

One Forward Beats Two: InnerZoom for Accurate and Efficient GUI Grounding

Chen Liu^{2*}, Ling Chen², Hanzhang Zhou^{1†}, Liangyu Chen¹,
Chenglin Cai¹, Xin Yu³, Steven HO¹, Yue Wang^{1†}

¹Alibaba Token Hub, Alibaba Group, ²University of Technology Sydney, ³Adelaide University

[†]Correspondence: yue.w@alibaba-inc.com, hanzhang.zhou@alibaba-inc.com

Abstract

MLLM-based GUI grounding methods commonly formulate target localization as autoregressive coordinate generation, enabling models to leverage strong instruction following and semantic understanding. However, by casting localization as coordinate-token prediction, these methods rely on implicit decoding to convert region-level target evidence into precise click coordinates, making point-level spatial alignment unreliable. Through diagnostic analysis, we find that intermediate decoder layers already exhibit target-region awareness, yet such region-level evidence fails to guide final point-level decisions. To address this, we propose **InnerZoom**, a single-forward cross-layer evidence bridging framework that refines target evidence inside the model during decoding, avoiding extra inferences required by Zoom-In. **InnerZoom** reframes coordinate decoding as an evidence-preserving process by identifying target-related cues within the original forward pass, refining them across decoder layers, and reinjecting the refined evidence into coordinate prediction. Extensive experiments show that **InnerZoom-4B** achieves state-of-the-art performance on all six mainstream GUI grounding benchmarks, outperforming the second-best result by 4.1 points on OSWorld-G. Under the 4B setting, **InnerZoom** improves the same SFT+RL baseline by 5.3 points on average and outperforms two-pass Zoom-In by 1.3 points on average, while reducing measured GPU inference latency by 44% and estimated TFLOPs by 29%. Code and models will be at [here](#).

1 Introduction

GUI grounding is a fundamental capability for GUI agents, requiring models to output the target click coordinates based on a user instruction and a GUI screenshot (Nguyen et al., 2025). Recent MLLM-based GUI grounding methods commonly formulate this task as autoregressive coordinate gener-

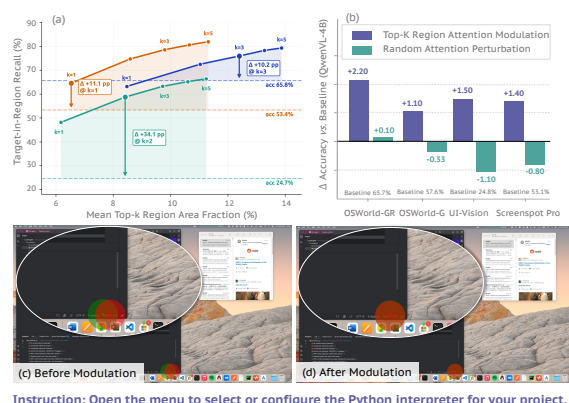


Figure 1: **Motivation and effect of region-guided evidence modulation.** (a) Top- k high-response regions from intermediate text-to-vision response maps often cover the target, yet final predictions can still be inaccurate, revealing a gap between region-level awareness and point-level decoding. (b) Controlled attention intervention suggests that amplifying these regions improves grounding accuracy over random-token perturbation, indicating that they provide decision-relevant evidence. (c,d) Qualitative examples show that target-region modulation strengthens local evidence and moves predictions closer to the intended target. These pre-experiment implementation details are provided in Appendix A.2.

ation, allowing models to inherit the instruction-following, semantic understanding, and unified generation abilities of MLLMs (Xu et al., 2026; Zhou et al., 2025; Li et al., 2025b).

However, this formulation exposes a critical mismatch between region-level visual evidence and point-level coordinate-token generation. Precise GUI clicking requires fine-grained evidence within the target region, yet autoregressive grounding leaves this evidence to be implicitly preserved in decoder states and finally expressed as discrete coordinate tokens (Lin et al., 2025b; Pantazopoulos and Özyiğit, 2025). As local spatial cues can weaken during this implicit conversion, the model may identify the correct region but still decode a biased coordinate, resulting in near-miss clicks. Recent Zoom-In-based methods (Zhang et al., 2026;

*Work done during an internship at Alibaba.

Jiang et al., 2025) mitigate this problem by cropping or zooming into the estimated target region, thereby re-exposing local cues at a higher effective resolution. However, despite their accuracy gains, crop-and-rerun methods require additional inference passes, introducing undesirable latency for interactive GUI grounding, where both accuracy and single-pass efficiency are critical.

This raises a natural question: *Does precise GUI grounding really require visual re-observation?* To answer this, we analyze text-conditioned visual response maps from intermediate decoder layers. As shown in Fig. 1(a), these maps often form compact high-response regions around the ground-truth target, indicating that useful target evidence already emerges before coordinate prediction. We further amplify these regions during decoding and compare them with a random-token control under the same amplification budget. As shown in Fig. 1(b)–(d), targeted amplification improves grounding accuracy and moves predictions closer to the target, while random amplification brings negative gain. This suggests that the bottleneck is not only missing fine-grained visual evidence, but also the unreliable conversion of intermediate region-level evidence into point-level localization. Therefore, instead of re-observing the image through cropping or zooming, we enhance the target evidence already formed within the original forward pass and turn it into a persistent decision signal for coordinate prediction.

To address this evidence-to-coordinate gap, we propose **InnerZoom**, a single-forward cross-layer evidence bridging framework for precise GUI grounding. **InnerZoom** reframes coordinate generation as an evidence-preserving decoding process, turning intermediate target evidence from transient activations into persistent decision signals. Within the original forward pass, the model’s own text-image responses indicate where target evidence emerges, and **InnerZoom** transforms the corresponding fine-grained visual features into a compact cross-layer evidence state. As decoding proceeds, this state is progressively refined to keep click-relevant cues active while accounting for nearby contextual distractions. In this fashion, **InnerZoom** enables a more reliable transition from region-level target awareness to point-level click prediction without additional test-time inputs or extra inference passes.

Extensive experiments show that **InnerZoom** achieves state-of-the-art performance on six mainstream GUI grounding benchmarks. Under con-

trolled same-backbone SFT+RL comparisons, it further surpasses two-pass Zoom-In in both accuracy and efficiency. Overall, our contributions are threefold. (1) We reveal an evidence-to-coordinate bottleneck in MLLM-based GUI grounding, where intermediate decoder layers already exhibit target-region awareness, but such region-level evidence is not reliably translated into precise click coordinates. (2) We introduce **InnerZoom**, which reframes single-forward coordinate generation as an evidence-preserving decoding process by carrying, refining, and reinjecting intermediate target evidence across decoder layers. (3) We provide extensive experiments showing that preserving and refining intermediate target evidence offers an effective and efficient alternative to additional visual re-observation for precise GUI grounding.

