device mockup** (233 examples): a GUI embedded in a rendered phone, tablet, laptop, or monitor with substantial surrounding non-screen visual content. **(6) Multi-panel design** (320 examples): multiple pages, interface states, artboards, or devices arranged on a presentation or design canvas, where the instruction identifies an element in one specific panel.

6 EXPERIMENTS

Training Details. We train two model variants at 3B and 7B scales. We use the same 1.25M training data and the same VLM backbone as GUI-Actor (Wu et al., 2025a), so the comparison against it shares the backbone and training data. We additionally train another version of INTENTLOC-GUI-7B on 0.25M more grounding examples, and it is therefore compared against frontier models rather than as a controlled ablation. Implementation, training, and inference details are in **Appendix E**.

Baselines. We use the same 1.25M GUI-Actor data to train a GroundingDINO (Liu et al., 2024a) model that detects a single class (the target GUI element) and uses the element-referring expression as the text prompt. To demonstrate the advantage of our INTENTLOC model over a pure language model, we also fine-tune the backbone LLMs on the same 1.25M data to obtain finetuned LLMs (Qwen2.5-VL-3B-Finetuned and Qwen2.5-VL-7B-Finetuned) as baselines.

6.1 RESULTS ON GROUNDING BENCHMARKS

We evaluate our models on four grounding benchmarks in Table 2 (the benchmarks themselves are described in **Appendix G**, while detailed and category-level results for all four benchmarks are provided in **Appendix H**). UI-I2E-Bench specifically evaluates the grounding of small GUI elements, such as small buttons and checkboxes, in high-resolution screenshots. Based on the evaluation results, we draw the following two key observations:

(1) INTENTLOC-GUI substantially outperforms the classical detection model and pure LLM baselines. Averaged across the four benchmarks, INTENTLOC-GUI-3B surpasses GroundingDINO-Finetuned and Qwen2.5-VL-3B-Finetuned by 50.0 and 14.3 points; INTENTLOC-GUI-7B, trained on the same data, outperforms Qwen2.5-VL-7B-Finetuned by an average of 7.0 points.

Table 3: **Success rates on online benchmarks**. MobileWorld results of GUI-Owl, UI-Venus, Qwen3-VL, Step-GUI-4B (GELab-Zero-4B), and Planner + UI-Ins-7B are quoted from Kong et al. (2026). We highlight in bold the best-performing results. Arrows give our improvement over the underlined planner used end to end.

Category	Model	MobileWorld				AndroidWorld	OSWorld
		GUI-Only	User-Int.	MCP	Overall		
End-to-End Model	GUI-Owl-7B (Ye et al., 2025)	7.7	–	–	4.5	66.4	29.4
	GUI-Owl-32B (Ye et al., 2025)	8.5	–	–	5.5	–	–
	UI-Venus-7B (Gu et al., 2025)	8.5	–	–	5.5	49.1	–
	UI-Venus-72B (Gu et al., 2025)	16.4	–	–	10.4	65.9	–
	Qwen3-VL-8B (Bai et al., 2025b)	9.4	0.0	0.0	5.5	47.6	33.9
	Qwen3-VL-32B (Bai et al., 2025b)	11.9	6.7	2.7	9.0	57.3	32.6
	Qwen3-VL-235B-A22B-Instruct (Bai et al., 2025b)	12.8	4.4	5.4	9.5	63.7	31.6
	Step-GUI-4B (GELab-Zero-4B) (Yan et al., 2025)	16.1	6.7	–	10.9	63.9	31.9
	Qwen3-VL-235B-A22B-Thinking (Bai et al., 2025b)	–	–	–	–	62.0	38.1
	Claude-4.5-Sonnet (Anthropic, 2025c)	47.8	38.6	50.0	46.2	83.2	–
Agentic Framework	Planner + UI-Ins-7B (Chen et al., 2026)	47.8	37.8	50.0	43.8	80.2	40.7
	Planner + GUI-Actor-7B (Wu et al., 2025a)	50.9	40.0	50.0	48.3	81.9	38.7
	Planner + IntentLoc-GUI-7B (GUI-Actor Data + 0.25M Data)	**54.7** (6.9↑)	**45.5** (6.9↑)	**57.5** (7.5↑)	**53.2** (7.0↑)	**85.1** (1.9↑)	**44.2** (6.1↑)

(2) IntentLoc-GUI consistently outperforms the corresponding GUI-Actor baseline across all four benchmarks, with average gains of 3.3 and 6.1 points for the 3B and 7B models trained on the same data. These consistent improvements suggest the effectiveness of our crosshair guidance and architectural innovations. Moreover, IntentLoc-GUI-7B trained on more data outperforms industry-developed Qwen2.5-VL-7B-based grounding models such as UI-Ins and GUI-Owl.

6.2 Results on online benchmarks

Additionally, we develop an agentic framework to apply IntentLoc-GUI to end-to-end agentic tasks. In this framework, the planner converts abstract, high-level user instructions into stepwise explicit interaction subgoals and determines the corresponding action types, while the executor predicts the precise interaction location for each step.

We evaluate our approach on three widely adopted online benchmarks, with task success rates reported in Table 3. Considering API costs, the planner is Claude-4.5-Sonnet (Anthropic, 2025c) on MobileWorld and AndroidWorld, and Qwen3-VL-235B-A22B-Thinking (Bai et al., 2025b) on OSWorld. With the planner held fixed, IntentLoc-GUI-7B is the strongest executor, ahead of both UI-Ins-7B and GUI-Actor-7B. The same ranking holds when Claude-Haiku-4.5 (Anthropic, 2025b) plans on OSWorld (**Appendix I**). Furthermore, with IntentLoc-GUI-7B as executor, the planner outperforms its end-to-end variant by 5.0 points on average across three benchmarks.

6.3 Ablation studies

We conduct ablation studies on several key components of our architecture: (1) Crosshair Attention, (2) dual visual feature fusion, (3) crosshair loss weight. Results are reported in Table 4. From these ablation results, we draw the following observations: (1) **Crosshair-guided cross-attention consistently outperforms vanilla cross-attention.** It improves accuracy by 3.7, 3.2, and 5.8 points on ScreenSpot-Pro, OSWorld-G, and UI-I2E-Bench, respectively. This gain relies on the layer-wise geometric supervision: with $\lambda_{\text{crosshair}} = 0$, the crosshair variant performs on par with vanilla cross-attention (54.3 vs. 54.4, 53.2 vs. 53.2, and 78.8 vs. 78.6), so the spatial bias helps only when each crosshair is supervised toward the target. The gain is largest on UI-I2E-Bench, which emphasizes small targets, although removing either visual feature also hurts this benchmark the most. (2) **Dual visual feature fusion is more effective than either visual feature alone.** (3) **A crosshair-loss weight around** 0.2 **provides the best balance.** Additional ablations on the number of decoder layers, together with further explanations of individual components, are provided in **Appendix J**.

7 Analysis

Convergence and Visualizations of Crosshairs. We analyze Crosshair Attention from three complementary perspectives. (1) **Qualitative analysis:** representative trajectories show that the three decoder-layer crosshairs progressively move from coarse spatial hypotheses toward the ground-truth target, after which the final prediction further refines the location. (2) **Quantitative analysis:** on ScreenSpot-Pro, the mean normalized Euclidean distance to the ground-truth bounding-box center

Table 4: **Ablation studies** on INTENTLOC-GUI-7B (GUI-Actor Data + 0.25M Data). Bold marks the best results; changes are relative to the full design, with all other settings identical.

Ablated Component	Component Value	ScreenSpot-Pro	OSWorld-G	UI-I2E-Bench
Attention Type (Whether to use crosshair)	Crosshair Attention	**58.1**	**56.4**	**84.4**
	Vanilla Cross-Attention	54.4 (3.7↓)	53.2 (3.2↓)	78.6 (5.8↓)
Visual Features	Dual Visual Feature Fusion	**58.1**	**56.4**	**84.4**
	Only Raw Visual Features	50.6 (7.5↓)	49.8 (6.6↓)	72.5 (11.9↓)
	Only Contextualized Visual Features	56.2 (1.9↓)	53.7 (2.7↓)	80.9 (3.5↓)
Crosshair Loss Weight	$\lambda_{\text{crosshair}} = 0.0$	54.3 (3.8↓)	53.2 (3.2↓)	78.8 (5.6↓)
	$\lambda_{\text{crosshair}} = 0.2$	**58.1**	**56.4**	**84.4**
	$\lambda_{\text{crosshair}} = 0.4$	56.1 (2.0↓)	**56.4**	84.2 (0.2↓)
	$\lambda_{\text{crosshair}} = 0.6$	54.8 (3.3↓)	53.9 (2.5↓)	82.7 (1.7↓)

decreases monotonically from 0.39 at layer-1 crosshair to 0.29 at layer-2 crosshair, 0.27 at layer-3 crosshair, and 0.15 at the final prediction. (3) **Attention visualization:** heatmaps obtained with and without the crosshair-guided spatial bias show that the bias strengthens and concentrates attention around the target. **Appendix K** provides the qualitative examples, statistics, and attention heatmaps.

Robustness and Generalization: Results on Atypical-GUI-Bench. We evaluate robustness and generalization under naturally occurring observation shifts on Atypical-GUI-Bench (Section 5). As shown in Table 5, INTENTLOC-GUI-3B achieves 56.1% accuracy, outperforming Qwen2.5-VL-3B-Finetuned by 16.9 points and GUI-Actor-3B by 12.4 points. The improvement is consistent across all categories, demonstrating stronger generalization beyond standard screenshots and greater robustness to heterogeneous visual shifts. More results are in **Appendix L**.

Table 5: **Accuracy on Atypical-GUI-Bench.** Arrows give our improvement over the underlined Qwen2.5-VL-3B-Finetuned baseline, which shares our backbone and training data.

Model	Device mockup	Long screenshot	Multi-panel design	Multi-window or pane	Partial UI view	Photographed screen	Overall
Qwen2.5-VL-3B-Finetuned	41.6	36.4	40.9	33.3	40.3	40.0	39.2
GUI-Actor-3B	39.5	41.6	47.5	47.8	47.8	35.0	43.7
INTENTLOC-GUI-3B	**52.8** (11.2↑)	**56.4** (20.0↑)	**59.1** (18.2↑)	**54.4** (21.1↑)	**61.2** (20.9↑)	**45.0** (5.0↑)	**56.1** (16.9↑)

Small Target. On UI-I2E-Bench (designed for small targets), INTENTLOC-GUI-3B outperforms GUI-Actor-3B by 7.0 points on compact icon targets. On Atypical-GUI-Bench, the advantage of INTENTLOC-GUI-3B over GUI-Actor-3B becomes more pronounced as the target size decreases, with accuracy gains of 16.6 points for targets occupying less than 0.1% of the image. Detailed size-stratified and element-type results are provided in **Appendix M**.

Under resolution scaling (**Appendix N**), INTENTLOC-GUI-3B is more accurate than the coordinate-prediction baseline at every scale and on par with or slightly ahead of GUI-Actor-3B. We additionally provide qualitative visualizations in **Appendix D** and failure cases in **Appendix O**.

8 LIMITATIONS

INTENTLOC currently always predicts a location and therefore cannot refuse ungroundable instructions, failing all 54 refusal examples in OSWorld-G; incorporating a no-action output is left to future work. Moreover, compared with pure language-model coordinate prediction, its encoder–decoder head adds approximately 500M parameters and the corresponding computation. INTENTLOC is about 2% slower than GUI-Actor and 9.9% faster than the pure LLM baseline, with 0.74 GiB and 2.27 GiB higher peak memory, respectively (**Appendix P**).

9 CONCLUSION

We revisit GUI grounding as intent-level object detection, revealing that conventional detectors propose GUI-element-like regions but lack the intent understanding needed to pick the intended one. By combining VLM-derived semantic queries with Crosshair Attention and direct spatial supervision, INTENTLOC consistently outperforms baselines trained on the same data. We further introduce Atypical-GUI-Bench and show that INTENTLOC improves robustness and generalization.

AI USE STATEMENT

In this work we used generative AI tools to edit the manuscript, to format LaTeX and tables, and to review code and assist with parts of its implementation. Annotators of our additional training data and of Atypical-GUI-Bench could also use a self-hosted VLM to draft instructions and suggest bounding boxes, but every such annotation was manually verified and independently checked by a second annotator (**Appendices E and F**). The research questions, the architecture, and the training objective are the authors' own; the authors decided which analyses to run and how to interpret the results. We have reviewed all AI-assisted work, and every number we report comes from our own saved experiment outputs or from the sources we cite rather than from a model. We take responsibility for the final content of this work, including any text, claims, code and artifacts produced with their aid.

ETHICS STATEMENT

INTENTLOC only locates the target of an instruction; it does not decide which action to take. In our agentic experiments (Section 6.2), that decision is made by a separate planner, and all actions are executed in the benchmarks' sandboxed environments. Our models are trained on the GUI grounding data released by GUI-Actor (Wu et al., 2025a), which aggregates several public datasets whose terms we inherit, and on 0.25M additional examples. These examples and Atypical-GUI-Bench were collected and annotated by our professional annotators (**Appendices E and F**) and will be released with this work. Because the images are captured from real interfaces, we review them before release and withhold any that contain personally identifiable information or that we are not permitted to redistribute. We release our code under Apache 2.0 and our annotations under CC BY 4.0; the screenshots themselves remain subject to their original terms. Each checkpoint follows the license of its backbone: INTENTLOC-GUI-7B is released under Apache 2.0, and INTENTLOC-GUI-3B under the non-commercial Qwen Research License. Finally, more reliable grounding also makes it easier to automate software that lacks a programmatic interface, so deployed systems should retain interface-level safeguards and should not treat a predicted location as permission to act.

REPRODUCIBILITY STATEMENT

Section 4 describes the architecture and the training objective, and **Appendix E** gives what is needed to retrain the models, from the two-stage schedule and the learning rates to the hardware and the inference read-out. **Appendices C and D** describe the GroundingDINO pre-analysis, **Appendix G** the benchmarks and the metric we report, and **Appendix H** the per-category results behind every aggregate number. **Appendices F and N** document how Atypical-GUI-Bench was built, why it is disjoint from our training data, and the resolution-scaling protocol. Our code, the trained models and both datasets will be publicly available.

REFERENCES

Anthropic. Developing a computer use model, 2024. URL https://www.anthropic.com/news/developing-computer-use.

Anthropic. Claude 3.7 Sonnet system card, 2025a. URL https://www.anthropic.com/claude-3-7-sonnet-system-card.

Anthropic. Introducing Claude Haiku 4.5, 2025b. URL https://www.anthropic.com/news/claude-haiku-4-5.

Anthropic. Introducing Claude Sonnet 4.5, 2025c. URL https://www.anthropic.com/news/claude-sonnet-4-5.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023. URL https://arxiv.org/abs/2308.12966.

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, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report, 2025a. URL https://arxiv.org/abs/2502.13923.

Shuai Bai et al. Qwen3-VL technical report, 2025b. URL https://arxiv.org/abs/2511.21631.

ByteDance Seed. UI-TARS-1.5 model card, 2025. URL https://huggingface.co/ByteDance-Seed/UI-TARS-1.5-7B.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pp. 213–229, 2020. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/html/832_ECCV_2020_paper.php.

Jikai Chen, Long Chen, Dong Wang, Qinglin Su, Zhixuan Chu, Bingguang Hao, Leilei Gan, Chenyi Zhuang, and Jinjie Gu. V2P: Visual attention calibration for GUI grounding via background suppression and center peaking, 2025. URL https://arxiv.org/abs/2508.13634.

Liangyu Chen, Hanzhang Zhou, Chenglin Cai, Jianan Zhang, Panrong Tong, Xu Zhang, Quyu Kong, Chen Liu, Yuqi Liu, Wenxuan Wang, Yue Wang, Qin Jin, and Steven Hoi. UI-Ins: Enhancing GUI grounding with multi-perspective instruction as reasoning. In *International Conference on Learning Representations*, pp. 54418–54444, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5965f3a748a8d41415db2bfa44635cc3-Abstract-Conference.html.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Annual Meeting of the Association for Computational Linguistics*, pp. 9313–9332, 2024. URL https://aclanthology.org/2024.acl-long.505.

Christopher Clark, Yue Yang, Jae Sung Park, Zixian Ma, Jieyu Zhang, Rohun Tripathi, Mohammadreza Salehi, Sangho Lee, Taira Anderson, Winson Han, and Ranjay Krishna. MolmoPoint: Better pointing for VLMs with grounding tokens, 2026. URL https://arxiv.org/abs/2603.28069.

Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023.

URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of DETR with spatially modulated co-attention. In *IEEE/CVF International Conference on Computer Vision*, pp. 3621–3630, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Gao_Fast_Convergence_of_DETR_With_Spatially_Modulated_Co-Attention_ICCV_2021_paper.html.

Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015. URL https://openaccess.thecvf.com/content_iccv_2015/html/Girshick_Fast_R-CNN_ICCV_2015_paper.html.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016. URL https://ieeexplore.ieee.org/document/7112511.

Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *International Conference on Learning Representations*, pp. 30851–30883, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4ca0e369689dadb25a5345ba9755ad6f-Abstract-Conference.html.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, Yue Wen, Jingya Dou, Fei Tang, Jinzhen Lin, Yulin Liu, Zhenlin Guo, Yichen Gong, Heng Jia, Changlong Gao, Yuan Guo, Yong Deng, Zhenyu Guo, Liang Chen, and Weiqiang Wang. UI-Venus technical report: Building high-performance UI agents with RFT, 2025. URL https://arxiv.org/abs/2508.10833.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: A visual language model for GUI agents. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Hong_CogAgent_A_Visual_Language_Model_for_GUI_Agents_CVPR_2024_paper.html.

JaidedAI. EasyOCR: Ready-to-use OCR with 80+ supported languages, 2020. URL https://github.com/JaidedAI/EasyOCR.

Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, Xinda Xue, Qinghang Su, Huaihai Lyu, Xiaolong Zheng, Jiaming Liu, Zhongyuan Wang, and Shanghang Zhang. RoboBrain: A unified brain model for robotic manipulation from abstract to concrete. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1724–1734, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Ji_RoboBrain_A_Unified_Brain_Model_for_Robotic_Manipulation_from_Abstract_CVPR_2025_paper.html.

Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25472–25483, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Jiang_Detect_Anything_via_Next_Point_Prediction_CVPR_2026_paper.html.

Zhiyuan Jiang, Shenghao Xie, Wenyi Li, Wenqiang Zu, Peihang Li, Jiahao Qiu, Siqi Pei, Lei Ma, Tiejun Huang, Mengdi Wang, and Shilong Liu. Zoom in, click out: Unlocking and evaluating the potential of zooming for GUI grounding, 2025. URL https://arxiv.org/abs/2512.05941.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, Zhidan Liu, Steven Hoi, and Yue Wang. MobileWorld: Benchmarking autonomous mobile agents in agent-user interactive and MCP-augmented environments. In *Annual Meeting of the Association for Computational Linguistics*, pp. 6142–6167, 2026. URL https://aclanthology.org/2026.acl-long.278.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. LISA: Reasoning segmentation via large language model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9579–9589, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Lai_LISA_Reasoning_Segmentation_via_Large_Language_Model_CVPR_2024_paper.html.