2 Related Work

2.1 GUI Grounding for GUI Agents.

GUI grounding is a core capability of GUI agents, enabling models to map natural-language instructions to executable interface actions such as clicking and typing, and navigation. Early works such as SeeClick (Cheng et al., 2024) train agents to locate instruction-relevant UI elements from screenshots, while recent datasets and agent models further scale GUI grounding across platforms and task settings (Qian et al., 2025; Wu et al., 2025b; Lin et al., 2025a; Qin et al., 2025a; Wang et al., 2025a; Yang et al., 2025a). Beyond data scaling and agent integration, recent methods improve GUI grounding through instruction-level reasoning (Chen et al., 2026b), explicit coordinate modeling (Wang et al., 2025b), coordinate-free action-region prediction (Wu et al., 2026), reward modeling (Yuan et al., 2026; Tang et al., 2026b), test-time reinforcement learning (Du et al., 2026), and stabilized reinforcement fine-tuning (Liu et al., 2026a; Kang et al., 2026). Despite these advances, most methods still follow the generative coordinate-prediction paradigm and largely overlook its mismatch with the fine-grained spatial alignment required for point-level GUI grounding. We address this gap by transforming the target-region cues emerging within a single forward pass into localized visual evidence for final coordinate decoding.

2.2 Test-Time Refinement for GUI Grounding

Zoom-based methods improve fine-grained GUI grounding by revisiting local regions at higher ef-

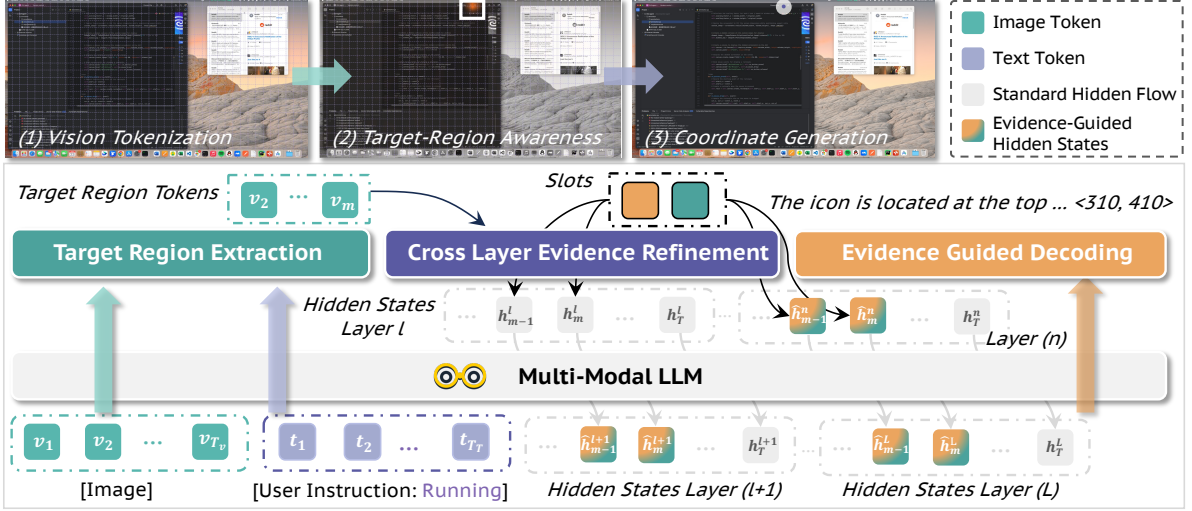


Figure 2: **Overview of InnerZoom.** InnerZoom bridges intermediate target-region awareness and final coordinate generation within a single forward pass. Given a GUI screenshot and an instruction, it identifies a target-aware region from intermediate decoder responses and retrieves fine-grained visual evidence from this region. The localized evidence is maintained and refined across decoder layers through a compact evidence workspace, and is then injected into the key/value projections of the hidden states at selected target-region positions. This enables evidence-guided coordinate decoding while preserving the backbone’s standard autoregressive interface.

fective resolution. Early works study zooming as visual test-time scaling or a training-free grounding prior (Luo et al., 2025; Jiang et al., 2025), while recent methods make this process more adaptive through uncertainty estimation (Wu et al., 2025a), instruction refinement (Chen et al., 2026b), modality-aware reasoning (Tang et al., 2026a), or attention-guided zooming (Pei et al., 2026). Beyond zooming, other test-time scaling methods refine grounding through multi-view aggregation (Zhang et al., 2026), multi-step visual reasoning (Li et al., 2025a), progressive cropping (Nguyen, 2024), tool-augmented focus refinement (Lei et al., 2025), or active visual perception (Chen et al., 2026a). Despite various designs, these methods share the goal of supplementing global GUI understanding with localized visual evidence for fine-grained coordinate prediction, but they typically obtain such evidence through additional test-time computation. In contrast, our method explores how to reuse and refine target-region evidence already formed within a single forward pass.

3 Method

Given a GUI screenshot and a natural-language instruction, the goal of MLLM-based GUI grounding is to generate the target click coordinate autoregressively. We introduce **InnerZoom**, which turns target-related visual cues emerging within the same forward pass into refined localized evidence for precise point-level coordinate prediction.

The inference pipeline of **InnerZoom** is illustrated in Fig. 2. We begin by deriving a text-image relevance map from intermediate decoder representations before coordinate decoding, which provides a coarse target-region proposal for retrieving fine-grained vision-encoder features (§3.1). Next, a shared Iterative Evidence Adapter refines this localized evidence across decoder layers through a compact evidence workspace (§3.2). This workspace maintains evolving evidence states across layers, allowing target-region evidence to be progressively updated and preserved as decoding proceeds. Finally, the refined evidence is injected into the key/value projections of selected target-region positions, allowing coordinate tokens to access position-sensitive evidence while preserving the original autoregressive decoding interface (§3.3).

3.1 Target-Region Evidence Extraction

Our diagnostic analysis in §1 shows that intermediate decoder responses can already highlight target-relevant UI regions, even when the final coordinate prediction fails. We therefore use these responses as self-guided localization cues to determine where fine-grained visual evidence should be extracted for subsequent coordinate decoding.

Specifically, given the hidden states $H^{\ell-1}$ before decoder layer ℓ , we extract the instruction-token states $H_T^{\ell-1} = \{\mathbf{h}_t^{\ell-1}\}_{t=1}^{T_T}$ and the decoder-side visual-token states $H_V^{\ell-1} = \{\mathbf{h}_v^{\ell-1}\}_{v=1}^{T_V}$. We first average the instruction-token states into a single

text-conditioned query state, and then compute its response to the decoder-side visual tokens using the original query and key projections of layer ℓ :

$$A_{T \rightarrow V}^\ell = \text{Softmax} \left(\frac{\bar{Q}_T^\ell (K_V^\ell)^\top}{\sqrt{d_h}} \right), \quad (1)$$

where $\bar{Q}_T^\ell$ and K_V^ℓ are projected from the averaged instruction state and visual-token states, respectively, d_h is the head dimension, and the softmax is applied over visual tokens. We average $A_{T \rightarrow V}^\ell$ across heads and normalize it into a response map. High-response connected components are converted into bounding boxes as coarse target-region priors for fine-grained evidence extraction. More details are provided in the Appendix A.1.1.

Guided by the target-region prior, we retrieve the corresponding final-layer vision-encoder features from the same forward pass. Unlike the compressed visual tokens passed to the language decoder, these features are extracted before visual token merging and thus preserve richer local layout and appearance details. We denote them as $X_R \in \mathbb{R}^{N \times U \times d_v}$, where each of the N selected region tokens corresponds to U unmerged visual patches of dimension d_v . In this way, the decoder response specifies where to gather evidence, and the unmerged features provide the local details needed for point-level coordinate prediction.