Aiden Yiliu Li, Bizhi Yu, Daoan Lei, Tianhe Ren, and Shilong Liu. Chain-of-Ground: Improving GUI grounding via iterative reasoning and reference feedback, 2025a. URL https://arxiv.org/abs/2512.01979.

Aiden Yiliu Li, Xinyue Hao, Shilong Liu, and Mengdi Wang. Avenir-Web: Human-experience-imitating multimodal web agents with mixture of grounding experts, 2026. URL https://arxiv.org/abs/2602.02468.

Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M. Ni, and Lei Zhang. DN-DETR: Accelerate DETR training by introducing query DeNoising. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13619–13627, 2022a. URL https://openaccess.thecvf.com/content/CVPR2022/html/Li_DN-DETR_Accelerate_DETR_Training_by_Introducing_Query_DeNoising_CVPR_2022_paper.html.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. ScreenSpot-Pro: GUI grounding for professional high-resolution computer use. In *ACM International Conference on Multimedia*, pp. 8778–8786, 2025b. URL https://dl.acm.org/doi/10.1145/3746027.3755688.

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10965–10975, 2022b. URL https://openaccess.thecvf.com/content/CVPR2022/html/Li_Grounded_Language-Image_Pre-Training_CVPR_2022_paper.html.

Pengfei Li, Beiwen Tian, Yongliang Shi, Xiaoxue Chen, Hao Zhao, Guyue Zhou, and Ya-Qin Zhang. TOIST: Task oriented instance segmentation transformer with noun-pronoun distillation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17597–17611, 2022c. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/70270a1bc28ecb2a2aefad566c5e556b-Abstract-Conference.html.

Weiming Li, Yan Shao, Jing Yang, Yujing Lu, Ling Zhong, Yuhan Wang, Min Yu, Tongxiao Ruan, and Manni Duan. How auxiliary reasoning unleashes GUI grounding in VLMs, 2025c. URL https://arxiv.org/abs/2509.11548.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. ShowUI: One vision-language-action model for GUI visual agent. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19498–19508, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Lin_ShowUI_One_Vision-Language-Action_Model_for_GUI_Visual_Agent_CVPR_2025_paper.html.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, pp.

2980–2988, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html.

Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: Dynamic anchor boxes are better queries for DETR. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=oMI9PjOb9Jl.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pp. 38–55, 2024a. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/6319_ECCV_2024_paper.php.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pp. 21–37, 2016. URL https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, Xinyi Liu, Xinying Chen, Xinyue Yang, Yang Yang, Yifan Xu, Yu Yang, Yujia Wang, Yulin Xu, Zehan Qi, Yuxiao Dong, and Jie Tang. AutoGLM: Autonomous foundation agents for GUIs, 2024b. URL https://arxiv.org/abs/2411.00820.

Xinyi Liu, Xiaoyi Zhang, Ziyun Zhang, and Yan Lu. UI-E2I-Synth: Advancing GUI grounding with large-scale instruction synthesis. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15668–15684, 2025. URL https://aclanthology.org/2025.findings-acl.809.

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

Yuhao Lu, Yixuan Fan, Beixing Deng, Fangfu Liu, Yali Li, and Shengjin Wang. VL-Grasp: a 6-Dof interactive grasp policy for language-oriented objects in cluttered indoor scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 976–983, 2023. URL https://ieeexplore.ieee.org/document/10341379.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, Guanjing Xiong, and Hongsheng Li. UI-R1: Enhancing efficient action prediction of GUI agents by reinforcement learning. In *AAAI Conference on Artificial Intelligence*, volume 40, pp. 17608–17616, 2026. URL https://ojs.aaai.org/index.php/AAAI/article/view/38816.

Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. GUI-R1: A generalist R1-style vision-language action model for GUI agents, 2025. URL https://arxiv.org/abs/2504.10458.

Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional DETR for fast training convergence. In *IEEE/CVF International Conference on Computer Vision*, pp. 3651–3660, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Meng_Conditional_DETR_for_Fast_Training_Convergence_ICCV_2021_paper.html.

Microsoft Research. UI-I2E-Bench leaderboard, 2026. URL https://microsoft.github.io/FIVE-UI-Evol. Accessed: 2026-08-21.

OpenAI. GPT-4o system card, 2024. URL https://openai.com/index/gpt-4o-system-card.

OpenAI. Operator system card, 2025. URL https://openai.com/index/operator-system-card.

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, Qixiang Ye, and Furu Wei. Grounding multimodal large language models to the world. In *International Conference on Learning Representations*, pp. 51575–51598, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e112a4671e8779aa9f640a0e3f81bd26-Abstract-Conference.html.

Renjie Pi, Jiahui Gao, Shizhe Diao, Rui Pan, Hanze Dong, Jipeng Zhang, Lewei Yao, Jianhua Han, Hang Xu, Lingpeng Kong, and Tong Zhang. DetGPT: Detect what you need via reasoning. In *Conference on Empirical Methods in Natural Language Processing*, pp. 14172–14189, 2023. URL https://aclanthology.org/2023.emnlp-main.876.

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

Mengxue Qu, Yu Wu, Wu Liu, Xiaodan Liang, Jingkuan Song, Yao Zhao, and Yunchao Wei. RIO: A benchmark for reasoning intention-oriented objects in open environments. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43041–43056, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8644353f7d307baaf29bc1e56fe8e0ec-Abstract-Datasets_and_Benchmarks.html.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, volume 139, pp. 8748–8763, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, pp. 406–441, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/01a83bc2f2732a58e6aa731e659e7101-Abstract-Conference.html.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. AndroidInTheWild: A large-scale dataset for Android device control. In *Advances in Neural Information Processing Systems*, volume 36, pp. 59708–59728, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/bbbb6308b402fe909c39dd29950c32e0-Abstract-Datasets_and_Benchmarks.html.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. URL https://ieeexplore.ieee.org/document/7485869.

Johann Sawatzky, Yaser Souri, Christian Grund, and Jurgen Gall. What object should I use? – task driven object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Sawatzky_What_Object_Should_I_Use_-_Task_Driven_Object_Detection_CVPR_2019_paper.html.

Tianlin Tim Shi, Andrej Karpathy, Linxi Jim Fan, Jonathan Hernandez, and Percy Liang. World of Bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, volume 70, pp. 3135–3144, 2017. URL https://proceedings.mlr.press/v70/shi17a.html.

Jiajin Tang, Ge Zheng, Jingyi Yu, and Sibei Yang. CoTDet: Affordance knowledge prompting for task driven object detection. In *IEEE/CVF International Conference on Computer Vision*, pp. 3068–3078, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Tang_CoTDet_Affordance_Knowledge_Prompting_for_Task_Driven_Object_Detection_ICCV_2023_paper.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution, 2024a. URL https://arxiv.org/abs/2409.12191.

Shihao Wang, Shilong Liu, Yuanguo Kuang, Xinyu Wei, Yangzhou Liu, Zhiqi Li, Yunze Man, Guo Chen, Andrew Tao, Guilin Liu, Jan Kautz, Lei Zhang, and Zhiding Yu. LocateAnything: Fast and high-quality vision-language grounding with parallel box decoding. In *European Conference on Computer Vision*, pp. 336–357, 2026a. URL https://link.springer.com/chapter/10.1007/978-3-032-37718-0_20.

Weiyun Wang, Yiming Ren, Haowen Luo, Tiantong Li, Chenxiang Yan, Zhe Chen, Wenhai Wang, Qingyun Li, Lewei Lu, Xizhou Zhu, Yu Qiao, and Jifeng Dai. The All-Seeing project V2: Towards general relation comprehension of the open world. In *European Conference on Computer Vision*, pp. 471–490, 2024b. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/4939_ECCV_2024_paper.php.

Wenkai Wang, Xiyun Li, Hongcan Guo, Wenhao Yu, Tianqing Fang, Haitao Mi, Dong Yu, and Shengyu Zhang. Measure twice, click once: Co-evolving proposer and visual critic via reinforcement learning for GUI grounding. In *Annual Meeting of the Association for Computational Linguistics*, pp. 20964–20984, 2026b. URL https://aclanthology.org/2026.acl-long.960.

Xiaoce Wang, Guibin Zhang, Junzhe Li, Jinzhe Tu, Chun Li, and Ming Li. ToolTok: Tool tokenization for efficient and generalizable GUI agents, 2026c. URL https://arxiv.org/abs/2602.02548.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, Weiyun Wang, Xiangyu Zhao, Jixuan Chen, Haodong Duan, Tianbao Xie, Chenyu Yang, Shiqian Su, Yue Yu, Yanting Zhang, Xiangyu Yue, Weijie Su, Xizhou Zhu, Wei Shen, Jifeng Dai, and Wenhai Wang. MMBench-GUI: A unified hierarchical evaluation framework for multi-platform GUI agents. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6239–6248, 2026d. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wang_MMBench-GUI_A_Unified_Hierarchical_Evaluation_Framework_for_Multi-Platform_GUI_Agents_CVPR_2026_paper.html.

Jiannan Wu, Muyan Zhong, Sen Xing, Zeqiang Lai, Zhaoyang Liu, Zhe Chen, Wenhai Wang, Xizhou Zhu, Lewei Lu, Tong Lu, Ping Luo, Yu Qiao, and Jifeng Dai. VisionLLM v2: An end-to-end generalist multimodal large language model for hundreds of vision-language tasks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 69925–69975, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/81a60d18e010b27b36cd465c6604b915-Abstract-Conference.html.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal LLMs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13084–13094, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Wu_V_Guided_Visual_Search_as_a_Core_Mechanism_in_Multimodal_CVPR_2024_paper.html.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, Si Qin, Lars Liden, Qingwei Lin, Huan Zhang, Tong Zhang, Jianbing Zhang, Dongmei Zhang, and Jianfeng Gao. GUI-Actor: Coordinate-free visual grounding for GUI agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 15101–15128, 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/16130af940e9dabb43c726119bd3b42e-Abstract-Conference.html.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. OS-ATLAS: Foundation action model for generalist GUI agents. In *International Conference on Learning Representations*, pp. 5090–5108, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0faa4bc5f522076947a030273629d4fe-Abstract-Conference.html.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis. In *Advances in Neural Information Processing Systems*, volume 38, pp. 27405–27462, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/22c868099177ee278eb7baccec649f35-Abstract-Datasets_and_Benchmarks_Track.html.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous GUI interaction. In *International Conference on Machine Learning*, volume 267, pp. 69772–69805, 2025. URL https://proceedings.mlr.press/v267/xu25ae.html.

Haolong Yan et al. Step-GUI technical report, 2025. URL https://arxiv.org/abs/2512.15431.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, Ran Xu, Liyuan Pan, Caiming Xiong, and Junnan Li. GTA1: GUI test-time scaling agent. In *International Conference on Learning Representations*, pp. 48295–48323, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4fb7d5059cdbe2db2b62dcfe14caa658-Abstract-Conference.html.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-UI: Visual grounding for GUI instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22418–22433, 2025. URL https://aclanthology.org/2025.findings-acl.1152.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. Mobile-Agent-v3: Fundamental agents for GUI automation, 2025. URL https://arxiv.org/abs/2508.15144.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-UI: Grounded mobile UI understanding with multimodal LLMs. In *European Conference on Computer Vision*, pp. 240–255, 2024. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/8095_ECCV_2024_paper.php.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. RoboPoint: A vision-language model for spatial affordance prediction in robotics. In *Conference on Robot Learning*, volume 270, pp. 4005–4020, 2025. URL https://proceedings.mlr.press/v270/yuan25c.html.

Yifu Yuan, Haiqin Cui, Yibin Chen, Zibin Dong, Fei Ni, Longxin Kou, Jinyi Liu, Pengyi Li, Yan Zheng, and Jianye Hao. From seeing to doing: Bridging reasoning and decision for robotic manipulation. In *International Conference on Learning Representations*, pp. 129927–129960, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d2eedb7ecf14b50967c5de98e30e0ffc-Abstract-Conference.html.

Yifu Yuan, Haiqin Cui, Yaoting Huang, Yibin Chen, Fei Ni, Zibin Dong, Pengyi Li, Yan Zheng, Hongyao Tang, and Jianye Hao. Embodied-R1: Reinforced embodied reasoning for general robotic manipulation. In *International Conference on Learning Representations*, pp. 8984–9009, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0f7ad318868d228b980b13cf827e85f4-Abstract-Conference.html.

Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. DINO: DETR with improved DeNoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3mRwyG5one.

Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, and Jianfeng Gao. RegionCLIP: Region-based language-image pretraining. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16793–16803, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Zhong_RegionCLIP_Region-Based_Language-Image_Pretraining_CVPR_2022_paper.html.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, and Steven Hoi. MAI-UI technical report: Real-world centric foundation GUI agents, 2025. URL https://arxiv.org/abs/2512.22047.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=gZ9hCDWe6ke.

Appendix

TABLE OF CONTENTS IN APPENDIX

A Natural-Image Generalization **20**

B Additional Related Work **20**

C Three Levels of GUI Instructions **22**

D GroundingDINO Pre-Analysis Details and Visualizations **22**

D.1 Experimental Setup 22

D.2 Detailed Findings 23

D.3 Qualitative Visualizations 23

E Implementation, Training and Inference Details **26**

F Details of Atypical-GUI-Bench **29**

G Benchmarks **33**

G.1 Grounding Benchmarks 33

G.2 Online Benchmarks 34

H Detailed Grounding Benchmark Results **34**

I Online Benchmark Results with a Different Planner **37**

J Additional Ablation Studies and Component Analysis **37**

K Convergence and Visualizations of Crosshairs **38**

L More Results on Atypical-GUI-Bench **42**

M Small-Target Analysis **42**

M.1 UI-I2E-Bench 42

M.2 Atypical-GUI-Bench 42

N Resolution-Scaling Analysis **43**

O Failure Cases of INTENTLOC **44**

P Inference Efficiency and Memory **49**

A NATURAL-IMAGE GENERALIZATION

To examine whether the INTENTLOC architecture generalizes beyond GUI screenshots, we evaluate it on a natural-image pointing task. Using the same model architecture and training recipe as INTENTLOC-GUI-3B, we train INTENTLOC-EMBODIED-3B with Qwen2.5-VL-3B-Instruct (Bai et al., 2025a) as the backbone on only 35,000 examples from the RoboRefIt training set (Lu et al., 2023). These examples are fully contained in the 200K training corpus used by the Embodied-R1 baseline (Yuan et al., 2026b); thus, our model uses substantially less training data from the same source. Embodied-R1 is a general-purpose embodied model rather than a matched in-domain fine-tune, so this is not a controlled comparison.

RoboRefIt (Lu et al., 2023) is a robotic referring-expression grounding benchmark collected for language-guided interaction in cluttered indoor scenes. It contains 10,872 real-world RGB-D images and 50,758 referring expressions spanning 66 object categories. Nearly half of the images involve multiple instances of the same category, making the target uniquely identifiable only from the language instruction. Following Embodied-R1, a prediction is counted as correct when the predicted point falls inside the target region.

Table 6: **Pointing accuracy (%) on RoboRefIt (Lu et al., 2023).** Baseline scores are taken from Embodied-R1 (Yuan et al., 2026b). The metric is the accuracy of predicted points falling inside the target region. The arrow gives our improvement over the underlined second-best baseline.

Model	RoboRefIt
GPT-4o (OpenAI, 2024)	15.28
RoboBrain (Ji et al., 2025)	10.10
ASMv2 (Wang et al., 2024b)	48.40
RoboPoint (Yuan et al., 2025)	49.82
FSD (Yuan et al., 2026a)	56.73
Qwen2.5-VL (Bai et al., 2025a)	74.90
Embodied-SFT (Yuan et al., 2026b)	83.85
Embodied-R1 (Yuan et al., 2026b)	85.58
INTENTLOC-EMBODIED-3B (ours)	**87.05** (1.47 ↑)

As shown in Table 6, INTENTLOC-EMBODIED-3B attains 87.05% accuracy, outperforming Embodied-R1 (85.58%) and all other reported baselines. These results indicate that the architecture is not limited to GUI grounding: it also improves referring-expression pointing in natural images of cluttered real-world scenes, even with substantially less training data.

B ADDITIONAL RELATED WORK

Task-driven detection and LLM-driven decoders. Localizing the object implied by a task or an intention, rather than one named explicitly, has been studied in natural images as task-driven or intention-oriented detection (Sawatzky et al., 2019; Li et al., 2022c; Tang et al., 2023; Qu et al., 2023). Several multimodal models also decode a mask or a box from the hidden state of a special LLM token (Pi et al., 2023; Lai et al., 2024; Wu et al., 2024), which is the same interface through which INTENTLOC obtains its query. ILOD adopts this view for GUIs, where targets are small, dense and text-heavy and the output is a single click point. Within GUI grounding, V2P (Chen et al., 2025) refines GUI-Actor-style patch attention by suppressing background patches and sharpening its center, whereas INTENTLOC changes how the query and the visual memory are formed and how they interact.

Relation to spatially guided DETR variants. Crosshair Attention shares with SMCA, DAB-DETR, and Deformable DETR the broad principle that an explicit spatial hypothesis can make decoder attention more effective (Gao et al., 2021; Liu et al., 2022; Zhu et al., 2021). The mechanisms and objectives, however, are different. SMCA predicts a center and two axis-specific scales

from each generic object query and uses the resulting Gaussian-like map to modulate dense co-attention, primarily to accelerate the convergence of DETR on multi-object detection. **Importantly, this Gaussian-like map receives no separate, explicit spatial supervision; it is learned as an implicit spatial prior and is supervised only indirectly through DETR's end-to-end detection losses.** Crosshair Attention adopts SMCA's additive Gaussian log-kernel (Eq. 3) but instead begins with a single intent-conditioned semantic query from a VLM, predicts a 2D point at every decoder layer, and **directly supervises each point against the target-region center**. Its learned spatial scale is layer-specific but isotropic, and its final biased attention logits are themselves converted into the patch-level localization output by DCAR.

DAB-DETR represents each object query by a 4D anchor box, updates that box layer by layer, and uses its width and height to modulate positional attention. A crosshair does not represent object extent or serve as an anchor-box query: it is an intermediate point hypothesis derived from the evolving semantic query, while the target region is represented separately by the supervised patch distribution. Deformable DETR takes another route, attending only to a small learned set of sampling points around each reference point over multi-scale feature maps. Crosshair Attention retains dense access to all visual patches and adds a smooth distance-dependent bias, so evidence outside the current hypothesis is suppressed rather than excluded. Conditional DETR (Meng et al., 2021) likewise predicts a reference point from each object query to form a spatial query for cross-attention, but the point is predicted once from a static object query, shared by all decoder layers, and trained only indirectly through the box loss, whereas a crosshair is re-predicted from the evolving intent query at every layer and supervised directly.

Thus, our novelty is not the use of a spatial reference alone. It lies in combining an intent-level VLM query, directly supervised layer-wise crosshairs, dense Gaussian-biased cross-attention, and attention-based patch readout for single-target GUI grounding. This design turns the reference point into an interpretable, progressively refined localization trajectory rather than a component of generic set-based box detection.

Criteria for Table 1. Each column of Table 1 marks one design property.

- *Intent Query from VLM*: the query that localizes the target is the hidden state of a VLM token that has read the instruction. GUI-Actor (its `<ACTOR>` token) and MolmoPoint (its grounding tokens) satisfy this, whereas the DETR variants use learned object queries or queries initialized from image features, and Grounding DINO encodes the text with a separate BERT encoder.
- *Iterative Query Refinement*: the query is updated over several steps before the final prediction, as in every DETR decoder; GUI-Actor scores patches with a single `<ACTOR>` hidden state.
- *Coarse-to-Fine Localization*: an early location estimate is refined by later steps. DAB-DETR and Grounding DINO refine anchor boxes layer by layer, whereas SMCA predicts its Gaussian center and scale once from each object query.
- *Location-Guided Attention*: attention is modulated or restricted by an explicit location estimate, through a Gaussian weight map (SMCA), anchor-box modulation (DAB-DETR), deformable sampling around reference points (Grounding DINO), or the crosshair bias (ours).
- *Explicit Location Supervision*: the location estimate that guides attention receives its own explicit loss. Our crosshairs are supervised by an L1 loss at every decoder layer, and DAB-DETR and Grounding DINO supervise the box predicted at each decoder layer, which serves as the next layer's anchor, through auxiliary detection losses. SMCA's Gaussian map receives no separate supervision: it is an implicitly learned spatial prior, trained only indirectly through DETR's detection loss. MolmoPoint supervises each patch and subpatch selection with the ground-truth location. Table 4 shows that this supervision is essential: keeping the crosshair but setting $\lambda_{\text{crosshair}} = 0$ performs on par with vanilla cross-attention.
- *Dual Visual Feature Fusion*: vision-encoder features and the VLM hidden states of the visual tokens are fused into a single visual memory that the localizer attends to (Eq. 1). MolmoPoint uses both kinds of features but at separate stages, LLM hidden states to select a patch and ViT features to select a subpatch within it, without fusing them. GUI-Actor

applies one self-attention layer to vision-encoder features only, and the DETR variants attend to the outputs of their own encoders.

- *Direct Cross-Attention Readout*: the output distribution is read from the decoder's own final cross-attention logits rather than from a separate scoring head. GUI-Actor and MolmoPoint instead score image features against a grounding token with dedicated projections, and the DETR variants predict boxes with FFN heads.

MolmoPoint obtains four of these properties by emitting additional tokens rather than through decoder layers (△): its <PATCH> token selects a patch using the LLM hidden states, its <SUBPATCH> token then attends only to the ViT features inside that patch, and a <LOCATION> token selects the final point, with each selection supervised by the ground-truth location.

C Three Levels of GUI Instructions

GUI instructions can be organized into three levels according to how directly they specify an executable action and its visual target:

1. **Task-level instructions** express a high-level goal that typically spans multiple actions, screenshots, or application states, such as "Order a cup of milk tea." Such an instruction cannot be grounded to a single GUI element immediately. A VLM planner first decomposes it into a sequence of intent-level instructions, and an executor grounds and performs each instruction based on the current screenshot. The planner may update subsequent instructions as the interface state changes. We evaluate this planner–executor treatment of task-level instructions in Section 6.2.
2. **Intent-level instructions** specify a single operation that can be completed on the current screenshot, while describing the desired outcome rather than necessarily naming the target element, such as "Run compile in VSCode." Resolving such an instruction requires understanding both the user's intent and the functional semantics of the visible GUI elements. Our training data and GUI grounding benchmarks, including ScreenSpot (Cheng et al., 2024) and ScreenSpot-Pro (Li et al., 2025b), primarily consist of instructions at this level.
3. **Target-span-level instructions** directly describe the visual element to be localized, such as "the button in the upper-left corner of the browser." They typically omit the action verb and retain only the explicit referring expression for the target. Compared with intent-level instructions, they reduce the need for functional reasoning and more closely resemble the category names or referring expressions used by conventional object detectors.

These levels form an abstraction hierarchy from a user goal to an executable operation and finally to a concrete visual referent. In our setting, a planner converts a task-level instruction into intent-level instructions, whereas target-span extraction removes the action-oriented context from an intent-level instruction for detector supervision. We focus on intent-level grounding in the single-step experiments.

D GroundingDINO Pre-Analysis Details and Visualizations

D.1 Experimental Setup

We choose GroundingDINO (Liu et al., 2024a), a strong and widely used open-vocabulary detection architecture, and fine-tune its grounded-pretrained backbone on GUI grounding data. GUI grounding differs from conventional object detection in two respects:

1. **Single-target supervision.** Whereas an object-detection image may contain multiple annotated objects, each GUI grounding instance specifies one target. We therefore associate each example with a single ground-truth box while otherwise following the standard open-vocabulary detection fine-tuning procedure.
2. **Intent-level instructions.** Object-detection prompts typically name explicit categories, whereas GUI instructions often describe an intended operation and identify the target only

indirectly. We focus on single-step intent-level instructions, following the taxonomy in **Appendix C**.

We compare two prompt formulations. *Whole-prompt* training treats the complete instruction as the positive textual span, while *target-span* training retains only the phrase describing the target element. We obtain the latter by removing a leading action phrase; for example, "Please click on the save button" becomes "save button." Instructions that already describe only the target, such as "the button in the upper-right corner of the browser window," remain unchanged. Specifically, we use a rule-based procedure: we compile a lexicon containing all action verbs in the instructions and remove any lexicon-matched leading action phrase to obtain the target span.

Because additional epochs yield only marginal preliminary gains, we train each variant for one epoch. To study scaling, we increase the number of fine-tuning examples from 500 to 5K, 50K, and 1M, evaluating each model on a held-out in-distribution split and on ScreenSpot, ScreenSpot-v2, and ScreenSpot-Pro (Cheng et al., 2024; Wu et al., 2025b; Li et al., 2025b). In the pre-analysis, both the GroundingDINO training data and the validation split are drawn from the 1.25M grounding examples in GUI-Actor. The validation split contains 500 examples and is completely disjoint from all fine-tuning subsets. At inference, GroundingDINO returns multiple scored boxes. We rank them by confidence and report Hit@k, considering an example correct when the center of any top-k box lies inside the ground-truth box.

We further decompose GUI grounding into *candidate localization*, which finds plausible interactive elements, and *semantic ranking*, which selects the element matching the user's intent. To isolate ranking errors, we introduce a training-free reranker that applies EasyOCR (JaidedAI, 2020) to each proposal and combines lexical similarity to the target phrase with the original detection confidence. Because the lexical evidence comes from OCR, a proposal with no readable text — most icons, for example — is ranked by detection confidence alone. The reranker can therefore only recover text-bearing targets, so its gain is a lower bound on how much weak semantic ranking costs GroundingDINO.

D.2 Detailed Findings

The experiments support three conclusions:

1. **Semantic ranking is the primary bottleneck.** Hit@1 is consistently much lower than Hit@3 and Hit@10, indicating that the correct target is often among the candidates but is not ranked first. The relatively high Hit@10 and the effectiveness of detector-based GUI parsers (Lu et al., 2024) show that object detectors retain useful GUI-element localization ability. In contrast, the low Hit@1 and gains from OCR reranking expose weak instruction-conditioned ranking.
2. **Scaling does not ensure generalization to challenging GUIs.** Performance generally improves with more data on the validation set, ScreenSpot, and ScreenSpot-v2. Gains on ScreenSpot-Pro are limited and occasionally non-monotonic, however, suggesting that adding examples from the existing simpler distribution does not close the gap on scarce professional interfaces.
3. **Target-span supervision provides only modest gains.** Removing action-oriented context helps GroundingDINO focus on the visual target concept, further indicating limited understanding of complete user instructions. The gain remains small partly because many instructions already contain no action phrase, making their whole-prompt and target-span formulations identical.

D.3 Qualitative Visualizations

GroundingDINO can localize GUI elements but cannot perform semantic ranking. We first examine whether GroundingDINO's poor target-grounding accuracy arises from an inability to detect GUI elements or from weak instruction-conditioned ranking. All statistics in this analysis are computed using the best-performing GroundingDINO checkpoint among those trained on GUI grounding data, as described in Section 3. We treat an element candidate as localized when its predicted center falls inside an interactive region detected by OmniParser (Lu et al., 2024) or a text

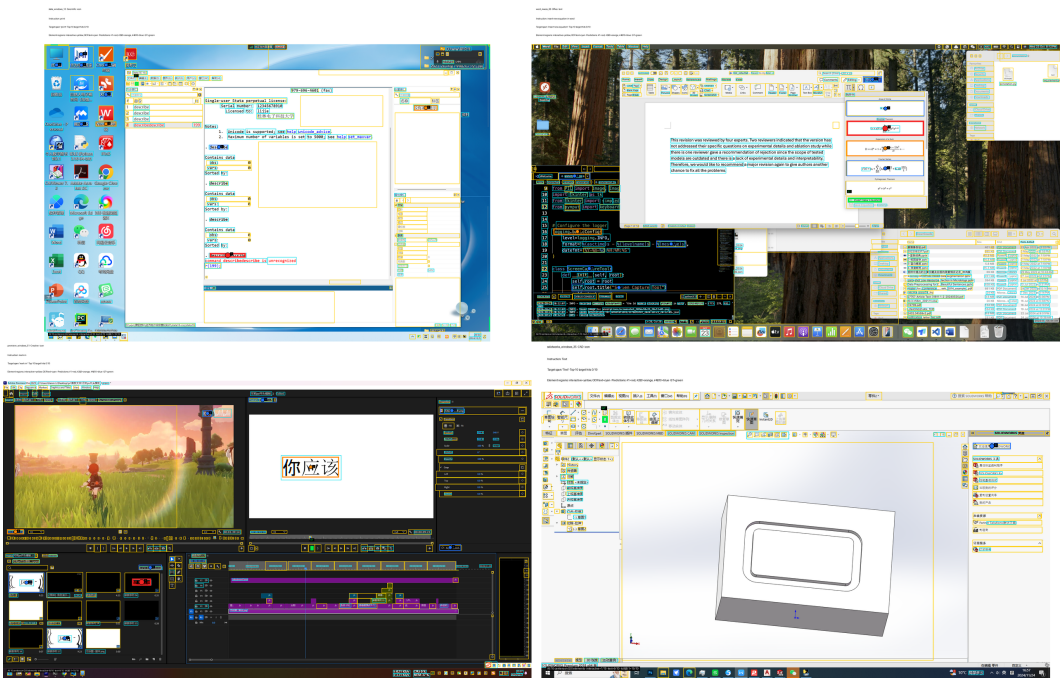


Figure 4: **GroundingDINO localizes GUI elements but ranks the instructed target poorly.** Four ScreenSpot-Pro examples from Stata (top left), Microsoft Word (top right), Adobe Premiere (bottom left), and SolidWorks (bottom right) are shown. In every example, all ten GroundingDINO candidate centers lie inside an OmniParser interactive region or EasyOCR text region, while none lies inside the instruction's ground-truth target. Yellow boxes denote OmniParser regions, cyan boxes denote OCR text regions, green boxes denote ground-truth targets, and red/orange/blue boxes denote GroundingDINO candidates ranked from 1 to 10.

region detected by EasyOCR (JaidedAI, 2020). Across all 1,581 ScreenSpot-Pro examples, 70.19% of GroundingDINO's Top-10 candidate centers fall inside such GUI-element regions, and 99.62% of screenshots contain at least one GUI-element candidate among the Top-10 predictions. However, the instruction-specified target appears in the Top 10 for only 9.45% of screenshots. These results provide additional quantitative and qualitative support for the first takeaway of the pre-analysis in Section 3.

Figure 4 visualizes four typical examples where GroundingDINO repeatedly proposes visually valid controls, icons, or text regions, yet assigns high scores to elements that are semantically unrelated to the instruction. These results separate *elementness* from *target relevance*: GroundingDINO retains substantial GUI-element localization ability, but its text encoder and matching mechanism are insufficient for selecting the element implied by the user's intent.

INTENTLOC can perform semantic target selection, whereas GroundingDINO cannot. Figure 5 compares INTENTLOC-GUI-3B with the target-span and whole-prompt variants of GroundingDINO (Liu et al., 2024a) on two representative examples. Green boxes denote ground-truth targets, red boxes indicate Top-1 predictions, and orange/yellow boxes show subsequent ranked candidates.

In the Android Studio example, INTENTLOC combines the complete instruction with the interface context and correctly assigns its Top-1 prediction to the *Contacts* button in the virtual phone's bottom navigation bar. In contrast, target-span GroundingDINO ranks the central illustration first, while whole-prompt GroundingDINO favors the search field at the top of the phone screen.

The Blender example requires a finer-grained compositional distinction. For the instruction "Change position along Y axis using text input," INTENTLOC jointly interprets *Y axis* and *text input* and cor-

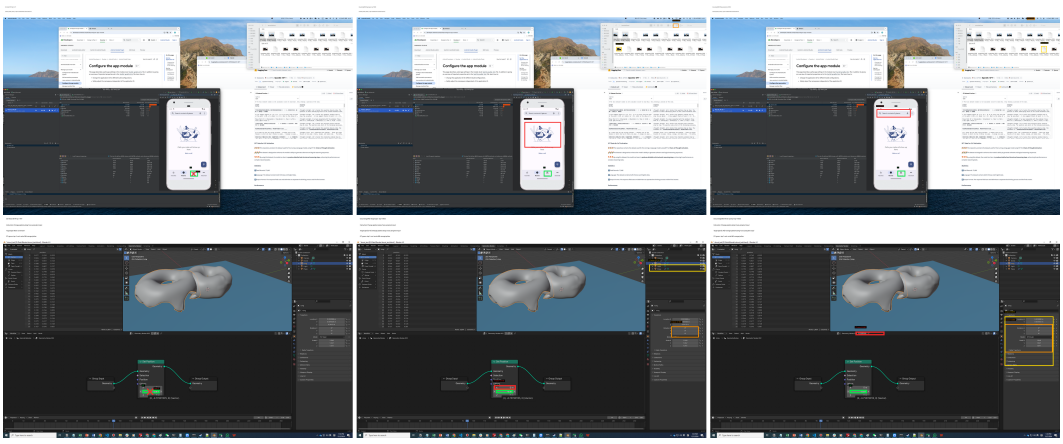


Figure 5: **Qualitative comparison of semantic target selection.** Each row shows, from left to right, INTENTLOC-GUI-3B, target-span GroundingDINO, and whole-prompt GroundingDINO. Top: the Android Studio instruction is "open contacts of the android machine," with target span "contacts of the android machine" (character offsets $[5, 36]$). Bottom: the Blender instruction is "Change position along Y axis using text input," and the target span is the complete instruction ($[0, 45]$). Green denotes the ground-truth target; red denotes the Top-1 prediction; orange/yellow denotes lower-ranked candidates.

rectly selects the Y-coordinate input field in the *Set Position* node. Target-span GroundingDINO detects a visually similar coordinate control but incorrectly selects the adjacent X-coordinate field, whereas whole-prompt GroundingDINO favors the much larger *Transform* properties panel. Notably, the target span is already the complete instruction in this example, showing that retaining all instruction tokens alone does not give GroundingDINO sufficient understanding of the relation among the action, axis, and control type.

Together, the two examples show that GroundingDINO can generate plausible GUI regions yet cannot reliably rank them according to either an implicit target object or fine-grained attribute relations. This qualitative difference is consistent with the overall Top-1 results: INTENTLOC achieves 44.7% accuracy, whereas target-span and whole-prompt GroundingDINO obtain only 3.3% and 1.4%, respectively. By using a VLM to encode the complete instruction and condition the detection queries, INTENTLOC better captures the semantic relationship between an action, its attributes, and the intended control.

INTENTLOC stays on target across resolution scales. Figure 6 compares predictions for the instruction "check my account" across four input scales. All four panels use the same original screenshot as their background and differ only in the predictions produced when each model receives the original image or a resized image with a long edge of 600, 1,000, or 1,400 pixels. The yellow rectangle marks the ground-truth account-entry region; the blue cross, orange hollow circle, and green hollow triangle denote predictions from Qwen2.5-VL-3B-Finetuned, GUI-Actor-3B, and INTENTLOC-GUI-3B, respectively. The coordinate-prediction baseline remains to the left of the target at every scale and therefore fails in all four settings. GUI-Actor similarly misses at the original and 600-pixel scales but returns to the target region at 1,000 and 1,400 pixels, yielding two failures. In contrast, INTENTLOC remains inside the account region at all four scales and makes no error.

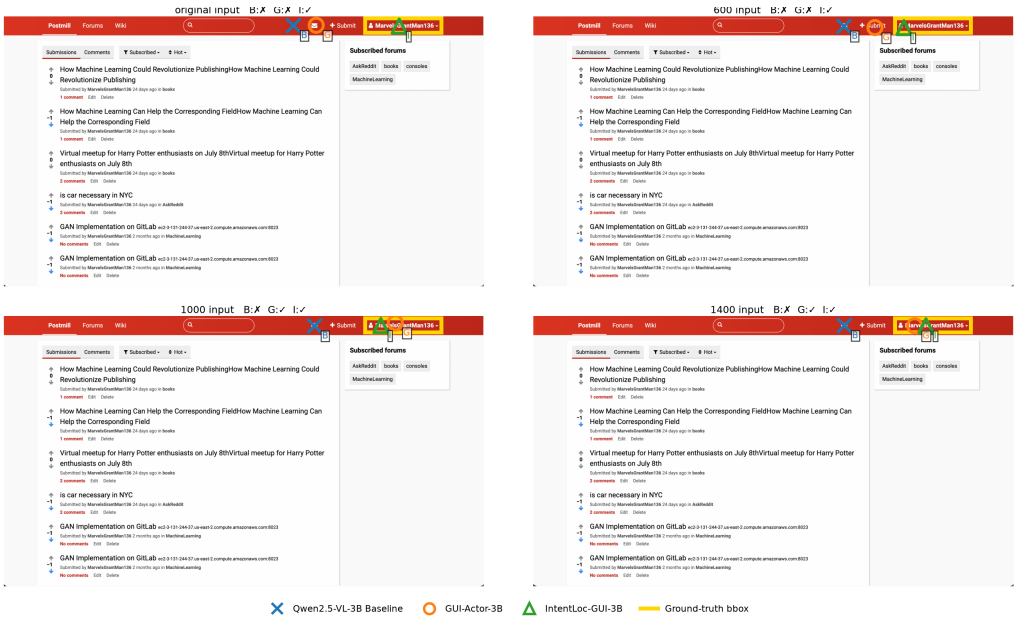


Figure 6: **Qualitative comparison of robustness to resolution scaling.** For the instruction "check my account," each panel overlays predictions produced at a different input scale onto the same original ScreenSpot-v2 screenshot. Qwen2.5-VL-3B-Finetuned, GUI-Actor-3B, and IntentLoc-GUI-3B make 4, 2, and 0 errors across the four scales, respectively.