3.2 Cross Layer Evidence Refinement

After extracting regional evidence X_R , we introduce a cross-layer evidence workspace for evidence maintenance and refinement during coordinate decoding. By propagating localized visual evidence across decoder layers, the workspace enables progressive refinement and injection of coordinate-relevant cues within a single forward pass.

Cross-layer evidence workspace. To enable progressive refinement, we introduce an *Iterative Dual-Slot Evidence Adapter*. Let $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_K\}$ denote the decoder layers where the adapter is inserted. The adapter A_θ is shared across these layers, while its evidence state is propagated from one layer to the next. In this way, the adapter performs recurrent evidence refinement within the original forward pass. At refinement step k , the workspace maintains two learnable evidence slots, $Z^{(k)} = [\mathbf{z}_1^{(k)}, \mathbf{z}_2^{(k)}] \in \mathbb{R}^{2 \times d_s}$. More details on the slot design are provided in the Appendix A.1.2.

Recurrent target-region evidence aggregation. Given the cross-layer workspace, we specify how

each slot aggregates regional evidence at decoder layer ℓ_k . We flatten X_R into $M = NU$ region tokens, denoted as $\bar{X}_R \in \mathbb{R}^{M \times d_v}$, and project them into keys and values $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{M \times d_s}$.

For each slot $j \in \{1, 2\}$, the shared adapter constructs a slot-specific query by combining the current decoding context with the current slot state:

$$\mathbf{q}_j^{(k)} = \mathbf{q}_{\text{ctx}}^{(k)} + W_z \text{LN}(\mathbf{z}_j^{(k)}) + \mathbf{e}_j, \quad (2)$$

where $\mathbf{q}_{\text{ctx}}^{(k)} \in \mathbb{R}^{d_s}$ is derived from the decoder state at layer ℓ_k and the instruction representation, W_z is a learnable projection, and $\mathbf{e}_j$ is a learnable slot embedding. This makes evidence aggregation conditioned on both the current decoding state and the evidence stored in each slot.

To keep aggregation aligned with the target region, we cache a normalized visual log-prior $\log \pi \in \mathbb{R}^M$ over the flattened region tokens and reuse it across refinement steps. The candidate evidence for slot j is obtained by prior-guided attention:

$$\tilde{\mathbf{z}}_j^{(k)} = \text{Softmax} \left(\frac{\mathbf{q}_j^{(k)} \mathbf{K}^\top}{\sqrt{d_s}} + \beta_j \log \pi \right) \mathbf{V}, \quad (3)$$

where β_j is a learnable slot-specific prior weight. The query-key term adapts evidence selection to the current slot state, while the visual prior guides aggregation toward the estimated target region.

Gated evidence update. After aggregating the candidate evidence $\tilde{\mathbf{z}}_j^{(k)}$, the adapter updates each slot with a learnable gate:

$$\begin{aligned} \mathbf{g}_j^{(k)} &= \sigma \left(W_g [\mathbf{z}_j^{(k)}; \tilde{\mathbf{z}}_j^{(k)}] \right), \\ \mathbf{z}_j^{(k+1)} &= \mathbf{z}_j^{(k)} + \mathbf{g}_j^{(k)} \odot \left(\tilde{\mathbf{z}}_j^{(k)} - \mathbf{z}_j^{(k)} \right). \end{aligned} \quad (4)$$

The gate controls how much candidate evidence is written into the slot, while preserving useful evidence from previous layers. The updated slots are passed to the next inserted layer, enabling cross-layer evidence refinement.

3.3 Evidence-Guided Coordinate Decoding

After cross-layer refinement, the evidence slots are injected into decoding through a KV-only strategy. Specifically, we derive evidence-enhanced states for the target-region positions and use them only to update their key/value projections. The token sequence and all query projections remain unchanged, so coordinate tokens still attend to the

Table 1: Representative comparison across GUI grounding benchmarks. Best and second-best results are highlighted in green and purple, respectively.

Method	Size	OSW-GR	OSW-G	SS-V2	SS-Pro	UI-V	MMB-GUI
Jedi (Xie et al., 2026)	3B	61.0	50.9	—	—	—	—
InfriGUI-G1 (Liu et al., 2026b)	3B	—	—	—	45.2	22.0	—
Ferret-UI Lite (Yang et al., 2025b)	—	—	—	—	53.3	—	—
MAI-UI (Zhou et al., 2025)	4B	69.1	60.3	94.8	65.1	37.0	85.0
OS-Atlas (Wu et al., 2024)	7B	—	27.7	85.1	—	9.0	41.4
Aguvus (Xu et al., 2024)	7B	—	38.7	—	—	—	45.7
UGround (Gou et al., 2025)	7B	—	36.4	87.7	—	12.9	65.7
UI-TARS (Qin et al., 2025b)	7B	—	47.5	91.6	35.7	17.6	—
UI-TARS-1.5 (Seed, 2025)	7B	64.2	52.8	91.6	35.7	22.3	64.3
GUI-Actor (Wu et al., 2026)	7B	—	—	92.1	44.6	—	76.5
SE-GUI (Yuan et al., 2026)	7B	—	—	90.8	47.2	—	76.6
GUI-G ² (Tang et al., 2026b)	7B	—	—	93.3	47.5	—	78.8
UI-Venus (Gu et al., 2025)	7B	61.7	54.6	94.1	50.8	26.5	79.9
GTA1 (Yang et al., 2026a)	7B	67.7	60.1	92.4	50.1	—	78.5
OpenCUA (Wang et al., 2025c)	7B	—	53.3	92.3	50.0	29.7	—
UI-Ins (Chen et al., 2026b)	7B	—	—	94.9	57.0	—	83.1
InfriGUI-G1 (Liu et al., 2026a)	7B	—	—	93.5	51.9	26.1	80.8
GUI-Owl (Ye et al., 2025)	7B	—	55.9	92.8	54.9	—	80.5
Qwen3-VL (Bai et al., 2025)	8B	64.4	55.7	92.9	49.5	23.3	81.3
OpenCUA (Wang et al., 2026)	32B	70.2	59.6	93.4	55.3	—	—
GUI-Owl (Ye et al., 2025)	32B	—	58.0	93.2	58.0	—	83.0
Qwen3-VL (Bai et al., 2025)	32B	69.0	60.6	93.0	54.9	26.9	85.3
UI-TARS-DPO (Qin et al., 2025b)	72B	—	57.1	—	38.1	25.5	74.3
InnerZoom	4B	73.1	64.7	95.2	66.2	40.2	87.6

original target-region positions, but receive refined local evidence through their keys and values.