IntentLoc is more accurate than pure-language coordinate prediction on extremely small targets. We further compare IntentLoc-GUI-3B with Qwen2.5-VL-3B-Finetuned, a matched pure-language baseline trained on the same grounding data and backbone but predicting coordinates as text tokens. Figures 7–10 present four representative examples from MMBench-GUI-L2 and OSWorld-G whose target regions occupy less than 0.01% of the screen. In every case, IntentLoc places its prediction inside the minute target region, whereas the coordinate-prediction baseline clicks a visually or spatially unrelated location. These examples complement the aggregate results in the main text and illustrate the advantage of formulating grounding as patch-level probability estimation when the target is represented by only a few visual patches. In all four figures, green denotes the ground-truth target region, the blue circle denotes the Top-1 prediction from IntentLoc-GUI-3B, and the red cross denotes the Top-1 prediction from Qwen2.5-VL-3B-Finetuned.

E Implementation, Training and Inference Details

Additional grounding data. Beyond the 1.25M GUI-Actor corpus, our professional annotators collect 0.25M further grounding examples for IntentLoc-GUI-7B. Annotation proceeds in two steps: an annotator first obtains a screenshot, and then, for that screenshot, writes a grounding instruction and annotates the ground-truth bounding box of the instructed element. These screenshots of additional grounding data cover professional, complex Chinese- and English-language software, e.g., the Adobe suite and SolidWorks. We encourage annotators to provide a mixture of intent-level and target-span-level instructions, as defined in **Appendix C**. Annotators may use advanced VLMs, such as our self-hosted Qwen3-VL-235B-A22B-Thinking (Bai et al., 2025b), to assist in drafting instructions and proposing bounding boxes, but they must manually verify every model-assisted annotation and avoid duplicate or near-duplicate examples, both within the training data and between the training data and the test sets of all evaluation benchmarks. Each completed annotation is independently checked by a second annotator, and low-quality or redundant samples are removed. We retain only one instruction per screenshot to increase data diversity. We release this data together with the benchmark. We verify that these training examples are disjoint from the test data of all evaluation benchmarks, including Atypical-GUI-Bench.

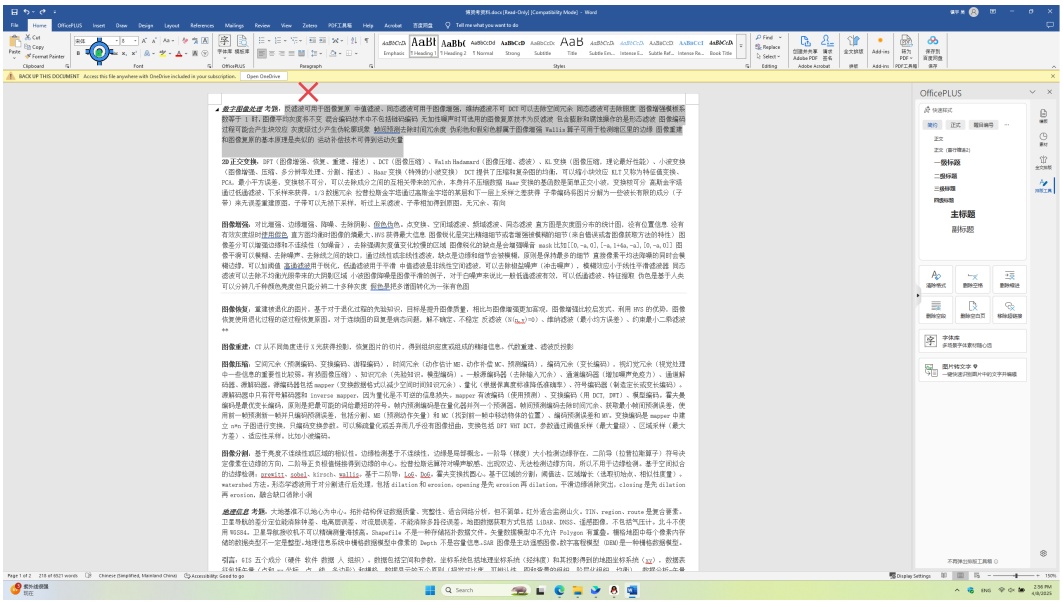


Figure 7: **Small-target comparison on MMBench-GUI-L2.** *Instruction:* "Click the underline button in the Font group of the Word ribbon." The target occupies only 0.005720% of the screen.

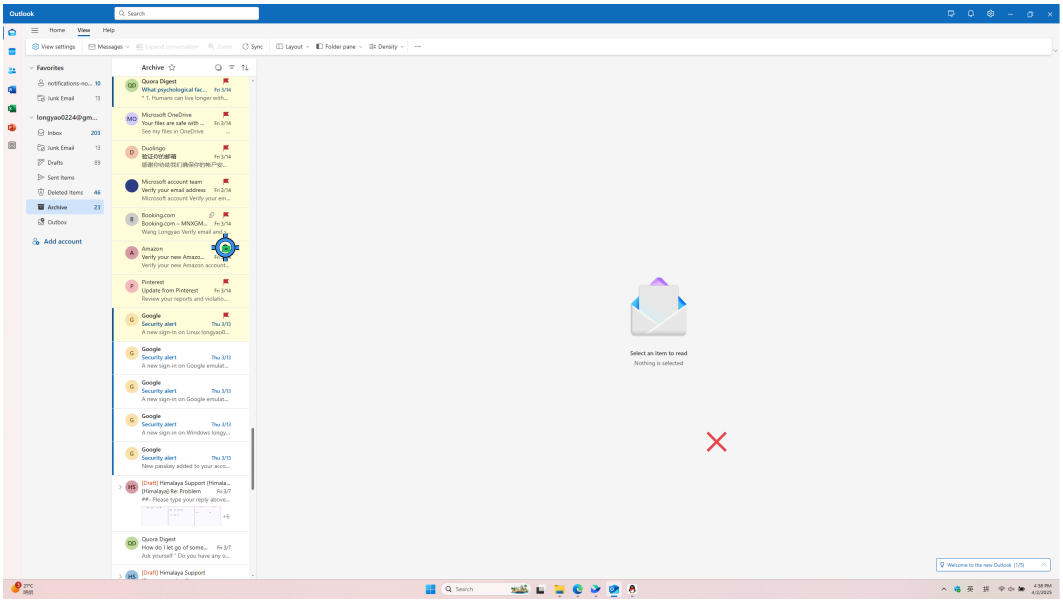


Figure 8: **Small-target comparison on MMBench-GUI-L2.** *Instruction:* "Click the red flag icon next to the Amazon email." The target occupies only 0.008112% of the screen.

Training and inference. We train in two stages: first warm-starting with the VLM frozen, then jointly optimizing the VLM backbone and the encoder-decoder parameters with different learning rates: 5e-6 for VLM backbone and 1e-6 for the encoder-decoder parameters. We use three layers for both the encoder and decoder; an ablation over different decoder depths is reported in **Appendix J**. We set crosshair loss weight $\lambda_{\text{crosshair}} = 0.2$. We initialize the learnable spatial scale to 1.0. The spatial scales at different decoder layers converge to different values close to 1.0. For dual visual

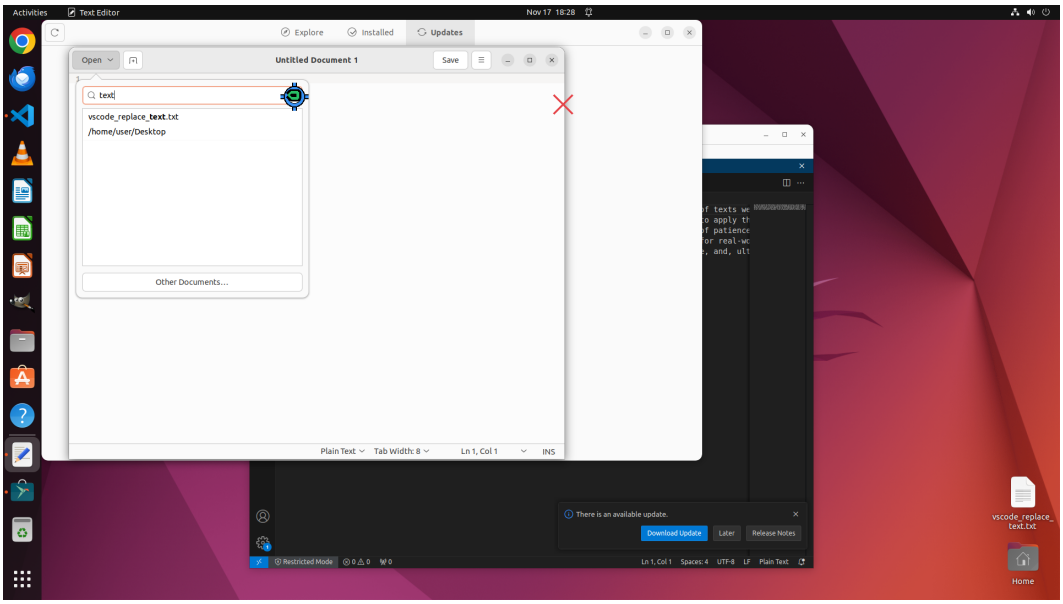


Figure 9: **Small-target comparison on OSWorld-G.** *Instruction:* "Click the icon that clears the current input." The target occupies only 0.007307% of the screen.

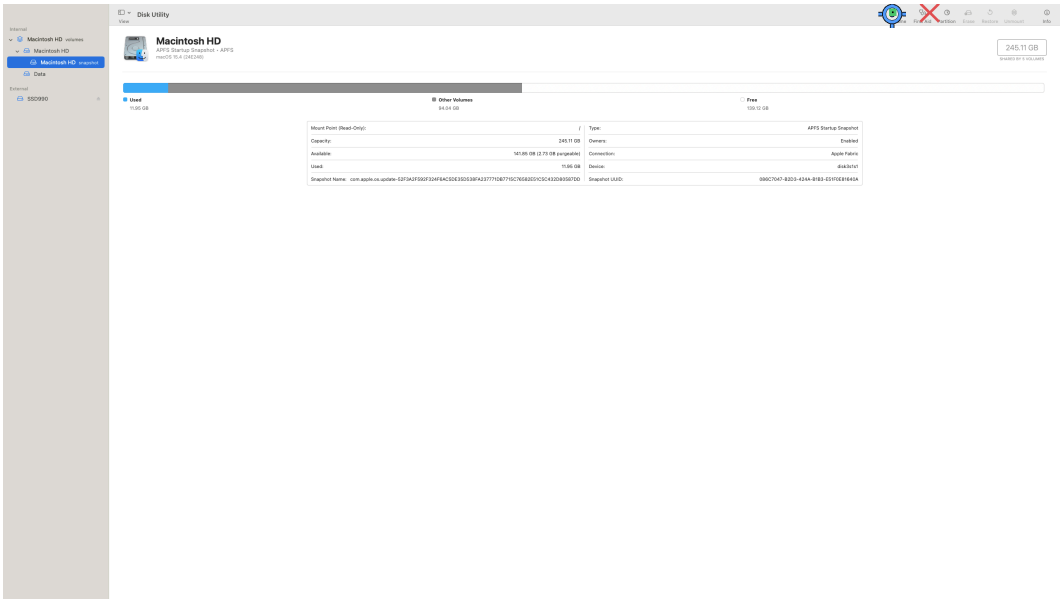


Figure 10: **Small-target comparison on MMBench-GUI-L2.** *Instruction:* "Mount the selected APFS volume." The target occupies only 0.008789% of the screen.

feature fusion, we initialize the learned scalar gate ρ to 0.5. It converges to 0.6151, assigning 61.51% weight to the raw visual features and 38.49% to the contextualized visual features.

We use the AdamW optimizer, clip the gradient norm to 1.0, set the batch size to 8, and train for one epoch. We train our model on 8 NVIDIA H20 GPUs, with the 2-stage training process taking approximately 60 hours for one model.

During inference, we set the threshold for the predicted patch distribution to 0.3 times the maximum patch probability. Specifically, all patches whose probabilities are strictly greater than $0.3 \times \max_j \hat{y}_j$ are retained and then partitioned into connected components using 4-neighborhood connectivity. For INTENTLOC-GUI-3B and INTENTLOC-GUI-7B, the patch size is 28×28 pixels (14-pixel ViT patches with 2×2 merging), consistent with the Qwen2.5-VL vision encoder, ensuring precise localization even for small UI elements such as buttons.

Unlike recent approaches such as GUI-R1 (Luo et al., 2025) and UI-R1 (Lu et al., 2026), we do not employ an additional reinforcement fine-tuning (RFT) or reinforcement learning (RL) stage. One useful property of their region-aware grounding rewards is that they do not treat a single annotated coordinate as the only correct response: multiple click points satisfying the target-region criterion can receive positive reward, alleviating the multi-solution dilemma of coordinate-token supervision. Our patch-distribution objective, which follows GUI-Actor (Wu et al., 2025a), provides a similar benefit directly during supervised fine-tuning. By assigning target probability mass to all visual patches intersecting the ground-truth region, it treats multiple valid target patches as equivalent positives instead of requiring the model to memorize one exact coordinate. INTENTLOC therefore addresses this ambiguity through its structured patch-level localization objective without relying on RFT/RL post-training.

Randomness and Seed Sensitivity. To assess inference randomness and seed sensitivity, we evaluate the same checkpoint with three random seeds (42, 123, 2026) on the complete ScreenSpot-Pro (1,581 examples) and OSWorld-G (564 examples) test sets. For each run, we seed Python's `random`, NumPy, PyTorch on the CPU, and PyTorch on all CUDA devices using `random.seed(seed)`, `np.random.seed(seed)`, `torch.manual_seed(seed)`, and `torch.cuda.manual_seed_all(seed)`, respectively. All runs use BF16 inference, batch size 1, and FlashAttention 2. Instead of NVIDIA H20-96GB GPUs, we use different machine (NVIDIA A100-SXM4-40GB GPUs). The three seeds produce identical top-1 predictions on every example, identical top-k predictions on ScreenSpot-Pro, and zero variance in all reported metrics.

This is because our inference pipeline is deterministic: the model operates in evaluation mode, which disables dropout; LLM backbone inference generates only one placeholder token, so autoregressive decoding introduces no randomness into the extracted semantic query, which is computed from the first-step hidden states; and the patch probability post-processing uses only deterministic thresholding, connected-component extraction, weighted-center computation, and sorting. Consequently, random seed has no observable effect on the results reported in this work under our evaluation setting. This empirical result does not imply strict bitwise determinism across different hardware or software environments.

F DETAILS OF ATYPICAL-GUI-BENCH

Differences between category *Multi-Window or Pane* and *Multi-Panel Design*. The distinction between the two multi-region categories is whether the visible interfaces could naturally coexist in one interactive workspace: simultaneous windows and panes belong to *multi-window or pane*, whereas different pages, temporal states, or devices assembled for presentation belong to *multi-panel design*.

Data collection and annotation. First, the benchmark annotators and authors jointly brainstorm categories of atypical and unconventional GUI screenshots commonly encountered in real-world use. Following the requirements and operational definitions of the observation categories, our benchmark annotators manually seek and collect the corresponding naturally occurring GUI data in the real world. All benchmark annotators are computer science Ph.D. students, providing the technical expertise needed to interpret atypical GUI observations and annotate interaction targets reliably. The benchmark annotators are entirely distinct from, and operate independently of, the annotators who collected the additional grounding data described in **Appendix E**; the two efforts also use different data sources and collection objectives, ensuring strict separation between the training data and Atypical-GUI-Bench.

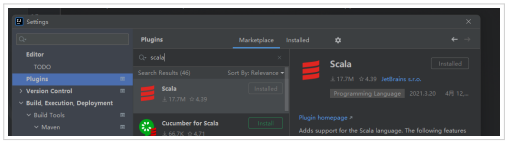
Annotation then proceeds in two steps, as for our additional training data: having obtained an image, the annotator writes a grounding instruction for it and annotates the ground-truth bounding box of

the instructed target. We encourage annotators to provide a mixture of intent-level and target-span-level instructions, as defined in **Appendix C**. Annotators may use advanced VLMs, such as our self-hosted Qwen3-VL-235B-A22B-Thinking (Bai et al., 2025b), to assist in drafting instructions and proposing bounding boxes, but they must manually verify every model-assisted annotation and avoid duplicate or near-duplicate examples, both within the benchmark and between the benchmark and our training data. Each completed annotation is independently checked by a second annotator, and low-quality or redundant samples are removed. We retain only one instruction per screenshot to increase data diversity. The authors then conduct a final round of manual inspection and filtering to remove ambiguous, incorrectly categorized, or inaccurately annotated samples, yielding 1,000 high-quality image–instruction pairs in total. To prevent evaluation leakage, we double-check that Atypical-GUI-Bench is independent of all our training data: all benchmark images, instructions, and annotations are completely excluded from the training set.

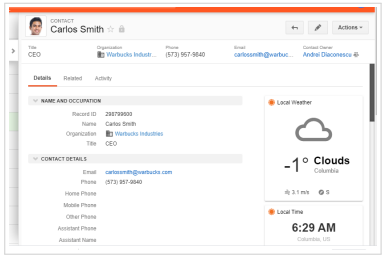
Notably, Atypical-GUI-Bench also contains many examples with small GUI targets. In categories such as *Long screenshot*, *Multi-window or pane*, and *Multi-panel design*, targets often occupy only a small fraction of the full image and consequently become very small after resizing. We therefore further stratify Atypical-GUI-Bench by target area ratio in **Appendix M** to evaluate model performance on small targets.

Qualitative examples of Atypical-GUI-Bench. Figures 11–13 present two representative examples from each Atypical-GUI-Bench observation category together with their grounding instructions. These examples retain their naturally collected image geometry and surrounding visual context; we do not artificially crop, rescale, or recompose them for benchmark construction. Together, they illustrate the distinct challenges introduced by incomplete context, unusual aspect ratios, competing windows or panels, camera artifacts, device frames, and multiple interface states.

Partial UI view. Local application regions provide insufficient global screen context and may expose only a dialog, toolbar, or settings panel.



Instruction: Click the “Installed” button next to the Scala plugin entry.



Instruction: Update Carlos Smith’s contact information to reflect his new phone number and email address.

Long screenshot. Extended pages and lists create extreme aspect ratios, long-range dependencies, and spatial dilution beyond a normal viewport.



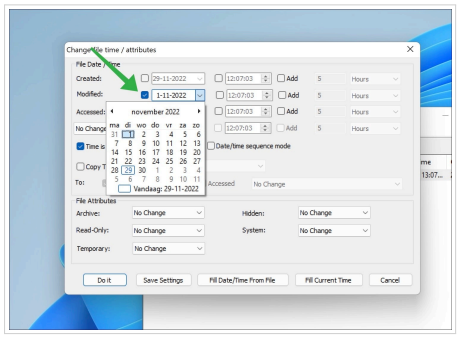
Instruction: A text link labeled “Melodifestivalen” in a vertical menu under the “Aktuella program” section, displayed in blue with a small right-pointing arrow icon.



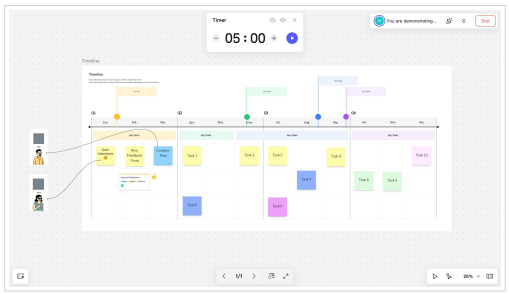
Instruction: Check the clinic’s operating schedule to plan a visit during business hours.

Figure 11: **Atypical-GUI-Bench examples for partial UI views and long screenshots.** Both original images and their corresponding grounding instructions are shown.

Multi-window or pane. Coexisting windows, dialogs, overlays, and workspace panes introduce competing interface regions and visual occlusion.

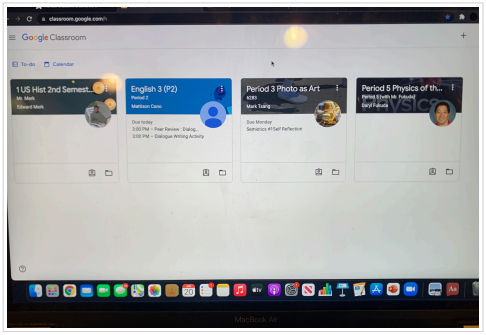


Instruction: A date input field displaying "1-11-2022" with a dropdown calendar showing November 2022, where the 29th is currently selected.



Instruction: Switch to a detailed view mode for the timeline by interacting with the view options control.

Photographed screen. Camera capture introduces perspective distortion, glare, moiré patterns, physical display boundaries, and environmental context.



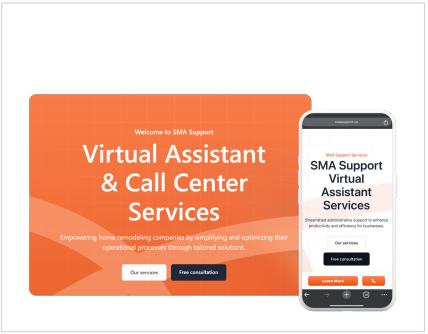
Instruction: Click the person icon at the bottom left of the "Period 3 Photo as Art" course card.



Instruction: Reset the current paint batch data to start a new production cycle.

Figure 12: **Atypical-GUI-Bench examples for multi-window or pane observations and photographed screens.** The paired examples expose complementary sources of clutter, occlusion, and capture artifacts.

Device mockup. The actionable interface is embedded in a rendered device and surrounded by substantial promotional or decorative content.

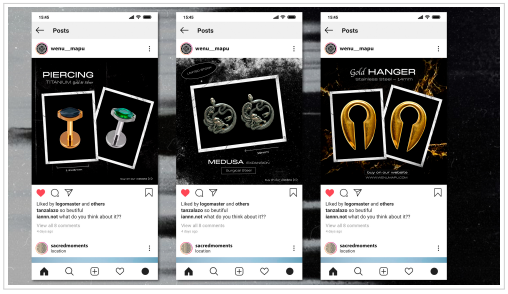


Instruction: A white rectangular button labeled "Our services" located above a black button on a mobile interface.



Instruction: A pink hexagonal icon with a white camera outline, labeled "Instagram", located in the bottom row of the app grid.

Multi-panel design. Multiple pages, temporal states, or device layouts appear on one presentation canvas, requiring panel selection before target localization.



Instruction: A heart-shaped icon located in the bottom navigation bar, filled with black color, representing the "Like" function.



Instruction: Click the center-align button in the MOVE panel to horizontally center the selected text element.

Figure 13: **Atypical-GUI-Bench examples for device mockups and multi-panel designs.** All examples preserve their original aspect ratios and naturally occurring surrounding context.

G BENCHMARKS

G.1 GROUNDING BENCHMARKS

We evaluate models' grounding accuracy on ScreenSpot-Pro (Li et al., 2025b), OSWorld-G (Xie et al., 2025), MMBench-GUI (Wang et al., 2026d), and UI-I2E-Bench (Liu et al., 2025). MMBench-GUI is a hierarchical suite covering several task levels; we evaluate only its L2 GUI element grounding subset and denote it MMBench-GUI-L2 throughout. UI-I2E-Bench specifically evaluates the grounding of small GUI elements, such as small buttons, checkboxes, and radio buttons, in high-resolution 1080p and 1440p screenshots. Across all four benchmarks, a prediction is counted as correct when the predicted point falls inside the ground-truth bounding box; for the GroundingDINO baselines, which emit multiple scored boxes, the prediction is the center of the top-1 box. Since most models already achieve over 90% accuracy on ScreenSpot (Cheng et al., 2024) and ScreenSpot-v2 (Wu et al., 2025b), indicating near-saturated performance, we do not use them for the main grounding comparison. They are still used where saturation is not a concern: the GroundingDINO pre-analysis (Section 3) and the resolution-scaling study (**Appendix N**).

ScreenSpot-Pro. ScreenSpot-Pro (Li et al., 2025b) contains 1,581 instructions paired with authentic high-resolution screenshots from 23 professional applications spanning multiple industries and operating systems, emphasizing small targets and dense, specialized interfaces.

OSWorld-G. OSWorld-G (Xie et al., 2025) comprises 564 grounding examples collected from real desktop applications and evaluates text matching, element recognition, layout understanding, and fine-grained manipulation.

MMBench-GUI. MMBench-GUI (Wang et al., 2026d) provides a hierarchical, multi-platform evaluation suite. We use its L2 GUI element grounding subset, which covers Windows, macOS, Linux, iOS, Android, and web interfaces with both basic and advanced instructions.

UI-I2E-Bench. UI-I2E-Bench (Liu et al., 2025) contains 1,477 grounding instructions collected from web, Windows, and Android interfaces. It was motivated by the observation that buttons and icons in earlier benchmarks occupy a disproportionately large fraction of the screen and can therefore overestimate grounding performance. Its screenshots and element scales better reflect typical 1080p and 1440p displays, making it particularly useful for evaluating spatial dilution at high resolution and localization of small buttons and toggle controls, including checkboxes and radio buttons. The benchmark also balances a broader set of long-tail element types and labels instructions as explicit or implicit, enabling diagnosis of scale sensitivity across element types and instruction complexity. Whereas ScreenSpot-Pro emphasizes professional software, very high-resolution screenshots, and extremely small targets, UI-I2E-Bench emphasizes realistic everyday display scales, long-tail controls, and diverse instructions.

G.2 Online Benchmarks

OSWorld. OSWorld (Xie et al., 2024) evaluates agents on open-ended tasks across real computer applications and operating-system workflows.

AndroidWorld. AndroidWorld (Rawles et al., 2025) provides reproducible tasks executed in live Android applications, testing multi-step mobile interaction.

MobileWorld. MobileWorld (Kong et al., 2026) evaluates mobile agents on diverse real-world tasks that require planning, grounding, and sequential action execution. We report task success rate on all three online benchmarks.

H Detailed Grounding Benchmark Results

We provide complete category-level results for ScreenSpot-Pro (Table 7), OSWorld-G (Table 8), and MMBench-GUI-L2 (Table 9) below. For UI-I2E-Bench (Table 10), baseline numbers are taken from the official leaderboard (Microsoft Research, 2026) and from published evaluations. Neither source reports a per-element-type breakdown for the most recent baselines (GUI-Actor-7B, GTA1, UI-Ins), nor any fine-grained breakdown for GPT-4o, Qwen2-VL-7B, and Qwen3-VL-8B; those cells are therefore left as dashes rather than filled with unverified numbers.

Table 7: **Performance comparison on *ScreenSpot-Pro* (Li et al., 2025b).** We highlight in bold the best-performing results among open-source models at each parameter scale. We also report in the table the improvement of our model over the underlined second-best baseline.

Model	Backbone	Dev	Creative	CAD	Scientific	Office	OS	Avg
Proprietary Model								
GPT-4o (OpenAI, 2024)	–	0.7	0.6	1.5	1.2	0.9	0.0	0.8
Claude Computer Use (Anthropic, 2024)	–	12.6	16.8	11.9	25.8	26.9	8.1	17.1
Claude 3.7 Sonnet (Anthropic, 2025a)	–	–	–	–	–	–	–	27.7
Operator (OpenAI, 2025)	–	–	–	–	–	–	–	36.6
Gemini-2.5-Pro (Comanici et al., 2025)	–	–	–	–	–	–	–	11.4
Open-Source Model (About 3B Parameter Size)								
GroundingDINO-Finetuned (Liu et al., 2024a)	–	3.3	3.9	4.7	1.8	2.2	3.6	3.3
GroundingDINO-Finetuned + OCR-Reranker	–	10.4	11.1	3.5	13.4	22.2	10.2	11.6
OS-Atlas-4B (Wu et al., 2025b)	InternVL2-4B	3.7	2.3	1.5	7.5	4.8	3.1	3.7
ShowUI-2B (Lin et al., 2025)	Qwen2-VL-2B	9.4	5.3	7.9	10.6	13.5	6.6	7.7
Qwen2.5-VL-3B (Bai et al., 2025a)	–	12.0	16.4	7.3	24.8	29.6	6.1	16.1
UGround-V1-2B (Gou et al., 2025)	Qwen2-VL-2B	27.4	26.7	14.6	34.3	38.3	17.9	26.6
UI-TARS-2B (Qin et al., 2025)	Qwen2-VL-2B	26.4	27.6	14.6	39.8	42.6	14.3	27.7
Qwen2.5-VL-3B-Finetuned	Qwen2.5-VL-3B	27.4	27.0	27.6	32.3	53.0	21.4	31.1
GUI-Actor-3B (Wu et al., 2025a)	Qwen2.5-VL-3B	39.8	36.7	34.1	49.6	61.3	35.2	42.2
INTENTLOC-GUI-3B (GUI-Actor Data)	Qwen2.5-VL-3B	**45.5**	**39.0**	**34.5**	**50.4**	**63.0**	**37.8**	**44.7** (2.5 ↑)
Open-Source Model (About 7B Parameter Size)								
Qwen2-VL-7B (Wang et al., 2024a)	–	1.3	0.9	0.4	3.5	3.0	0.5	1.6
SeeClick-9.6B (Cheng et al., 2024)	Qwen-VL-Chat	0.3	0.6	1.9	2.0	0.9	1.5	1.1
Aria-UI (Yang et al., 2025)	Aria	8.4	14.7	6.1	18.1	16.1	2.6	11.3
OS-Atlas-7B (Wu et al., 2025b)	Qwen2-VL-7B	17.7	17.9	10.3	24.4	27.4	16.8	18.9
Aguvis-7B (Xu et al., 2025)	Qwen2-VL-7B	16.1	21.4	13.8	34.6	34.3	19.4	22.9
Qwen2.5-VL-7B (Bai et al., 2025a)	–	26.1	24.0	13.0	31.1	45.2	23.5	26.8
UGround-V1-7B (Gou et al., 2025)	Qwen2-VL-7B	28.1	31.7	14.6	39.0	49.6	24.5	31.1
UI-TARS-7B (Qin et al., 2025)	Qwen2-VL-7B	36.1	32.8	18.0	50.0	53.5	24.5	35.7
JEDI-7B (Xie et al., 2025)	Qwen2.5-VL-7B	27.4	34.0	32.2	52.4	68.7	26.0	39.5
Qwen2.5-VL-7B-Finetuned	Qwen2.5-VL-7B	44.2	36.1	36.4	43.7	59.6	37.2	42.4
GUI-Actor-7B (Wu et al., 2025a)	Qwen2.5-VL-7B	38.1	41.4	38.3	50.8	63.0	38.8	44.6
INTENTLOC-GUI-7B (GUI-Actor Data)	Qwen2.5-VL-7B	44.8	44.9	43.7	53.9	66.5	44.4	49.2
UI-TARS-1.5-7B (ByteDance Seed, 2025)	Qwen2.5-VL-7B	–	–	–	–	–	–	49.6
GTA1-7B (Yang et al., 2026)	Qwen2.5-VL-7B	44.5	44.0	44.4	57.1	75.2	38.3	50.1
UI-Venus-Ground-7B (Gu et al., 2025)	Qwen2.5-VL-7B	50.2	42.8	51.0	57.1	67.8	37.2	50.8
UI-Ins-7B (Chen et al., 2026)	Qwen2.5-VL-7B	47.8	45.8	50.9	58.7	70.0	42.8	52.2
Qwen3-VL-8B (Bai et al., 2025b)	–	51.8	**49.6**	46.4	57.5	72.2	**53.1**	54.5
GUI-Owl-7B (Ye et al., 2025)	Qwen2.5-VL-7B	54.5	46.1	54.1	61.0	68.7	47.9	54.9
Qwen3-VL-8B-Finetuned (GUI-Actor Data)	Qwen3-VL-8B	**55.2**	48.4	51.3	59.8	74.4	52.6	56.3
INTENTLOC-GUI-7B (GUI-Actor Data + 0.25M Data)	Qwen2.5-VL-7B	52.8	**49.6**	**58.6**	**62.6**	**77.0**	52.6	**58.1** (1.8 ↑)

Table 8: **Performance comparison on *OSWorld-G* (Xie et al., 2025).** We also report in the table the improvement of our model over the underlined second-best baseline.

Model	Backbone	Text Matching	Element Recognition	Layout Understanding	Fine-grained Manipulation	Overall
Proprietary Model						
Operator (OpenAI, 2025)	–	51.3	42.4	46.6	31.5	40.6
Gemini-2.5-Pro (Comanici et al., 2025)	–	59.8	45.5	49.0	33.6	45.2
Open-Source Model (About 3B Parameter Size)						
GroundingDINO-Finetuned (Liu et al., 2024a)	–	5.8	5.5	5.5	2.7	4.3
GroundingDINO-Finetuned + OCR-Reranker	–	18.4	10.6	14.2	5.4	10.1
Qwen2.5-VL-3B (Bai et al., 2025a)	–	41.4	28.8	34.8	13.4	27.3
Qwen2.5-VL-3B-Finetuned	Qwen2.5-VL-3B	43.0	40.9	37.8	20.1	34.8
GUI-Actor-3B (Wu et al., 2025a)	Qwen2.5-VL-3B	53.3	45.5	46.6	27.5	40.3
INTENTLOC-GUI-3B (GUI-Actor Data)	Qwen2.5-VL-3B	**60.9**	**50.6**	**53.0**	**29.5**	**45.7** (5.4 ↑)
Open-Source Model (About 7B Parameter Size)						
OS-Atlas-7B (Wu et al., 2025b)	Qwen2-VL-7B	44.1	29.4	35.2	16.8	27.7
Qwen2.5-VL-7B (Bai et al., 2025a)	–	45.6	32.7	41.9	18.1	31.4
UGround-V1-7B (Gou et al., 2025)	Qwen2-VL-7B	51.3	40.3	43.5	24.8	36.4
Aguvis-7B (Xu et al., 2025)	Qwen2-VL-7B	55.9	41.2	43.9	28.2	38.7
GUI-Actor-7B (Wu et al., 2025a)	Qwen2.5-VL-7B	55.9	47.0	47.0	26.2	42.2
Qwen2.5-VL-7B-Finetuned	Qwen2.5-VL-7B	60.9	51.8	54.6	30.2	47.5
UI-TARS-7B (Qin et al., 2025)	Qwen2-VL-7B	60.2	51.8	54.9	35.6	47.5
INTENTLOC-GUI-7B (GUI-Actor Data)	Qwen2.5-VL-7B	66.3	57.0	58.9	36.2	51.8
UI-TARS-1.5-7B (ByteDance Seed, 2025)	Qwen2.5-VL-7B	–	–	–	–	52.8
JEDI-7B (Xie et al., 2025)	Qwen2.5-VL-7B	65.9	55.5	57.7	46.9	54.1
UI-Venus-Ground-7B (Gu et al., 2025)	Qwen2.5-VL-7B	–	–	–	–	54.6
GTA1-7B (Yang et al., 2026)	Qwen2.5-VL-7B	–	–	–	–	55.1
Qwen3-VL-8B (Bai et al., 2025b)	–	67.4	59.4	60.9	45.6	55.3
GUI-Owl-7B (Ye et al., 2025)	Qwen2.5-VL-7B	64.8	**63.6**	61.3	41.0	55.9
Qwen3-VL-8B-Finetuned (GUI-Actor Data)	Qwen3-VL-8B	67.4	61.2	61.7	**46.3**	55.9
INTENTLOC-GUI-7B (GUI-Actor Data + 0.25M Data)	Qwen2.5-VL-7B	**67.8**	62.7	**62.9**	45.0	**56.4** (0.5 ↑)

Table 9: **Performance comparison on *MMBench-GUI-L2* (GUI element grounding) (Wang et al., 2026d).** We also report in the table the improvement of our model over the underlined second-best baseline.