Slot-conditioned evidence fusion. At layer ℓ_k , let $Z^{(k+1)} = [\mathbf{z}_1^{(k+1)}; \mathbf{z}_2^{(k+1)}] \in \mathbb{R}^{2 \times d_s}$ denote the updated evidence workspace, and let $\mathcal{R}_{\text{lm}}$ denote the target-region positions in the language-model sequence. For each $r \in \mathcal{R}_{\text{lm}}$, we use its decoder state $\mathbf{h}_r^{(k)}$ to attend over the two evidence slots and obtain slot-conditioned evidence $\mathbf{h}_{\text{slot},r}^{(k)}$. This evidence is fused through a gated residual path:

$$\hat{\mathbf{h}}_r^{(k)} = \mathbf{h}_r^{(k)} + \eta_r^{(k)} \odot W_o \mathbf{h}_{\text{slot},r}^{(k)}, \quad (5)$$

where $\eta_r^{(k)}$ is a learned gate controlling the contribution of slot evidence, and W_o maps the slot evidence to the decoder hidden dimension. The resulting $\hat{\mathbf{h}}_r^{(k)}$ is used only for KV evidence injection. **Target-region KV evidence injection.** For each target-region position $r \in \mathcal{R}_{\text{lm}}$, we replace its key/value projections with those computed from the evidence-enhanced state:

$$K_r^{(\ell_k)} \leftarrow W_K^{(\ell_k)} \hat{\mathbf{h}}_r^{(k)}, \quad V_r^{(\ell_k)} \leftarrow W_V^{(\ell_k)} \hat{\mathbf{h}}_r^{(k)}. \quad (6)$$

The query projections and the key/value projections of all non-region positions remain unchanged. This minimally invasive injection preserves the original decoding flow while making the selected target-region positions carry refined, position-specific visual evidence for coordinate prediction.

3.4 Training Stages

In the supervised fine-tuning (SFT) stage, we optimize the standard autoregressive cross-entropy loss

Table 2: Comparison at matched model scales. Green and purple indicate the best and second-best results within each group. *Base SFT+RL* uses the original Qwen3-VL backbone with the same SFT+RL framework and training data, but without our proposed designs. Δ rows show gains/drops over the corresponding baseline. The first group compares test-time scaling methods that use zoom/focus operations.

Method	Size	UI-V	OSW-GR	OSW-G	SS-Pro
Test-time scaling / zoom-based methods					
ZoomClick (Jiang et al., 2025)	7B	34.0	—	—	65.7
ZoomUI (Liu et al., 2026c)	7B	27.1	—	54.2	52.0
GUI-Spotlight (Lei et al., 2025)	7B	23.4	—	62.7	52.8
MVP (Zhang et al., 2026)	8B	31.9	72.7	—	65.3
Region-Focus (Luo et al., 2025)	72B	—	—	—	61.6
InnerZoom	4B	40.2	73.1	64.7	66.2
2B models					
Qwen3-VL	2B	13.7	59.6	45.7	38.3
+ Zoom-In	2B	15.8	64.1	51.2	50.9
Base SFT+RL	2B	27.3	61.5	50.0	51.3
+ Zoom-In	2B	28.9	64.4	53.2	56.2
InnerZoom	2B	30.5	66.8	53.7	53.5
Δ vs. Base SFT+RL	—	+3.2	+5.3	+3.7	+2.2
Δ vs. Base SFT+RL + Zoom-In	—	+1.6	+2.4	+0.5	-2.7
4B models					
Qwen3-VL	4B	24.8	65.7	57.6	53.1
+ Zoom-In	4B	26.8	71.5	61.0	63.1
Base SFT+RL	4B	34.0	67.2	58.3	62.6
+ Zoom-In	4B	36.4	72.5	62.8	67.3
InnerZoom	4B	40.2	73.1	64.7	66.2
Δ vs. Base SFT+RL	—	+6.2	+5.9	+6.4	+3.6
Δ vs. Base SFT+RL + Zoom-In	—	+3.8	+0.6	+1.9	-1.1

over the full target sequence, including grounding reasoning and the final coordinate answer. This adapts the newly introduced evidence pathway while preserving the instruction-following and output-formatting behavior of the base model. We also apply a lightweight slot separation regularization to reduce redundancy between the two evidence slots, with details provided in the Appendix.

In the reinforcement learning (RL) stage, we initialize from the SFT model and apply GRPO (Shao et al., 2024) to further optimize point-level grounding accuracy. For each instruction, the policy samples multiple responses, and each response is rewarded according to whether its parsed coordinate falls inside the target bounding box. GRPO estimates relative advantages within each sampled group, encouraging responses with better grounding outcomes beyond token-level imitation. More training details are provided in the Appendix A.1.3.

4 Experiments

4.1 Experimental Settings

Training Data. Following UI-Ins (Chen et al., 2026b), we train **InnerZoom** on public datasets including OS-Atlas (Wu et al., 2024), Omni-Act (Kapoor et al., 2024), AndroidControl (Li et al., 2024), AMEX (Chai et al., 2025), and Agent-

Table 3: Fine-grained comparison on the **UI-Vision** grounding dataset. **Green/purple** indicates the best/second-best results within each scale group.

Model	Basic	Functional	Spatial	Avg.
<i>Larger models for reference</i>				
Qwen3-VL-32B	32.8	34.2	14.7	26.9
UI-TARS-72B	31.4	30.5	14.7	25.5
UI-Venus-72B	45.6	42.3	23.7	36.8
<i>2B/3B models</i>				
Qwen3-VL-2B	0.0	19.2	0.1	6.2
InfGUI-G1-3B	31.2	28.0	8.2	22.0
MAI-UI-2B	41.0	41.2	10.4	30.3
InnerZoom-2B	37.5	39.5	15.3	30.5
<i>Recent 4B/7B/8B models</i>				
Qwen3-VL-8B	25.0	27.9	1.2	17.5
UI-TARS-1.5-7B	28.8	27.5	10.7	22.3
InfGUI-G1-7B	36.2	31.9	11.5	26.1
UI-Venus-7B	36.1	32.8	11.9	26.5
Phi-Ground	36.8	37.1	7.6	27.2
MAI-UI-4B	47.80	46.50	18.40	37.00
InnerZoom-4B	49.15	47.52	25.43	40.24

Net (Yang et al., 2026b). We adopt the same data processing pipeline as UI-Ins, resulting in 283K SFT samples and 100K RL samples. We use Qwen3-VL-Instruct 2B/4B as the backbone model. More training details are provided in the Appendix A.3.

Metrics and Benchmarks. We evaluate **InnerZoom** using action accuracy (Yang et al., 2026a), where a prediction is counted as correct if the predicted coordinate falls inside the target bounding box. We report results on six GUI grounding benchmarks spanning desktop, mobile, and web interfaces: ScreenSpot-Pro (SS-Pro) (Li et al., 2025b), ScreenSpot-V2 (SS-V2) (Wu et al., 2024), OSWorld-G (OSW-G) and OSWorld-G-Refine (OSW-GR) (Xie et al., 2026), UI-Vision (UI-V) (Nayak et al., 2025), and MMBench-GUI-L2 (MMB-GUI) (Wang et al., 2025d).

Baselines. We compare **InnerZoom** with representative GUI grounding models, such as JEDI (Xie et al., 2026), Qwen3-VL (Bai et al., 2025), OS-Atlas (Wu et al., 2024), GTA1 (Yang et al., 2026a), UI-TARS (Seed, 2025), UI-Venus (Team et al., 2026), UI-Ins (Chen et al., 2026b), and MAI-UI (Zhou et al., 2025).

Parameter Settings. We attain the target regions from decoder layer 19 and keep the top-3 candidate regions, which cover the ground-truth target in around 55%–90% of cases across benchmarks. We then perform cross-layer evidence refinement at decoder layers 20, 23, 26, and 29 to progressively update localized evidence for coordinate decoding. More details are provided in the Appendix.