Model	Backbone	Windows		MacOS		Linux		iOS		Android		Web		Avg
		Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	
Proprietary Model														
GPT-4o (OpenAI, 2024)	–	1.48	1.10	8.69	4.34	1.05	1.02	5.10	3.33	2.53	1.41	3.23	2.92	2.87
Claude 3.7 Sonnet (Anthropic, 2025a)	–	1.48	0.74	12.46	7.51	1.05	0.00	13.69	10.61	1.40	1.40	3.23	2.27	4.66
Qwen-Max-VL (Bai et al., 2023)	–	43.91	36.76	58.84	56.07	53.93	30.10	77.39	59.09	79.49	70.14	74.84	58.77	58.03
Open-Source Model (About 3B Parameter Size)														
GroundingDINO-Finetuned (Liu et al., 2024a)	–	13.28	2.57	9.57	1.73	8.38	3.06	35.99	8.79	31.18	8.17	21.94	4.55	13.07
GroundingDINO-Finetuned + OCR-Reranker	–	15.13	4.41	12.17	3.76	9.95	1.53	34.39	8.48	23.60	6.20	25.48	6.49	13.15
ShowUI-2B (Lin et al., 2025)	Qwen2-VL-2B	9.23	4.41	24.06	10.40	25.13	11.73	28.98	19.70	17.42	8.73	22.90	12.66	15.96
Qwen2.5-VL-3B (Bai et al., 2025a)	–	25.09	8.46	31.59	11.85	28.80	12.24	60.83	33.64	61.52	33.52	48.39	22.40	32.87
Qwen2.5-VL-3B-Finetuned	Qwen2.5-VL-3B	38.38	25.37	48.41	34.97	41.88	29.59	70.70	54.85	77.53	58.03	63.23	43.83	50.54
GUI-Actor-3B (Wu et al., 2025a)	Qwen2.5-VL-3B	**81.18**	48.16	**79.71**	54.05	67.02	**42.86**	**93.95**	73.94	89.89	74.37	90.97	66.88	73.41
IntentLoc-GUI-3B (GUI-Actor Data)	Qwen2.5-VL-3B	78.97	**52.94**	**79.71**	**56.94**	**66.49**	41.33	**93.95**	**76.97**	**92.98**	**78.31**	**92.58**	**69.48**	**75.10** (1.69↑)
Open-Source Model (About 7B Parameter Size)														
Qwen2.5-VL-7B (Bai et al., 2025a)	–	31.37	16.54	31.30	21.97	21.47	12.24	66.56	55.15	35.11	35.21	40.32	32.47	33.85
Aguvis-7B (Xu et al., 2025)	Qwen2-VL-7B	37.27	21.69	48.12	33.27	33.51	25.00	67.52	65.15	60.96	50.99	61.61	45.45	45.66
OS-Atlas-7B (Wu et al., 2025b)	Qwen2-VL-7B	36.90	18.75	44.35	21.68	31.41	13.27	74.84	48.79	69.60	46.76	61.29	35.39	41.42
UGround-V1-7B (Gou et al., 2025)	Qwen2-VL-7B	66.79	38.97	71.30	48.55	56.54	31.12	92.68	70.91	93.54	70.99	88.71	64.61	65.68
UI-TARS-1.5-7B (ByteDance Seed, 2025)	Qwen2.5-VL-7B	68.27	38.97	68.99	44.51	64.40	37.76	88.54	69.39	90.45	69.29	80.97	56.49	64.32
Qwen2.5-VL-7B-Finetuned	Qwen2.5-VL-7B	77.12	51.10	78.84	57.23	67.54	51.02	86.62	69.09	88.76	76.62	89.68	67.21	72.95
JEDI-7B (Xie et al., 2025)	Qwen2.5-VL-7B	–	–	–	–	–	–	–	–	–	–	–	–	75.10
GTA1-7B (Yang et al., 2026)	Qwen2.5-VL-7B	–	–	–	–	–	–	–	–	–	–	–	–	78.50
GUI-Actor-7B (Wu et al., 2025a)	Qwen2.5-VL-7B	80.8	55.1	81.4	60.4	64.9	41.8	94.3	82.7	93.5	79.7	89.7	72.1	76.5
IntentLoc-GUI-7B (GUI-Actor Data)	Qwen2.5-VL-7B	83.76	59.56	86.09	70.23	72.77	57.65	94.27	82.12	95.51	82.54	94.52	78.25	81.15
Qwen3-VL-8B (Bai et al., 2025b)	–	81.92	65.07	84.06	68.79	73.30	54.59	90.76	79.39	93.82	82.82	87.74	72.40	79.17
UI-Venus-Ground-7B (Gu et al., 2025)	Qwen2.5-VL-7B	–	–	–	–	–	–	–	–	–	–	–	–	79.90
GUI-Owl-7B (Ye et al., 2025)	Qwen2.5-VL-7B	86.35	61.76	81.74	64.45	74.35	**61.73**	94.90	83.03	95.78	83.66	93.22	72.72	80.49
Qwen3-VL-8B-Finetuned (GUI-Actor Data)	Qwen3-VL-8B	**89.30**	63.24	84.35	70.23	72.77	59.69	94.90	86.36	94.10	85.15	93.87	78.25	82.43
UI-Ins-7B (Chen et al., 2026)	Qwen2.5-VL-7B	–	–	–	–	–	–	–	–	–	–	–	–	83.10
IntentLoc-GUI-7B (GUI-Actor Data + 0.25M Data)	Qwen2.5-VL-7B	87.08	**67.65**	**88.99**	**76.30**	**79.06**	**61.73**	**96.18**	**87.58**	**97.47**	**88.17**	**97.10**	**86.04**	**85.73** (2.63↑)

Table 10: **Performance comparison on *UI-I2E-Bench* (Liu et al., 2025).** Public baseline results are taken from the official leaderboard (Microsoft Research, 2026) and additional published evaluations (Li et al., 2025c; Chen et al., 2025; 2026; Wang et al., 2026b). The fine-grained columns follow the benchmark's diagnostic dimensions. We bold the best available result in each parameter-scale group; dashes mark unavailable results. Arrows give the improvement of our model over the underlined second-best baseline.

Model	**Backbone**	**Instruction**		**Platform**			**Element Type**					**Avg.**
		Explicit	Implicit	Web	Desktop	Mobile	Button	Icon	Dropdown	Input	Toggle	
Proprietary Model												
GPT-4o (OpenAI, 2024)	–	–	–	–	–	–	–	–	–	–	–	15.2
Open-Source Model (About 3B Parameter Size)												
GroundingDINO-Finetuned (Liu et al., 2024a)	–	12.9	11.2	13.8	7.3	14.5	12.3	11.5	15.2	20.8	4.0	11.9
GroundingDINO-Finetuned + OCR-Reranker	–	27.9	9.8	17.0	11.0	20.7	20.1	8.0	14.6	28.7	15.8	16.7
OS-Atlas-4B (Wu et al., 2025b)	InternVL2-4B	51.5	39.9	54.6	19.9	58.6	43.5	44.1	46.6	46.3	42.2	44.3
ShowUI-2B (Lin et al., 2025)	Qwen2-VL-2B	51.3	35.6	29.6	30.4	53.9	52.0	44.1	51.1	52.8	18.9	41.5
Qwen2.5-VL-3B (Bai et al., 2025a)	–	51.4	35.8	39.9	38.7	44.5	48.3	46.9	69.7	29.0	32.0	41.7
UGround-V1-2B (Gou et al., 2025)	Qwen2-VL-2B	72.9	47.9	66.4	49.5	59.9	57.6	50.7	82.0	64.8	44.7	57.4
UI-TARS-2B (Qin et al., 2025)	Qwen2-VL-2B	**74.1**	54.5	62.2	54.0	66.7	59.1	55.6	82.6	**72.7**	**50.1**	62.0
Qwen2.5-VL-3B-Finetuned	Qwen2.5-VL-3B	65.5	55.4	56.9	48.9	67.7	68.0	59.1	84.8	69.8	33.3	59.2
GUI-Actor-3B (Wu et al., 2025a)	Qwen2.5-VL-3B	70.3	59.7	65.3	58.2	67.2	74.0	67.1	84.3	66.9	42.6	63.7
IntentLoc-GUI-3B (GUI-Actor Data)	Qwen2.5-VL-3B	70.4	**65.1**	**67.6**	**61.3**	**71.2**	**77.3**	**74.1**	**87.1**	69.8	44.2	**67.1** (3.4↑)
Open-Source Model (About 7B Parameter Size)												
Qwen2-VL-7B (Wang et al., 2024a)	–	53.8	45.6	–	–	–	–	–	–	–	–	48.7
SeeClick-9.6B (Cheng et al., 2024)	Qwen-VL-Chat	37.1	19.9	18.2	15.8	37.2	31.6	26.2	22.5	29.6	22.1	26.4
OS-Atlas-7B (Wu et al., 2025b)	Qwen2-VL-7B	63.2	55.8	52.2	48.9	68.1	69.1	58.7	80.3	70.1	32.3	58.6
Aguvis-7B (Xu et al., 2025)	Qwen2-VL-7B	61.1	48.4	45.1	47.6	60.3	60.2	56.3	74.2	54.8	35.7	53.2
Qwen2.5-VL-7B (Bai et al., 2025a)	–	58.4	51.0	56.9	41.6	61.7	59.5	59.4	74.7	42.8	46.2	53.8
Qwen2.5-VL-7B-Finetuned	Qwen2.5-VL-7B	67.9	63.0	65.6	59.5	68.5	70.3	70.3	86.5	59.2	52.6	64.9
UGround-V1-7B (Gou et al., 2025)	Qwen2-VL-7B	81.3	63.6	70.8	65.7	73.5	72.9	62.9	83.7	75.4	63.5	70.3
UI-TARS-7B (Qin et al., 2025)	Qwen2-VL-7B	71.4	55.3	56.5	58.0	65.7	66.5	63.3	75.3	60.4	51.4	61.4
GUI-Actor-7B (Wu et al., 2025a)	Qwen2.5-VL-7B	71.6	66.1	65.2	63.2	72.9	–	–	–	–	–	68.2
IntentLoc-GUI-7B (GUI-Actor Data)	Qwen2.5-VL-7B	75.9	72.2	72.3	70.3	76.5	83.6	78.0	84.3	77.1	56.1	73.6
UI-TARS-1.5-7B (ByteDance Seed, 2025)	Qwen2.5-VL-7B	81.3	68.2	79.5	68.8	74.1	76.6	71.7	82.0	75.3	66.3	73.2
Qwen3-VL-8B (Bai et al., 2025b)	–	–	–	–	–	–	–	–	–	–	–	76.5
Qwen3-VL-8B-Finetuned (GUI-Actor Data)	Qwen3-VL-8B	82.7	75.4	79.4	76.7	78.7	82.9	78.0	91.0	83.3	65.0	78.1
GTA1-7B (Yang et al., 2026)	Qwen2.5-VL-7B	87.0	72.8	77.5	71.3	83.5	–	–	–	–	–	78.2
UI-Ins-7B (Chen et al., 2026)	Qwen2.5-VL-7B	88.9	76.3	90.5	72.8	**83.8**	–	–	–	–	–	81.1
IntentLoc-GUI-7B (GUI-Actor Data + 0.25M Data)	Qwen2.5-VL-7B	**90.7**	**80.6**	**95.2**	**80.4**	83.6	**83.6**	**81.1**	**93.8**	**84.2**	**83.4**	**84.4** (3.3↑)

I Online Benchmark Results with a Different Planner

The OSWorld column of Table 3 uses Qwen3-VL-235B-A22B-Thinking (Bai et al., 2025b) as the planner. Since the planner and the executor jointly determine the success rate, a single planner leaves open whether the ranking among executors reflects the executors themselves or an interaction with that particular planner. We therefore repeat the OSWorld comparison with Claude-Haiku-4.5 (Anthropic, 2025b) as the planner, holding the agentic framework, the action space, and the evaluation protocol fixed and swapping only the executor.

Table 11 reports the results. IntentLoc-GUI-7B again ranks first, ahead of UI-Ins-7B by 1.4 and GUI-Actor-7B by 2.0 points. The ordering of the three executors is the same as in Table 3, so it is not an artifact of the planner used in the main text. The margins between executors are narrower here, which is consistent with a stronger planner emitting subgoals that are easier to ground. Absolute values are comparable only within a planner group, not across Tables 3 and 11.

Table 11: **Success rates on OSWorld with Claude-Haiku-4.5 as the planner.** Only the executor differs across the agentic-framework rows, and we highlight in bold the best result among them. The end-to-end row is quoted from the model announcement (Anthropic, 2025b).

Category	Model	OSWorld
End-to-End Model	Claude-Haiku-4.5 (Anthropic, 2025b)	50.7
Agentic Framework	Planner + UI-Ins-7B (Chen et al., 2026)	56.9
	Planner + GUI-Actor-7B (Wu et al., 2025a)	56.3
	Planner + IntentLoc-GUI-7B (GUI-Actor Data + 0.25M Data)	**58.3**

J Additional Ablation Studies and Component Analysis

Ablation on decoder layer number. Table 12 reports ablations over the number of decoder layers, the inference-time crosshair bias, and layer-wise crosshair placement. Three decoder layers achieve the best overall performance, while shallower or deeper variants yield modestly lower accuracy on most benchmarks. With three layers in both the encoder and decoder, the encoder–decoder module contains approximately 500M parameters, providing the best balance between accuracy and efficiency. We therefore use three layers for both the encoder and decoder in all main experiments.

Table 12: **Additional Ablation Studies.** We highlight in bold the best-performing results. We also report the performance change of each ablation variant compared with the corresponding full-design model (IntentLoc-GUI-7B trained on GUI-Actor and 0.25M Data). All other settings are identical.

Ablated Component	Component Value	ScreenSpot-Pro	OSWorld-G	UI-I2E-Bench
Decoder Layers Num	1 Layer	54.2 (3.9↓)	52.4 (4.0↓)	81.8 (2.6↓)
	2 Layers	56.7 (1.4↓)	55.9 (0.5↓)	83.1 (1.3↓)
	3 Layers	**58.1**	**56.4**	**84.4**
	4 Layers	56.3 (1.8↓)	56.1 (0.3↓)	**84.4**
Crosshair Bias at Inference	Removed	13.1 (45.0↓)	7.0 (49.4↓)	22.4 (62.0↓)
Crosshair Placement	First Decoder Layer Only	56.4 (1.7↓)	55.0 (1.4↓)	82.7 (1.7↓)

Direct spatial supervision on the crosshair is necessary. We further investigate which component primarily accounts for the improvement from Crosshair Attention. In Table 4 of the main paper, the *Vanilla Cross-Attention* variant removes both the crosshair prediction branch from the decoder and the crosshair loss, namely the L1 distance between each predicted crosshair and the target location. In contrast, the $\lambda_{\text{crosshair}} = 0.0$ variant retains the crosshair prediction branch but removes its direct spatial supervision, i.e., the crosshair loss. Both variants suffer substantial performance drops relative to the full-design model, and retaining the unsupervised prediction branch provides almost no improvement over vanilla cross-attention. These results show that direct spatial supervision is necessary for learning an effective spatial bias: it guides the layer-wise attention maps to progres-

sively focus on the target region. Direct spatial supervision therefore turns the crosshair from an unconstrained latent variable into a learned spatial prior.

Inference-time removal of the crosshair bias. To test whether the trained decoder actually uses the crosshair bias, we take the full model trained with Crosshair Attention and remove the crosshair bias only at inference. As shown in Table 12, accuracy collapses to 13.1%, 7.0%, and 22.4% on ScreenSpot-Pro, OSWorld-G, and UI-I2E-Bench, respectively, far below the vanilla cross-attention model (54.4%, 53.2%, and 78.6%; Table 4). The trained decoder therefore relies on the crosshair bias at test time rather than merely treating the crosshair as an auxiliary training signal. Together with the $\lambda_{\text{crosshair}} = 0$ result, this indicates that the spatial bias and its direct supervision work jointly.

Layer-wise crosshairs outperform a single crosshair. We additionally keep the decoder depth at three but introduce the crosshair only in the first decoder layer. This re-trained variant achieves 56.4%, 55.0%, and 82.7% accuracy on ScreenSpot-Pro, OSWorld-G, and UI-I2E-Bench, respectively, corresponding to drops of 1.7, 1.4, and 1.7 percentage points relative to the full-design model. These results further demonstrate that layer-wise crosshairs are more effective than introducing a crosshair at only one decoder layer.

K Convergence and Visualizations of Crosshairs

Convergence of Crosshairs: Qualitative analysis. We visualize the spatial hypotheses generated by the three layers of the Crosshair Attention decoder in two representative successful cases. In Figures 14 and 15, C1, C2, and C3 denote the crosshairs predicted by the first, second, and third decoder layers, respectively; P denotes the final prediction, and the green rectangle and GT marker denote the ground-truth bounding box and its center. The yellow trajectory connects C1→C2→C3→P. Distances below are Euclidean distances to the GT center in normalized screen coordinates.

For the Excel example, whose instruction is "search for excel functionalities," the crosshair-to-GT distance decreases from 0.4670 at C1 to 0.3214 at C2 and 0.0482 at C3, before reaching 0.0364 at the final prediction. The first two layers form coarse hypotheses over the upper portion of the interface, while the third layer moves close to the target search control in the upper-right corner. For the Unreal Engine example, with the instruction "stop level demo," the corresponding distance decreases from 0.4063 to 0.3133 and 0.0434, and then to 0.0034. Here, the trajectory moves from the central viewport toward the small stop control in the top toolbar.

These examples illustrate the intended coarse-to-fine behavior of Crosshair Attention: early layers establish an approximate instruction-conditioned region, while later layers incorporate visual evidence to refine the spatial hypothesis toward the target. The final prediction then makes a smaller correction from the last crosshair. Although these selected examples are qualitative rather than an aggregate convergence measurement, their monotonic trajectories show that the intermediate crosshairs can serve as interpretable stages of the decoder's localization process.

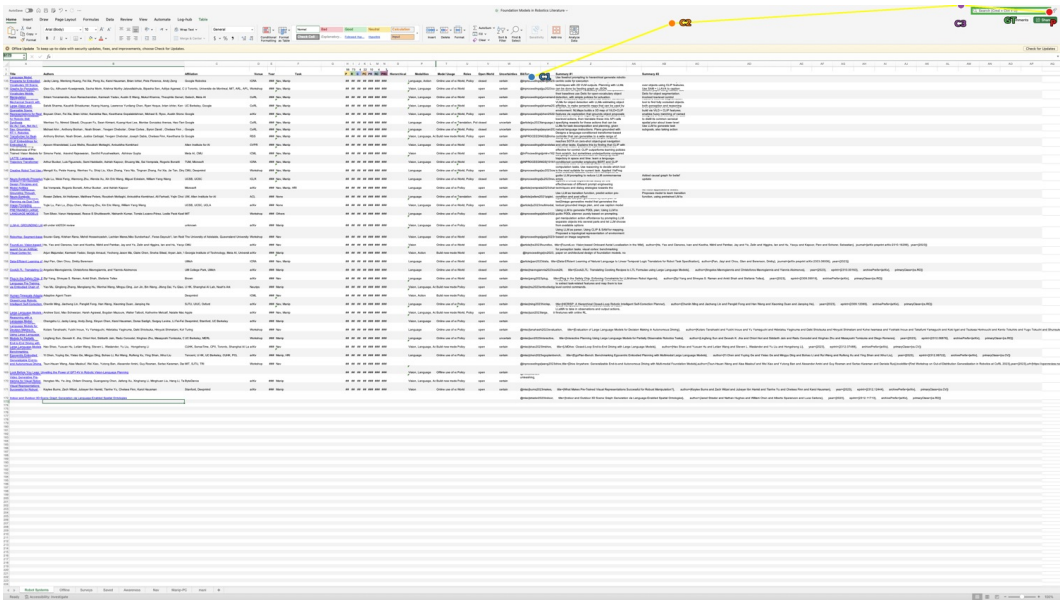


Figure 14: **Layer-wise convergence of crosshairs toward the target: Excel on macOS**, with the instruction "search for excel functionalities." Blue, orange, and purple markers indicate the crosshairs from decoder layers 1–3 (C1–C3), red indicates the final prediction (P), and green indicates the ground-truth target (GT), including its bounding box. The yellow polyline traces the progressive localization trajectory.



Figure 15: **Layer-wise convergence of crosshairs toward the target: Unreal Engine on Windows**, with the instruction "stop level demo." Markers follow Figure 14.

Convergence of Crosshairs: Quantitative analysis. Because ScreenSpot-Pro screenshots have different dimensions, we measure distance in a normalized image coordinate system: the horizontal and vertical coordinates are independently divided by the image width and height, respectively, so

both axes lie in $[0, 1]$. We then compute the Euclidean distance from each crosshair or final prediction to the ground-truth bounding-box center in this coordinate system. The resulting distance is dimensionless rather than measured in pixels.