4.2 Main Results

Comparison with Recent SOTA Methods. Table 1 reports our main results with the 4B version

of **InnerZoom**. With a 4B backbone, **InnerZoom** achieves the best accuracy on all six benchmarks, outperforming recent 4B GUI agents and substantially larger 7B/8B/32B/72B models. Compared with the strongest prior result on each benchmark, **InnerZoom** improves OSW-GR, OSW-G, SS-V2, SS-Pro, UI-V, and MMB-GUI by 2.9, 4.1, 0.3, 1.1, 3.2, and 2.3 points, respectively. We further evaluate a 2B variant to verify the effectiveness of our design, and the results are reported in the following tables and the Appendix.

Scale-Matched Comparison. Table 2 provides controlled scale-matched comparisons and compares with recent test-time scaling methods. In the test-time scaling group, **InnerZoom-4B** achieves the best results on all four benchmarks, outperforming zoom/focus-based methods with larger 7B/8B/72B backbones. This shows that our method improves grounding accuracy without relying on repeated zooming or larger model scale.

Under the same 4B setting, **InnerZoom** outperforms the two-pass Base SFT+RL + Zoom-In baseline on UI-V, OSW-GR, and OSW-G by +3.8, +0.6, and +1.9 points, respectively, while being slightly lower on SS-Pro. This gap is likely caused by the ultra-wide dual-screen samples in SS-Pro, where explicit cropping can better adapt the input to the training distribution. Nevertheless, **InnerZoom** achieves stronger performance on most benchmarks within a single forward pass, showing that its improvements come from effective cross-layer evidence bridging rather than extra supervision or repeated inference. The 2B results show the same trend, further verifying that our design remains effective under smaller model capacity.

4.3 Fine-Grained Analysis

To complement the overall results, we further examine category-level breakdowns on representative GUI grounding benchmarks.

General UI grounding. UI-Vision evaluates grounding under basic, functional, and spatial instruction types. As shown in Table 3, **InnerZoom-4B** achieves the best results among recent 4B/7B/8B models across all three categories. The improvement is especially clear on spatial grounding, where **InnerZoom** improves over MAI-UI-4B from 18.4 to 25.4. This indicates that cross-layer evidence bridging helps preserve local target evidence for more precise point-level localization. The 2B results show a similar trend, where **InnerZoom-2B** achieves the best spatial accuracy

Table 4: Fine-grained comparison on the **MMBench-GUI L2** benchmark. Green/purple indicates the best/second-best results within each scale group. Larger models are shown in gray for reference.

Model	Windows		MacOS		Linux		iOS		Android		Web		Avg.
	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	
Larger models for reference													
GUI-Owl-32B (Ye et al., 2025)	85.6	65.1	84.9	67.1	77.0	63.3	95.2	85.5	96.1	87.0	95.5	80.8	83.0
GTA1-32B (Yang et al., 2026a)	82.3	66.9	89.0	74.0	73.3	52.0	96.2	88.2	95.8	88.5	95.2	79.9	83.4
Qwen3-VL-32B (Bai et al., 2025)	93.4	71.3	92.8	74.3	78.0	56.1	95.5	88.8	97.2	88.5	92.6	78.6	85.3
Recent 4B/7B/8B models													
UI-TARS-1.5-7B (Seed, 2025)	68.3	39.0	69.0	44.5	64.4	37.8	88.5	69.4	90.5	69.3	81.0	56.5	64.3
UGround-V1-7B (Gou et al., 2025)	66.8	39.0	71.3	48.6	56.5	31.1	92.7	70.9	93.5	71.0	88.7	64.6	65.7
GUI-Actor-7B (Wu et al., 2026)	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
SE-GUI-7B (Yuan et al., 2026)	77.5	57.7	77.1	60.7	68.6	44.9	95.5	80.0	95.5	83.7	89.7	68.8	76.6
Qwen3-VL-8B (Bai et al., 2025)	88.6	61.8	85.5	69.1	74.9	53.1	95.2	82.4	95.5	84.5	96.8	72.1	81.3
GTA1-7B (Yang et al., 2026a)	76.8	57.4	80.3	63.9	68.6	53.6	93.9	83.3	96.3	84.5	90.3	74.7	78.5
GUI-G ² -7B (Tang et al., 2026b)	79.7	55.1	79.7	64.7	69.6	50.0	95.2	82.7	96.6	85.4	91.9	75.6	78.8
GUI-Owl-7B (Ye et al., 2025)	86.4	61.8	81.7	64.5	74.4	61.7	94.9	83.0	95.8	83.7	93.2	72.7	80.5
InfGUI-G1-7B (Liu et al., 2026b)	82.7	61.8	83.8	63.9	72.3	52.0	94.9	89.4	95.2	85.6	93.5	76.3	80.8
MAI-UI-4B (Zhou et al., 2025)	91.9	72.4	85.2	74.3	79.1	63.8	95.5	87.3	96.6	87.6	94.2	79.6	85.0
InnerZoom-4B	92.3	72.8	91.9	81.2	83.8	67.9	97.5	88.2	97.2	88.7	96.1	82.1	87.6

Table 5: Fine-grained comparison on **OSWorld-G-Refine**. Green/purple indicates the best/second-best results within each scale group.

Agent Model	Text Matching	Element Recognition	Layout Understanding	Fine-grained Manipulation	Avg
<i>Larger models for reference</i>					
OpenCUA-32B	63.2	79.9	84.9	62.1	70.2
Qwen3-VL-32B	77.4	73.6	76.3	57.7	69.0
GTA1-32B	63.2	83.6	84.4	70.5	72.2
<i>2B/3B models</i>					
Qwen3-VL-2B	69.3	60.9	69.2	45.0	57.4
MAI-UI-2B	70.9	69.1	72.7	47.7	63.5
InnerZoom-2B	73.2	73.0	75.5	52.0	66.8
<i>Recent 4B/7B/8B models</i>					
UI-TARS-1.5-7B	52.6	75.4	72.4	66.7	64.2
Qwen3-VL-8B	73.9	68.2	73.1	54.4	64.4
GTA1-7B	63.2	82.1	74.2	70.5	67.7
MAI-UI-4B	84.6	80.4	82.0	62.2	69.1
InnerZoom-4B	88.3	84.6	85.4	68.9	73.1

and the best average result within the 2B/3B group. **Complex desktop grounding.** OSWorld-G-Refine evaluates desktop grounding scenarios that require text matching, element recognition, layout understanding, and fine-grained manipulation. As shown in Table 5, **InnerZoom-4B** ranks first in text matching, element recognition, and layout understanding, and remains competitive in fine-grained manipulation. This suggests that **InnerZoom** strengthens the visual-semantic alignment and layout-level discrimination required for complex desktop interfaces. Under the 2B setting, **InnerZoom** also achieves the best results across all subcategories.

Cross-platform GUI grounding. MMB-GUI covers diverse platforms with both basic and advanced instructions. As shown in Table 4, **InnerZoom-4B** achieves the best results in most platform-level subcategories, including both basic and advanced settings on Windows, MacOS, Linux, and Android. It also performs strongly on Web advanced instructions, where precise localization often requires

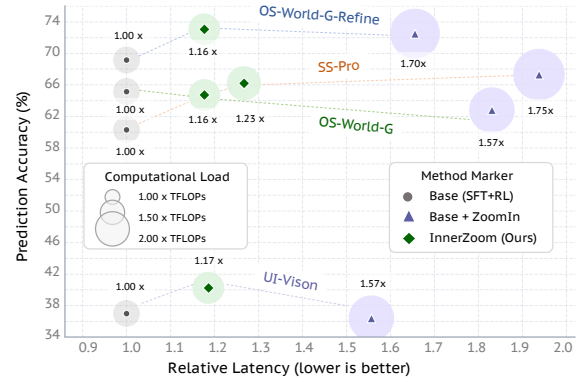


Figure 3: Accuracy–efficiency trade-off under the scale-matched 4B setting. The x-axis denotes relative end-to-end latency normalized by Base SFT+RL on the same benchmark, and the y-axis reports action accuracy. Bubble size represents relative TFLOPs, also normalized by the corresponding Base SFT+RL result. Compared with two-pass Zoom-In, **InnerZoom** achieves comparable accuracy with substantially lower latency and computational cost on most benchmarks.

maintaining fine-grained target evidence until co-ordinate decoding. These results demonstrate that **InnerZoom** generalizes across operating systems and interface styles, especially in categories that require more than coarse UI recognition. More fine-grained results on OSWorld-G, SS-Pro, and SS-V2 are provided in the Appendix A.4.3.

4.4 Ablation Study

Table 6 evaluates the cross-layer interaction layers, number of slots, injection target, visual feature source, and the individual components of the evidence aggregation module across all benchmarks.

Cross-Layer Interaction and Slot Number. Increasing the interaction layers from two to four

Table 6: Ablations of InnerZoom-4B under the full SFT+RL pipeline. † denotes our final configuration. Results are prediction accuracy (%) on six benchmarks.

Variant	OSW-GR	OSW-G	SS-V2	SS-Pro	UI-V	MMB-L2
<i>Cross-layer interaction layers</i>						
20, 23 (90.2M)	69.3	61.1	93.4	63.8	35.8	85.4
20, 23, 26 (135.3M)	70.9	62.8	94.4	64.2	37.5	85.8
20, 23, 26, 29† (180.4M)	73.1	64.7	95.2	66.2	40.2	87.6
20-last (721.8M)	73.3	64.8	95.5	66.6	40.0	87.8
<i>Number of slots</i>						
1 slot	71.2	63.3	94.8	64.5	37.9	86.2
2 slots†	73.1	64.7	95.2	66.2	40.2	87.6
3 slots	71.4	63.1	94.3	65.3	38.5	86.2
<i>Injection target</i>						
KV-only injection†	73.1	64.7	95.2	66.2	40.2	87.6
QKV injection	72.0	63.5	94.7	65.1	38.9	86.8
<i>Visual feature source</i>						
Pre-merge features†	73.1	64.7	95.2	66.2	40.2	87.6
Post-merge features	72.0	63.8	94.8	65.4	39.0	86.9
<i>Individual components</i>						
Full InnerZoom†	73.1	64.7	95.2	66.2	40.2	87.6
w/o prior-guided bias	71.6	63.3	94.4	65.0	39.0	86.6
w/o gated slot update	70.6	62.6	94.2	64.5	38.3	86.0
w/o slot-separation reg.	71.8	63.7	94.6	65.2	39.1	86.7

steadily improves performance. Densely interacting with every layer from layer 20 onward further improves weighted accuracy by only 0.1 points (64.2 to 64.3), while increasing trainable parameters from 180.4M to 721.8M. We therefore use layers 20, 23, 26, and 29 as the accuracy–efficiency trade-off. The two-slot design also consistently outperforms the single- and three-slot variants, indicating additional slots do not provide further benefit.

Injection Target and Visual Feature Source. KV-only injection outperforms QKV injection on all benchmarks, indicating that enriching K/V while preserving the query-side attention dynamics yields more effective integration. Pre-merge visual features likewise consistently outperform post-merge features, supporting the use of fine-grained representations before visual-token merging.

Individual Component Contributions. Removing the prior-guided bias, gated slot update, or slot-separation regularizer consistently reduces performance. The largest degradation occurs without the gated slot update, highlighting the importance of controlling evidence accumulation across interaction layers. The other two components also provide consistent gains across benchmarks.

4.5 Accuracy–Efficiency Trade-off

We further analyze the accuracy–efficiency trade-off under the 4B scale-matched setting. Fig. 3 compares Base SFT+RL, Base SFT+RL + Zoom-In, and InnerZoom on OSW-GR, OSW-G, SS-Pro, and UI-V. Since these benchmarks differ in image resolution, we normalize latency and TFLOPs

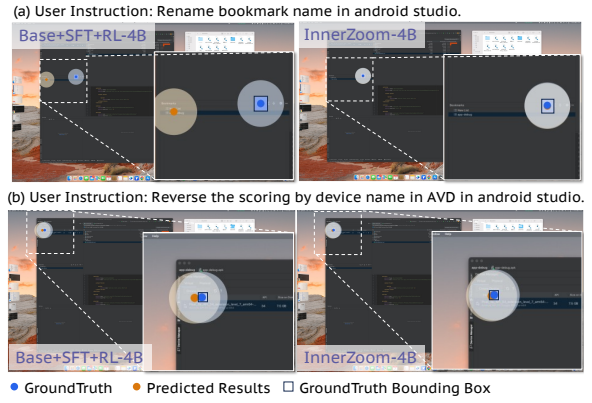


Figure 4: Qualitative comparison between InnerZoom and Base+SFT+RL on ScreenSpot-Pro. InnerZoom achieves more accurate point-level grounding. More examples are provided in the Appendix.

Table 7: Performance across three random seeds. Mean and standard deviation are reported over the three runs.

Benchmark	Seed 100	Seed 1234	Seed 3000	Mean $\pm$ Std.
ScreenSpot-V2	94.9	95.1	95.6	95.2 $\pm$ 0.4
ScreenSpot-Pro	65.7	66.2	66.7	66.2 $\pm$ 0.5

within each benchmark using Base SFT+RL as the reference. Detailed measurement and estimation protocols are provided in the Appendix A.4.1.

As shown in Fig. 3, Base SFT+RL + Zoom-In relies on a conditional second forward pass, leading to substantially higher overhead with $1.56\text{--}1.94\times$ latency and $1.57\text{--}1.75\times$ TFLOPs. In contrast, InnerZoom achieves stronger accuracy on most benchmarks while staying close to the base model, requiring only $1.18\text{--}1.27\times$ latency and $1.16\text{--}1.23\times$ TFLOPs. Compared with Zoom-In, this reduces latency by $23.8\text{--}35.7\%$ and TFLOPs by $26.0\text{--}32.0\%$, cutting most of the additional overhead introduced by repeated inference. With this much smaller computational cost, InnerZoom still outperforms Zoom-In on OSW-GR, OSW-G, and UI-V by $+0.6$, $+1.9$, and $+3.8$ points, respectively. These results show that reusing localized evidence within a single forward pass provides a more favorable accuracy–efficiency trade-off than external zoom-based reprocessing.

We further measure synchronous GPU inference latency with batch size 4, where triggered samples are gathered for a second-pass Zoom-In inference and non-triggered samples wait for batch completion. As shown in Table 10, InnerZoom incurs only $1.04\text{--}1.06\times$ the Base latency, versus $1.82\text{--}1.91\times$ for conditional Zoom-In, yielding a $1.72\text{--}1.84\times$ speedup over Zoom-In.

Table 8: Transferability across architectures and model scales. For each backbone, the baseline is trained with the same data and training configuration as InnerZoom.