Table 13: **Distance to the ground-truth bounding-box center across decoder stages on ScreenSpot-Pro.** The two constant rows always predict the same point: the screen center, or the mean target location $(0.377, 0.358)$ computed on the ScreenSpot-Pro test set itself, which is therefore an optimistic reference. All distances are dimensionless Euclidean distances in normalized image coordinates.

Stage	Mean	Median	P25	P75	P90
Constant: screen center	0.3771	0.3956	0.2289	0.5259	0.6146
Constant: mean target location	0.3389	0.3352	0.2285	0.4295	0.5301
R1	0.3905	0.3762	0.2070	0.5289	0.6770
R2	0.2929	0.2485	0.1598	0.3768	0.5423
R3	0.2710	0.2392	0.1618	0.3361	0.4798
Final	0.1480	0.0255	0.0069	0.1455	0.5154

As shown in Table 13 and Figure 16, the mean distance decreases monotonically from 0.3905 at R1 to 0.2929 at R2, 0.2710 at R3, and 0.1480 at the final prediction. The first crosshair is no better than a constant guess: its mean distance of 0.3905 is slightly worse than always predicting the screen center (0.3771), so R1 carries little location information of its own. The later crosshairs, and especially the final prediction, improve on both constants in mean and median distance. The median exhibits an even larger final-stage reduction, falling from 0.2392 at R3 to 0.0255, while the final-stage P75 is only 0.1455. These results provide aggregate evidence for progressive spatial refinement. The higher final-stage P90 of 0.5154 nevertheless indicates a remaining long tail of difficult examples.

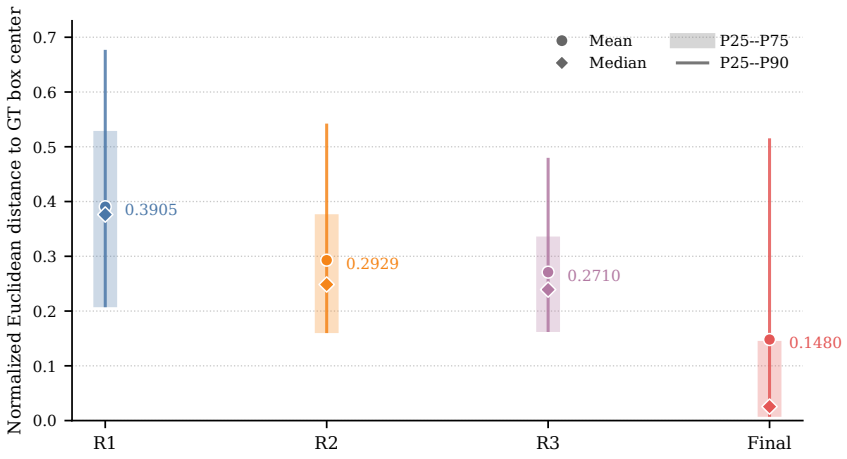


Figure 16: **Aggregate crosshair convergence on ScreenSpot-Pro.** Circles and diamonds denote the mean and median normalized distances, respectively. Thick translucent bars span P25–P75, and thin lines extend from P25 to P90.

Attention Heatmap: Effect of crosshair-guided spatial bias on attention localization. We first compare two checkpoints under matched training and inference configurations on the same example. The vanilla checkpoint is trained without Crosshair Attention and evaluated without the crosshair bias, whereas the full checkpoint is trained with Crosshair Attention and evaluated with the bias enabled. As shown in Figure 17, the crosshair-trained checkpoint produces a stronger and more spatially concentrated attention response around the ground-truth target, while reducing responses at several distractor regions. This qualitative comparison indicates that Crosshair Attention guides the model to learn a more target-focused attention pattern, rather than merely changing the final coordinate readout.

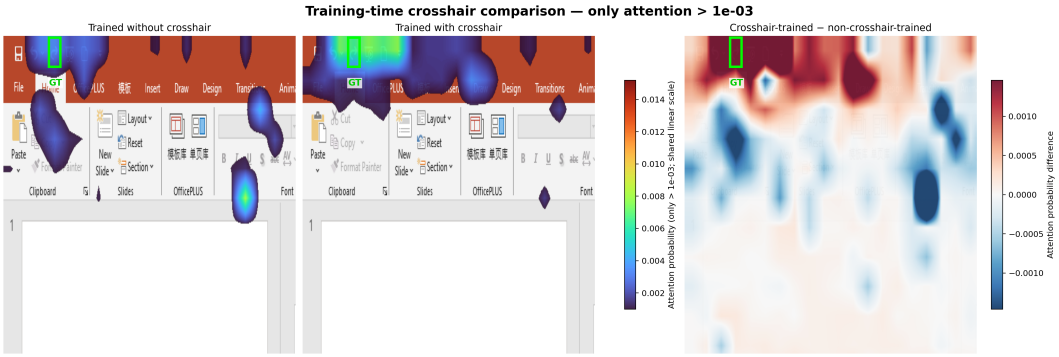


Figure 17: **Attention localization with and without Crosshair Attention during training.** Left: attention from a checkpoint trained and evaluated without Crosshair Attention. Middle: attention from a checkpoint trained and evaluated with Crosshair Attention. Both panels show only attention probabilities above 10^{-3} on the same input and shared linear color scale. Right: the attention difference between the crosshair-trained and vanilla checkpoints, where red and blue indicate increased and decreased attention, respectively. The green bounding box marks the ground-truth *Redo* button.

We further isolate the inference-time effect of the crosshair bias within the same trained checkpoint. Figure 18 compares target-region attention maps obtained with and without the bias, where the "without bias" map is produced by disabling the crosshair bias only at inference in the crosshair-trained model. This intervention is used solely to visualize how the crosshair focuses attention toward the target. The visualization includes only patches whose attention probabilities exceed 10^{-3}. The first two panels use the same linear color scale for direct comparison. After introducing the crosshair bias, the attention response around the ground-truth target—the *Redo* button enclosed by the green bounding box—becomes noticeably stronger and more concentrated. The difference map shows the resulting probability redistribution, with red indicating increased attention and blue indicating decreased attention. Quantitatively, the attention mass inside the ground-truth bounding box increases from 0.00432 to 0.00636, a relative improvement of approximately 47.1%. This example shows that the crosshair-guided spatial prior suppresses patches farther from the predicted reference location and reallocates attention probability toward the target region.

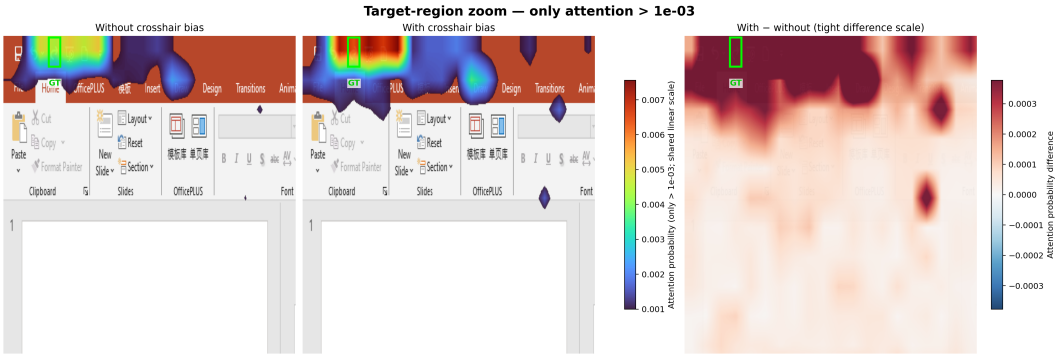


Figure 18: **Attention Heatmap: Effect of crosshair-guided spatial bias on attention localization.** Left and middle: target-region attention without and with crosshair bias, respectively, displaying only probabilities above 10^{-3} on a shared linear scale. Right: their difference on a tighter scale, where red and blue indicate increased and decreased attention, respectively. The green bounding box marks the ground-truth *Redo* button.

L MORE RESULTS ON ATYPICAL-GUI-BENCH

Table 14 provides category-wise results for the 7B models. Under the same training data and Qwen2.5-VL-7B backbone, INTENTLOC-GUI-7B substantially outperforms both Qwen2.5-VL-7B-Finetuned and GUI-Actor-7B, improving overall accuracy by 11.1 and 8.7 percentage points, respectively. The particularly large observed gains over Qwen2.5-VL-7B-Finetuned on *Multi-window or pane* and *Partial UI view* are partly amplified by their smaller test sets of 90 and 67 examples, respectively, for which each additional correct prediction produces a larger change in accuracy. Because device mockups and multi-panel designs make up 55.3% of the benchmark, we also compute the category-balanced accuracy, i.e., the mean of the six category accuracies: INTENTLOC-GUI-7B reaches 69.2% versus 58.6% for GUI-Actor-7B, and INTENTLOC-GUI-3B reaches 54.8% versus 43.2% for GUI-Actor-3B (Table 5).

The results of 7B baselines are obtained by adapting their official code and evaluating the models on Atypical-GUI-Bench. GUI-Owl-7B and UI-TARS-1.5-7B output both coordinates and action types; here, we evaluate only the grounding accuracy of their predicted coordinates.

Table 14: **More Results on Atypical-GUI-Bench.** Arrows give our improvement over the underlined Qwen2.5-VL-7B-Finetuned baseline, which shares our backbone and training data.

Model	Device mockup	Long screenshot	Multi-panel design	Multi-window or pane	Partial UI view	Photographed screen	Overall
Test Data Number	233	250	320	90	67	40	1000
GUI-Owl-7B (Ye et al., 2025)	49.8	39.6	43.1	43.3	43.3	45.0	43.9
Qwen2.5-VL-7B (Bai et al., 2025a)	49.4	44.8	52.2	35.6	53.7	40.0	47.8
UI-TARS-1.5-7B (ByteDance Seed, 2025)	52.8	58.4	53.4	51.1	62.7	42.5	54.5
Qwen2.5-VL-7B-Finetuned (Bai et al., 2025a)	56.7	51.6	59.7	50.0	53.7	62.5	55.8
JEDI-7B (Xie et al., 2025)	59.2	55.6	60.3	51.1	55.2	52.5	57.4
GUI-Actor-7B (Wu et al., 2025a)	54.1	55.2	61.3	62.2	68.7	50.0	58.2
UI-Ins-7B (Chen et al., 2026)	61.4	58.4	64.7	56.7	61.2	70.0	61.6
UI-Venus-Ground-7B (Gu et al., 2025)	64.8	61.2	63.1	51.1	64.2	55.0	61.7
INTENTLOC-GUI-7B (GUI-Actor Data)	65.2 (8.5↑)	62.4 (10.8↑)	**67.8** (8.1↑)	71.1 (21.1↑)	**76.1** (22.4↑)	72.5 (10.0↑)	66.9 (11.1↑)
INTENTLOC-GUI-7B (GUI-Actor Data + 0.25M Data)	**70.0**	**67.2**	66.6	**74.4**	73.1	**82.5**	**69.3**

M SMALL-TARGET ANALYSIS

M.1 UI-I2E-BENCH

UI-I2E-Bench (Liu et al., 2025) complements the direct size-stratified analysis above by evaluating realistic small GUI controls in high-resolution 1080p and 1440p screenshots, where the element-to-screen ratio is substantially lower than in earlier grounding benchmarks. As detailed in **Appendix H**, INTENTLOC-GUI-3B achieves an average accuracy of 67.1%, outperforming the matched GUI-Actor-3B baseline (63.7%) by 3.4 percentage points and obtaining the best result among models of a similar parameter scale.

The advantage is consistent across all five reported element types. INTENTLOC-GUI-3B improves over GUI-Actor-3B on buttons (77.3% vs. 74.0%), icons (74.1% vs. 67.1%), dropdowns (87.1% vs. 84.3%), input controls (69.8% vs. 66.9%), and toggles (44.2% vs. 42.6%). The 7.0-point gain on icons is particularly notable because icons are often spatially compact and visually ambiguous in high-resolution interfaces. Moreover, among models with a comparable parameter scale, INTENTLOC-GUI-3B achieves the highest accuracy on buttons, icons, and dropdowns. These results provide evidence that INTENTLOC preserves a clear localization advantage when interaction targets occupy only a small fraction of the visual input.

M.2 ATYPICAL-GUI-BENCH

To examine whether the improvement on atypical visual observations extends to small interaction targets, we stratify all Atypical-GUI-Bench examples by two complementary measures of ground-truth target size: the bounding-box area relative to the full image and the length of its shortest side in pixels. Table 15 reports top-1 point-in-box accuracy for each bin.

Table 15: **Small-target analysis on Atypical-GUI-Bench.** Target area is the ground-truth bounding-box area as a percentage of the full image area; shortest-side bins are measured in pixels. Δ denotes the percentage-point improvement of INTENTLOC-GUI-3B over GUI-Actor-3B.

Grouping	Target-size bin	# Samples	INTENTLOC-GUI-3B	GUI-Actor-3B	Δ
Area ratio	$< 0.1\%$	145	**44.83**	28.28	+16.55
	0.1%–0.5%	433	**53.81**	42.73	+11.08
	0.5%–2%	309	**58.58**	47.57	+11.01
	$\geq 2\%$	113	**72.57**	56.64	+15.93
Shortest side	< 16 px	68	**36.76**	19.12	+17.64
	16–32 px	321	**53.89**	43.93	+9.96
	32–64 px	355	**58.03**	43.38	+14.65
	≥ 64 px	256	**61.33**	50.39	+10.94

INTENTLOC-GUI-3B consistently outperforms GUI-Actor-3B in every target-size bin. As expected, both methods become less accurate as targets shrink, but INTENTLOC retains a substantial advantage in the most difficult regimes. For targets occupying less than 0.1% of the image, it achieves 44.83% accuracy compared with 28.28% for GUI-Actor, a 16.55-point gain. The advantage is even larger when the target's shortest side is below 16 pixels: 36.76% versus 19.12%, corresponding to a 17.64-point improvement. INTENTLOC also maintains clear gains for larger targets, reaching 72.57% accuracy for targets covering at least 2% of the image and 61.33% when the shortest side is at least 64 pixels. These results suggest that the advantage of INTENTLOC holds across the full target-size range, especially targets represented by only a few visual patches.

N RESOLUTION-SCALING ANALYSIS

Following the resolution-robustness evaluation protocol introduced by ToolTok (Wang et al., 2026c), we construct a benchmark for evaluating robustness to GUI screenshot resolution rescaling. On this benchmark, we compare INTENTLOC-GUI-3B and GUI-Actor-3B's robustness to resolution scaling. We use a fixed random seed to select 1,560 examples from ScreenSpot (240), ScreenSpot-Pro (320), and ScreenSpot-v2 (1,000). For each example, we retain the original image and generate three aspect-ratio-preserving variants whose long edges are 600, 1,000, and 1,400 pixels, respectively. The original images have an average long-edge resolution of 2,553 pixels; therefore, all three resized settings mainly constitute downsampling conditions, and this experiment specifically evaluates robustness to resolution reduction. The ground-truth bounding box is transformed using the same scale factor and clipped to the valid image boundary. Both models are evaluated on exactly the same images using their native pointer heads and top-1 point-in-box accuracy. Because the public ToolTok repository does not release the official robustness sample manifest or complete preprocessing implementation, these results should not be interpreted as an evaluation on an official ToolTok benchmark release.

Table 16: **Resolution robustness on our ToolTok-style benchmark reproduction.** We report top-1 point-in-box accuracy (%). "Mean" averages the original and three resized settings. Arrows give our margin over the underlined GUI-Actor-3B scores in each setting.

Model	Original (2553 px avg.)	600 px	1,000 px	1,400 px	Mean
Qwen2.5-VL-3B-Finetuned	70.26	58.97	68.85	72.31	67.60
GUI-Actor-3B (Wu et al., 2025a)	81.35	60.19	70.45	**75.00**	71.75
INTENTLOC-GUI-3B (ours)	**82.31** (0.96 ↑)	**61.86** (1.67 ↑)	**71.86** (1.41 ↑)	74.87 (0.13 ↓)	**72.72** (0.98 ↑)

As shown in Table 16, INTENTLOC improves the four-setting mean by 0.98 percentage points. Its advantage is largest at 600 pixels (+1.67) and remains consistent at 1,000 pixels (+1.41), while at 1,400 pixels GUI-Actor is ahead by 0.13 points. The relative retention from the original input is also slightly higher for INTENTLOC at 600 pixels (75.16% vs. 74.00%) and 1,000 pixels (87.31% vs. 86.60%). Further inspection shows that both methods degrade sharply when high-resolution ScreenSpot-Pro images are aggressively downscaled, indicating that irreversible loss of small text and icon details remains a dominant bottleneck regardless of pointer-head design. This controlled

experiment isolates scaling: no additional crop is applied, and we do not report unsupported cropping results without a reproducible public protocol. The four-scale example in **Appendix D** further illustrates a case in which INTENTLOC stays on target at every scale, whereas GUI-Actor and the coordinate-prediction baseline miss at two and four scales, respectively.

O FAILURE CASES OF INTENTLOC

We analyze the Top-1 errors of INTENTLOC-GUI-3B on four grounding benchmarks. Figures 19–22 present representative examples of common failure patterns observed in our analysis. Green outlines denote ground-truth target regions, and blue bullseyes denote INTENTLOC's Top-1 predictions, with the center dot marking the exact predicted coordinate.

Fine-grained localization. Although INTENTLOC-GUI-3B outperforms both Qwen2.5-VL-3B-Finetuned and GUI-Actor-3B in small-target settings, fine-grained localization remains a substantial source of error. Small targets with an area no greater than 0.05% account for 32.18% of the 895 MMBench-GUI-L2 failures and 51.19% of the 252 non-refusal OSWorld-G failures. Some of these errors are near-misses in which the model identifies the correct local region but its predicted point falls only slightly outside the target boundary: 81 MMBench-GUI-L2 failures (9.05%) and 19 OSWorld-G failures (7.54%) lie within 0.5% normalized screen distance of the target region. Because point-in-box evaluation is binary, even such minute deviations are counted as complete failures. Figure 19 illustrates a nearly subpixel miss on the correct upvote control and a character-level insertion target occupying only 0.000944% of the screen. Consistently, MMBench-GUI-L2 accuracy is lower for icons than for text targets (69.86% versus 81.50%).

Instance binding and relational reasoning. Finding the correct visual neighborhood is insufficient when an instruction identifies one instance through an ordinal or relational constraint. In Figure 20, the model selects the wrong underline style within the correct menu and clicks the correct spreadsheet row but a neighboring column. These errors indicate that repeated-element counting and row–column binding remain challenging.

Window ownership and candidate ranking. Overlapping windows further require reasoning about z-order and which window owns a control. Lower-ranked predictions contain a correct target for 134 MMBench-GUI-L2 failures and 20 OSWorld-G failures, raising the available top-k oracle accuracy from 75.1% to 78.8% and from 50.6% to 54.5%, respectively. The OSWorld-G figures are computed over the 510 non-refusal examples rather than the full benchmark, which is why they differ from the overall accuracy reported in the main text. The top example in Figure 21 is representative: the correct background-window close control appears at rank 2, while Top-1 selects the foreground close control. The bottom example shows a separate directional-relation error on *left*.