Variant	OSW-GR	OSW-G	SS-Pro	UI-V	MMB-L2	SS-V2
Qwen2.5-VL-7B						
Pretrained	42.9	34.6	12.3	12.9	62.7	82.9
Baseline (SFT+RL)	66.8	55.1	52.2	19.1	83.1	94.0
Baseline (SFT+RL) + ZoomIn	71.1	58.9	56.9	21.2	84.6	93.2
InnerZoom	72.2	60.3	58.1	24.5	85.2	94.7
Qwen3-VL-8B						
Pretrained	64.5	55.7	49.5	23.3	81.3	92.9
Baseline (SFT+RL)	67.9	59.4	65.4	39.6	88.2	94.9
Baseline (SFT+RL) + ZoomIn	72.3	63.8	69.8	40.8	88.5	94.7
InnerZoom	73.1	66.0	68.9	41.1	90.4	96.5

Table 9: Accuracy decomposition by joint region-hit status. Each entry reports Base SFT+RL / InnerZoom. Both-Hit/-Miss include samples for which the two models share the same Top-3 region-hit status; samples with different hit statuses are excluded. Weighted contribution is the subset proportion multiplied by its accuracy.

Benchmark	Recall@3 Base / Ours	Both-Hit Acc. Base / Ours	Both-Hit W. Contrib. Base / Ours	Both-Miss Acc. Base / Ours	Both-Miss W. Contrib. Base / Ours
UI-V	85.11 / 94.47	36.31 / 42.00	30.74 / 35.55	21.94 / 24.10	1.11 / 1.22
OSW-G	84.75 / 88.65	61.91 / 68.51	51.59 / 57.09	42.86 / 42.86	4.26 / 4.26
OSW-GR	87.59 / 88.83	70.49 / 76.64	60.99 / 66.31	47.37 / 47.37	4.79 / 4.79
SS-Pro	93.55 / 98.61	64.30 / 67.48	60.03 / 63.00	47.37 / 52.63	0.57 / 0.63
MMB-L2	90.18 / 98.05	86.89 / 88.22	78.16 / 79.36	67.74 / 70.97	1.17 / 1.22
SS-V2	93.40 / 98.90	95.45 / 95.95	89.00 / 89.46	25.00 / 25.00	0.24 / 0.24

Table 10: Batch-4 GPU inference latency. Latency is measured with a batch size of 4 and reported in milliseconds per sample. CPU-side cropping, resizing, and input preparation are excluded.

Benchmark	Base SFT+RL	Base + Zoom-In	InnerZoom
OSW-GR	1981.6	3745.9 (1.89 $\times$)	2071.7
OSW-G	1224.7	2225.4 (1.82 $\times$)	1293.5
SS-Pro	3904.9	7465.5 (1.91 $\times$)	4052.8
UI-V	1604.1	2928.5 (1.83 $\times$)	1660.3

4.6 Qualitative and Failure Analysis

Qualitative Results. Fig. 4 presents a qualitative comparison between Base SFT+RL and **InnerZoom** on SS-Pro. **InnerZoom** produces more accurate point-level predictions in dense UI regions with visually similar neighboring elements, showing that localized evidence bridging helps refine coordinate prediction beyond coarse target-region awareness (see more in App. A.4.3.).

Failure cases. The remaining errors mainly arise from semantically challenging instructions, ambiguous text-to-image grounding, and interference from on-screen text. These cases suggest that although **InnerZoom** improves fine-grained localization, robust domain-specific instruction understanding and command-interface text disentanglement remain open challenges (see more in App. 4.6.).

4.7 Further Analysis

Hyperparameter Sensitivity. Table 11 evaluates the sensitivity of InnerZoom to its main hyperpa-

Table 11: Sensitivity analysis of InnerZoom-4B. † denotes the default setting, and bold indicates the best result within each sensitivity sweep.

Setting	Value	OSW-GR	OSW-G	SS-Pro	UI-V
Extraction layer	19 [†] / 20	73.0 [†] / 73.2	64.7[†] / 64.5	66.2 [†] / 66.3	40.2[†] / 40.1
q_{thr}	0.85 / 0.90 [†]	72.9 / 73.0[†]	64.9 / 64.7 [†]	66.3 / 66.2 [†]	40.1 / 40.2[†]
r_{pad}	0.20 / 0.30 [†] / 0.40	73.2 / 73.0 [†] / 72.9	64.5 / 64.7 [†] / 64.9	66.0 / 66.2 [†] / 66.4	40.0 / 40.2[†] / 40.1

rameters. Across all sweeps, performance varies by at most 0.4 points, showing that InnerZoom is insensitive to reasonable hyperparameter choices. For the extraction layer, layers 19 and 20 exhibit only minor and complementary differences across benchmarks; when using layer 20, we shift the interaction layers from 20, 23, 26, and 29 to 21, 24, 27, and 30 to preserve their relative depth. We therefore use one fixed configuration for all benchmarks without dataset-specific tuning.

Where Do the Gains Come From? Table 9 separates gains from improved region coverage and more accurate grounding within localized regions. InnerZoom improves Recall@3 on all benchmarks, while Both-Hit samples contribute most of the gains on UI-Vision, OSW-G, OSW-GR, and ScreenSpot-Pro, with Both-Miss contributions changing by at most 0.11 points. On MMBench-GUI L2 and ScreenSpot-V2, where Both-Hit accuracy is already high, gains mainly come from improved region coverage.

Transferability across Backbones and Scales. Table 8 evaluates InnerZoom on Qwen2.5-VL-7B and Qwen3-VL-8B. InnerZoom improves the matched SFT+RL baselines on all benchmarks by 0.7–5.9 points and 1.5–6.6 points, respectively. It also outperforms external ZoomIn on all benchmarks with Qwen2.5-VL-7B and on five of six with Qwen3-VL-8B. These results show that InnerZoom generalizes across backbones and model scales.

5 Conclusion

We presented **InnerZoom**, a single-forward cross-layer evidence-bridging framework for precise GUI grounding. Our analysis reveals an evidence-to-coordinate bottleneck, where intermediate decoder layers already form useful target-region evidence but fail to reliably convert it into final click coordinates. **InnerZoom** addresses this gap by preserving and refining intermediate target evidence across decoder layers and making it available for coordinate decoding. Experiments suggest that this evidence-preserving design improves GUI grounding accuracy while maintaining efficient single-forward inference, providing an effective alternative to two-pass zoom-in refinement.

6 Limitations

Despite its strong performance, **InnerZoom** still has several limitations. First, its evidence extraction depends on target-region cues emerging from intermediate decoder layers. When these internal responses are incomplete or biased, the selected region may limit subsequent evidence refinement. Since region selection is only indirectly optimized during training, future work could explore learnable or weakly supervised mechanisms for more robust evidence extraction.

Second, although **InnerZoom** achieves strong results on high-resolution benchmarks such as ScreenSpot-Pro, explicit visual re-observation may still be complementary in extreme cases. For ultra-wide or dual-screen interfaces, a two-pass zoom-in strategy can provide higher effective resolution over the target area, offering visual details that are difficult to fully recover from the original forward pass. This suggests that single-forward evidence bridging and zoom-based re-observation address different aspects of precise grounding, and future work may combine them adaptively for more challenging high-resolution GUI scenarios.