Uncertainty, boundary collapse, and refusal. Under greater uncertainty, predictions may collapse to the image boundary. Among boundary-adjacent predictions, the target itself is not near an edge in 54 MMBench-GUI-L2 failures and 27 OSWorld-G failures. Figure 22 shows this behavior together with a failure to combine slider identity, direction, and endpoint precision. Finally, the current pointer head always produces a positive location and fails all 54 OSWorld-G refusal examples, motivating uncertainty calibration and an explicit no-action output.

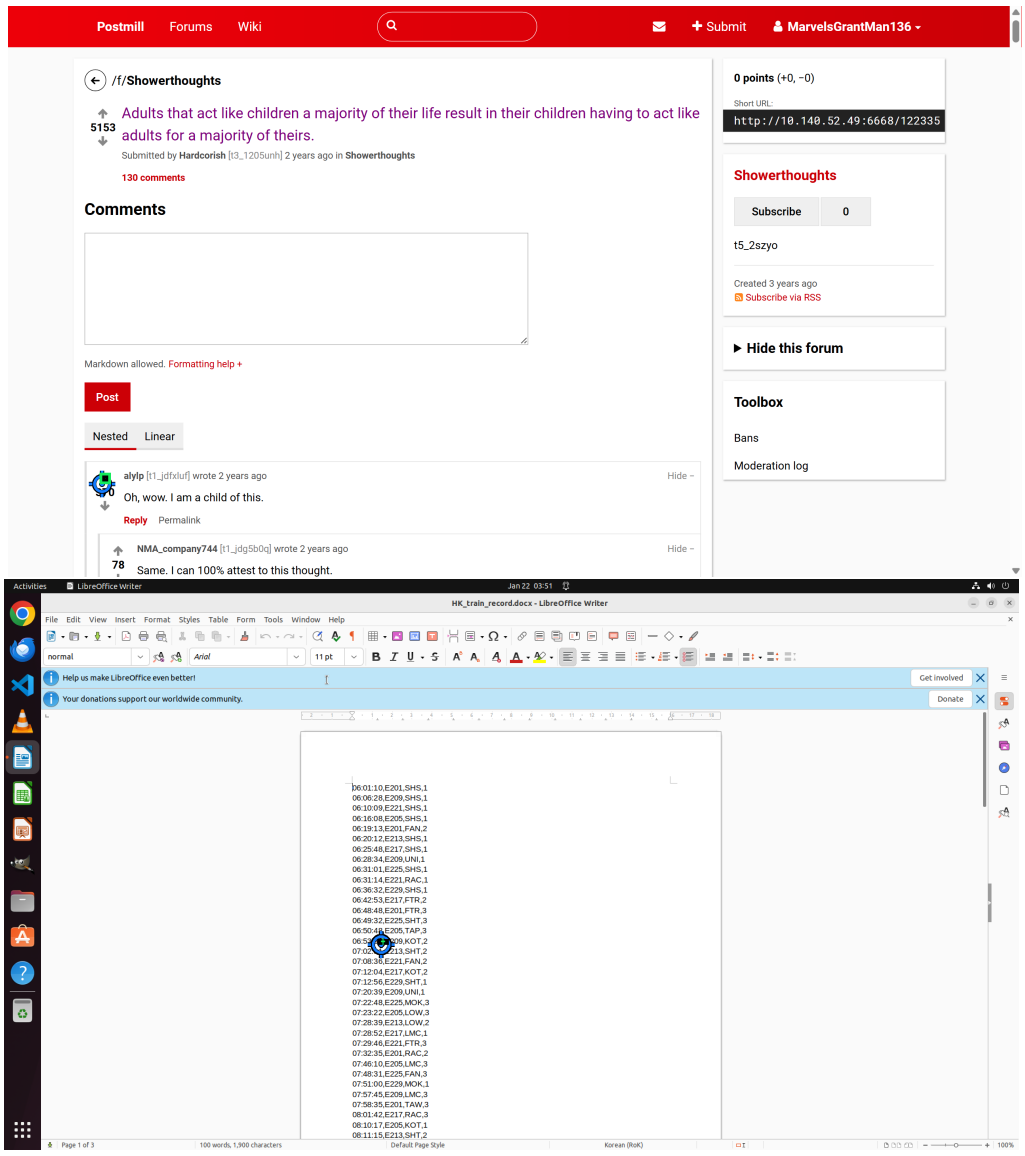


Figure 19: **Fine-grained localization failures.** Top: for "Increase the upvote counts of this comment that currently has 370 points," the prediction identifies the correct upvote control but its center lies approximately 0.08 source-image pixels outside the ground-truth boundary, so binary point-in-box evaluation marks it incorrect. Bottom: for "Click the place right after '06:53:30' and before the comma," the target is a character-level insertion position occupying 0.000944% of the screen; the prediction is nearby but outside this extremely narrow region.

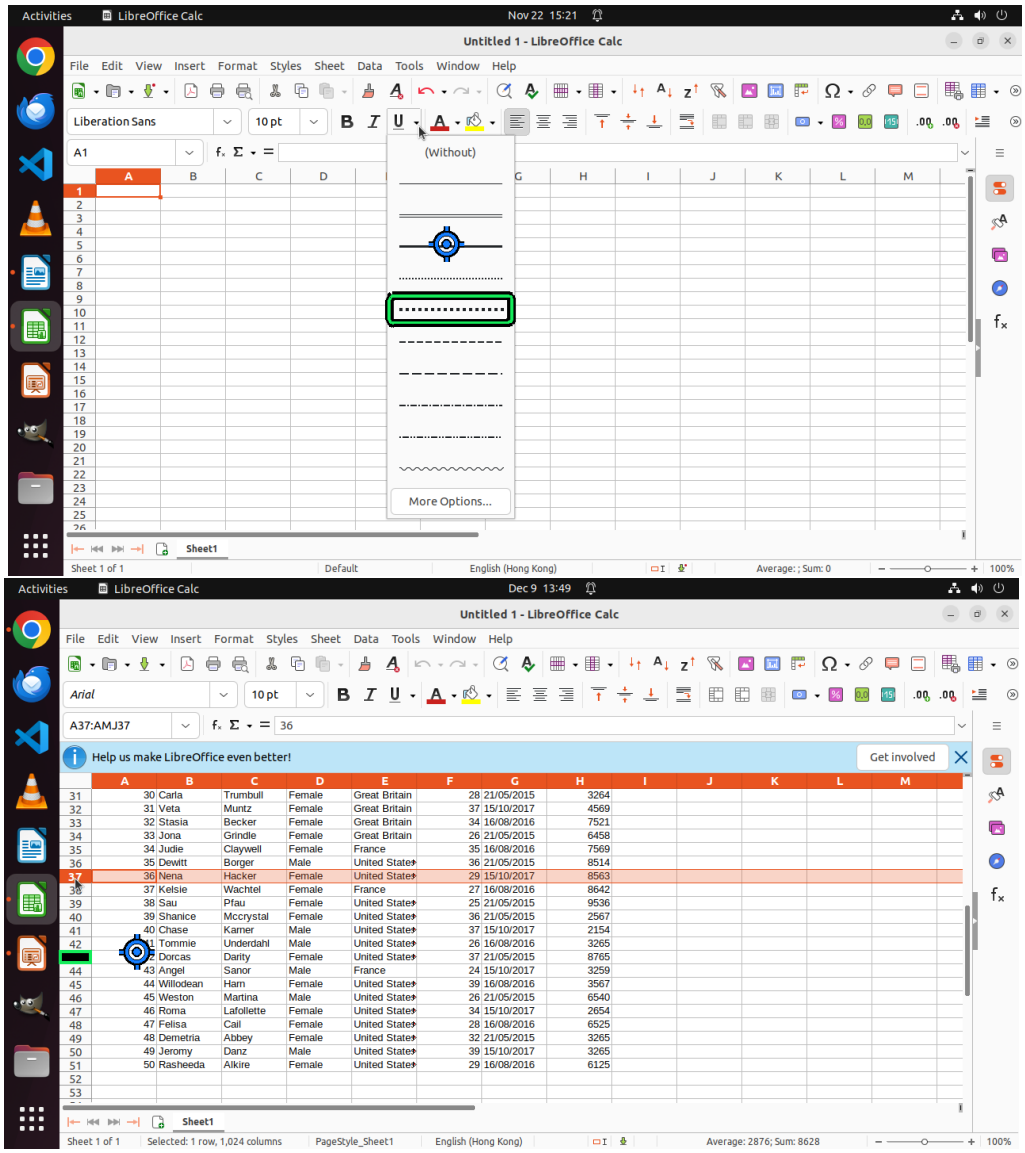


Figure 20: **Instance-binding and relational failures.** Top: for "Choose the fifth underline option from the top down," INTENTLOC finds the correct menu but selects another visually similar row. Bottom: for "Select the row where the value in column A is 42," the model localizes the correct row but clicks a neighboring column, failing to bind the row condition to column A.

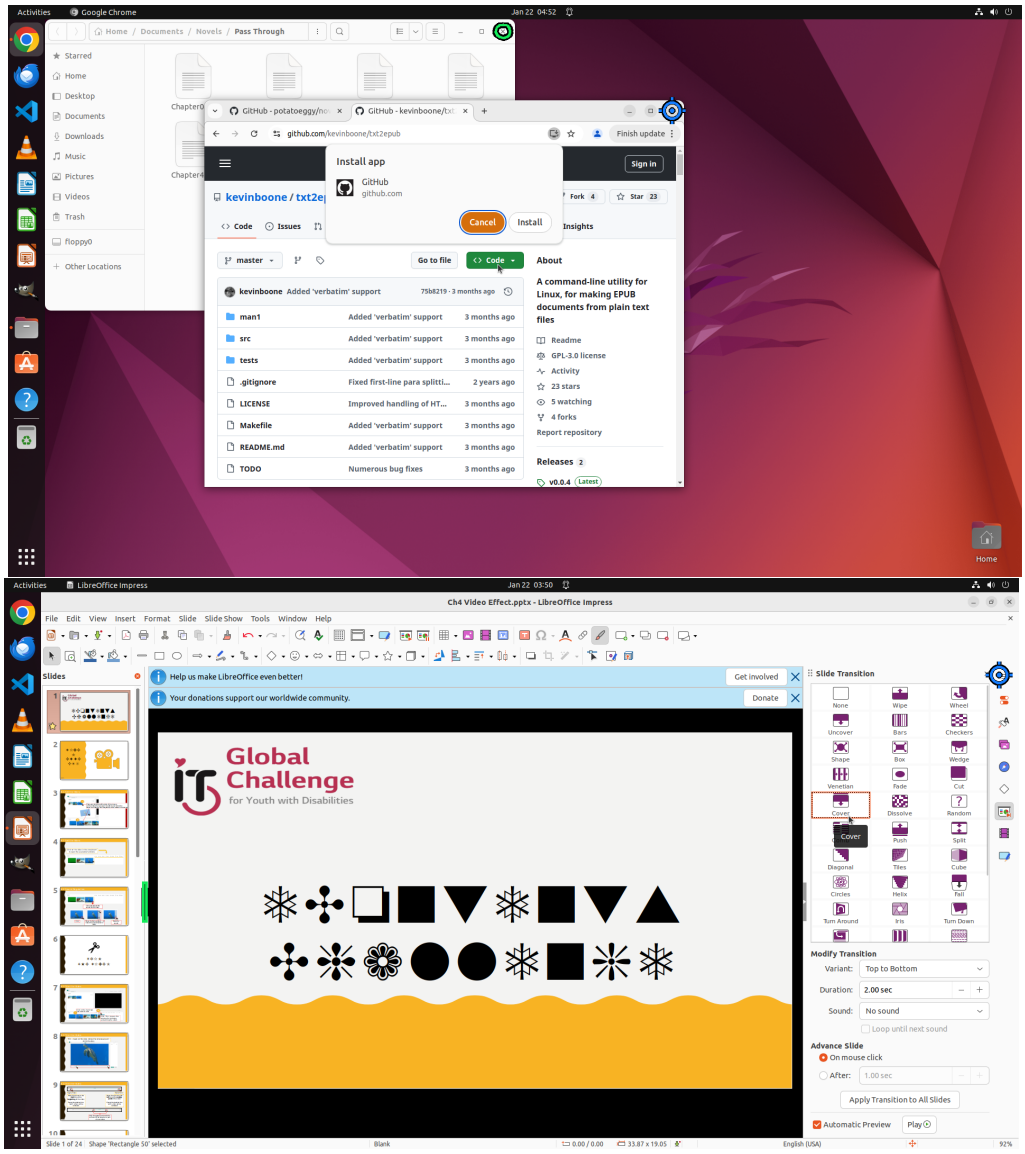


Figure 21: **Window-ownership and spatial-relation failures.** Top: for "Close the partially covered window," Top-1 selects the foreground browser's close control, whereas the correct close control of the partially covered background window appears at rank 2. Bottom: for "Hide the left side panel," the prediction moves to the upper-right boundary instead of the target control on the left, revealing weak grounding of directional relations.

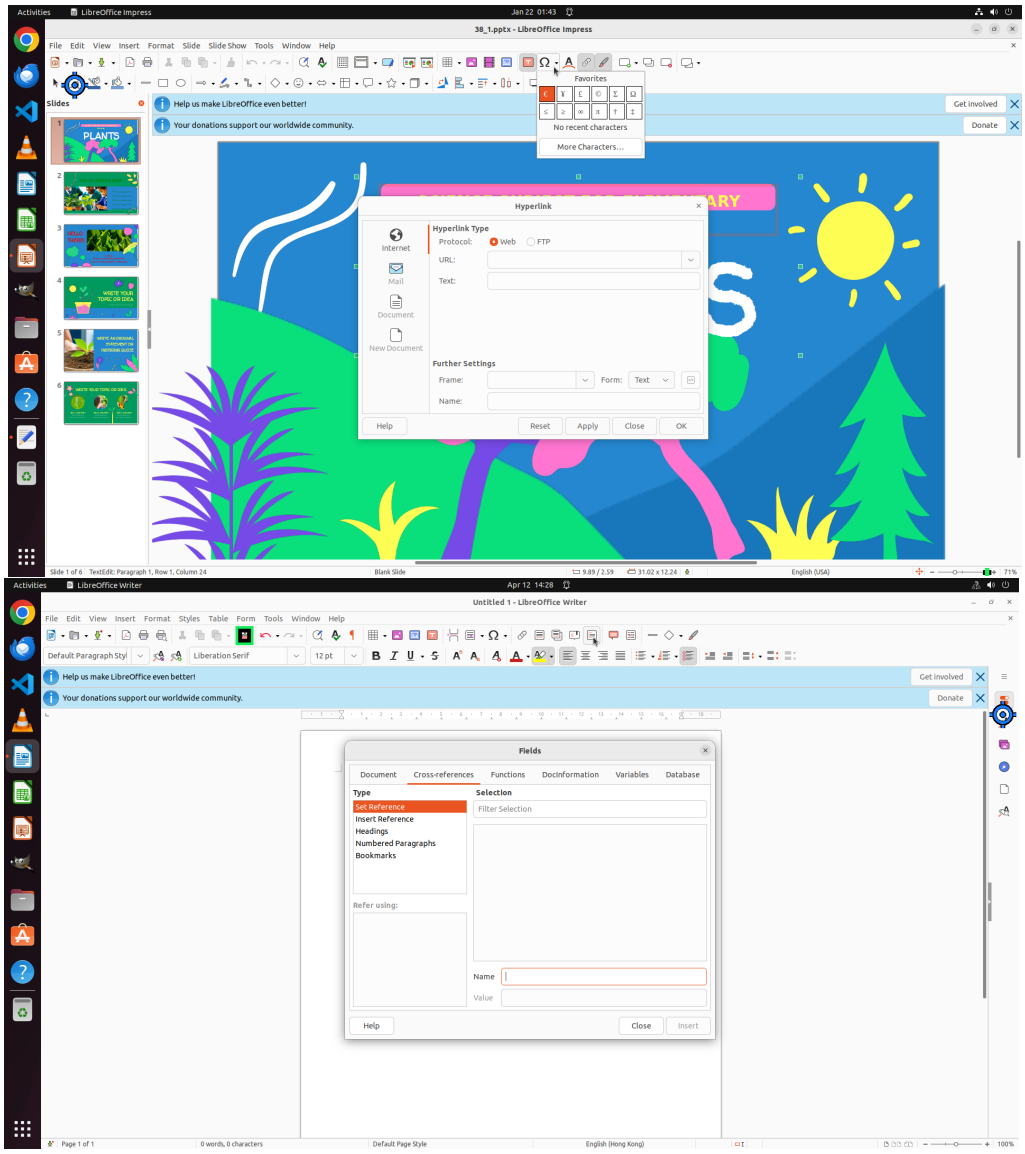


Figure 22: **Endpoint reasoning and boundary-collapse failures.** Top: for "Set the zoom level to maximum by click the right end of the zoom slider," the target is the minute right endpoint of the bottom zoom slider, but the model predicts near the opposite image corner. Bottom: for "A gray-orange brush button in the toolbar of LibreOffice Writer," the target is an interior toolbar icon, while the prediction collapses to the far-right image boundary.

P INFERENCE EFFICIENCY AND MEMORY

We compare the inference cost of INTENTLOC-GUI-7B with the Qwen2.5-VL-7B coordinate-prediction baseline and GUI-Actor-7B under the same Qwen2.5-VL-7B backbone. We measure end-to-end latency on 24 approximately stratified ScreenSpot-Pro samples, after three warm-up samples, using a single NVIDIA H20 96GB GPU, BF16 precision, FlashAttention 2, and batch size 1. Each image is resized to approximately 5.76M pixels while preserving its aspect ratio. Latency includes preprocessing, host-to-device transfer, model inference, coordinate decoding, and post-processing, but excludes checkpoint loading and disk I/O.

Table 17: **Inference efficiency on ScreenSpot-Pro.** Results use each model's native output procedure. Peak memory is the maximum memory allocated by PyTorch during a request.

Model	Avg. latency (s)	Throughput (sample/s)	Peak memory (GiB)	Avg. output tokens
Qwen2.5-VL-7B-Finetuned (Bai et al., 2025a)	2.567	0.390	**18.308**	17.6
GUI-Actor-7B (Wu et al., 2025a)	**2.267**	**0.441**	19.836	-
INTENTLOC-GUI-7B (ours)	2.313	0.432	20.578	-

INTENTLOC is only 2.05% slower than GUI-Actor in the main run (46.6 ms per sample), while its peak memory is 0.742 GiB higher. Compared with the Qwen baseline under native inference, INTENTLOC is 9.9% faster because it avoids autoregressively generating a full coordinate string, at the cost of 2.27 GiB higher peak memory. When the Qwen baseline is artificially restricted to one output token, its latency is 2.282 s, indicating that high-resolution visual encoding and LLM prefill dominate the common inference cost. Overall, INTENTLOC's accuracy-oriented encoder–decoder introduces a modest latency and memory overhead relative to GUI-Actor, while remaining faster than native coordinate generation.