References

- Shuai Bai, Yuxuan Cai, Ruizhe Chen, Kebin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.
- Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. 2025. Amex: Android multi-annotation expo dataset for mobile gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2138–2156.
- Chen Chen, Jiawei Shao, Dakuan Lu, Haoyi Hu, Xiangcheng Liu, Hantao Yao, and Wu Liu. 2026a. Guieyes: Tool-augmented perception for visual grounding in gui agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 29350–29358.
- Liangyu Chen, Hanzhang Zhou, Jianan Zhang, Panrong Tong, Xu Zhang, Quyu Kong, Chen Liu, Yuqi Liu, Wenxuan Wang, Yue Wang, and 1 others. 2026b. Ui-ins: Enhancing gui grounding with multi-perspective instruction as reasoning. In *International Conference on Learning Representations*, volume 2026, pages 54418–54444.
- Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332.
- Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.
- Yong Du, Yuchen Yan, Fei Tang, Zhengxi Lu, Chang Zong, Weiming Lu, Shengpei Jiang, and Yongliang Shen. 2026. Test-time reinforcement learning for gui grounding via region consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30593–30601.
- Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for gui agents. In *International Conference on Learning Representations*, volume 2025, pages 30851–30883.
- 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, and 5 others. 2025. [Ui-venus technical report: Building high-performance ui agents with rft](#). *Preprint*, arXiv:2508.10833.
- Zhiyuan Jiang, Shenghao Xie, Wenyi Li, Wenqiang Zu, Peihang Li, Jiahao Qiu, Siqi Pei, Lei Ma, Tiejun Huang, Mengdi Wang, and 1 others. 2025. Zoom in, click out: Unlocking and evaluating the potential of zooming for gui grounding. *arXiv preprint arXiv:2512.05941*.
- Weitai Kang, Bin Lei, Gaowen Liu, Caiwen Ding, and Yan Yan. 2026. Guirlvg: Incentivize gui visual grounding via empirical exploration on reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pages 63861–63875.
- Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. 2024. [Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web](#). *Preprint*, arXiv:2402.17553.
- Bin Lei, Nuo Xu, Ali Payani, Mingyi Hong, Chunhua Liao, Yu Cao, and Caiwen Ding. 2025. Gui-spotlight: Adaptive iterative focus refinement for enhanced gui visual grounding.
- Aiden Yiliu Li, Bizhi Yu, Daoan Lei, Tianhe Ren, and Shilong Liu. 2025a. Chain-of-ground: Improving gui grounding via iterative reasoning and reference feedback. *arXiv preprint arXiv:2512.01979*.

- Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. 2025b. [Screenspot-pro: Gui grounding for professional high-resolution computer use](#). *Preprint*, arXiv:2504.07981.
- Wei Li, William Bishop, Alice Li, Chris Rawles, Fola Lawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. [On the effects of data scale on ui control agents](#). *Preprint*, arXiv:2406.03679.
- Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2025a. Showui: One vision-language-action model for gui visual agent. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19498–19508. IEEE.
- Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. 2025b. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 5334–5342.
- Yuhang Liu, Zeyu Liu, Shuanghe Zhu, Pengxiang Li, Congkai Xie, Jiasheng Wang, Xueyu Hu, Xiaotian Han, Jianbo Yuan, Xinyao Wang, and 1 others. 2026a. Infogui-g1: Advancing gui grounding with adaptive exploration policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32267–32275.
- Yuhang Liu, Zeyu Liu, Shuanghe Zhu, Pengxiang Li, Congkai Xie, Jiasheng Wang, Xueyu Hu, Xiaotian Han, Jianbo Yuan, Xinyao Wang, and 1 others. 2026b. Infogui-g1: Advancing gui grounding with adaptive exploration policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32267–32275.
- Ziwei Liu, Tao Feng, Borui Kang, Yanbing Yang, and Jun Luo. 2026c. Zoom to essence: Trainless gui grounding by inferring upon interface elements. *arXiv preprint arXiv:2603.14448*.
- Tiange Luo, Lajanugen Logeswaran, Justin Johnson, and Honglak Lee. 2025. Visual test-time scaling for gui agent grounding. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19989–19998. IEEE.
- Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, and 1 others. 2018. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, pages 561–577.
- Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M. Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. 2025. [Ui-vision: A desktop-centric gui benchmark for visual perception and interaction](#). *Preprint*, arXiv:2503.15661.
- Anthony Nguyen. 2024. Improved gui grounding via iterative narrowing. *arXiv preprint arXiv:2411.13591*.
- Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, and 1 others. 2025. Gui agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22522–22538.
- Georgios Pantazopoulos and Eda B Özyiğit. 2025. Towards understanding visual grounding in visual language models. *arXiv preprint arXiv:2509.10345*.
- Siqi Pei, Liang Tang, Tiaonan Duan, Long Chen, Shuxian Li, Kaer Huang, Yanzhe Jing, Yiqiang Yan, Bo Zhang, Chenghao Jiang, and 1 others. 2026. Adazoom-gui: Adaptive zoom-based gui grounding with instruction refinement. *arXiv preprint arXiv:2603.17441*.
- Rui Qian, Xin Yin, Chuanhang Deng, Zhiyuan Peng, Jian Xiong, Wei Zhai, and Dejing Dou. 2025. Uground: Towards unified visual grounding with unrolled transformers. *arXiv preprint arXiv:2510.03853*.
- Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025a. Uitars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.
- Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025b. Uitars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pages 1–16. IEEE.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- ByteDance Seed. 2025. Ui-tars-1.5. <https://seed-tars.com/1.5>.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

- Fei Tang, Bofan Chen, Zhengxi Lu, Tongbo Chen, Songqin Nong, Tao Jiang, Wenhao Xu, Weiming Lu, Jun Xiao, Yueting Zhuang, and 1 others. 2026a. Ui-zoomer: Uncertainty-driven adaptive zoom-in for gui grounding. *arXiv preprint arXiv:2604.14113*.
- Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, and 1 others. 2026b. Gui-g²: Gaussian reward modeling for gui grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33214–33222.
- Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, and 1 others. 2026. Ui-venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*.
- Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, and 1 others. 2025a. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*.
- Suyuchen Wang, Tianyu Zhang, Ahmed Masry, Christopher Pal, Spandana Gella, Bang Liu, and Perouz Taslakian. 2025b. Improving gui grounding with explicit position-to-coordinate mapping. *Preprint*, arXiv:2510.03230.
- Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, and 1 others. 2026. Open-cua: Open foundations for computer-use agents. *Advances in Neural Information Processing Systems*, 38:139756–139806.
- Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, and 23 others. 2025c. Opencua: Open foundations for computer-use agents. *Preprint*, arXiv:2508.09123.
- Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, Weiyun Wang, Xiangyu Zhao, Jixuan Chen, Haodong Duan, Tianbao Xie, Chenyu Yang, Shiqian Su, Yue Yu, Yuan Huang, and 9 others. 2025d. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. *Preprint*, arXiv:2507.19478.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
- Hang Wu, Hongkai Chen, Yujun Cai, Chang Liu, Qingwen Ye, Ming-Hsuan Yang, and Yiwei Wang. 2025a. Dimo-gui: Advancing test-time scaling in gui grounding via modality-aware visual reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26246–26256.
- Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, and 1 others. 2026. Gui-actor: Coordinate-free visual grounding for gui agents. *Advances in Neural Information Processing Systems*, 38:15101–15128.
- Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. 2024. Os-atlas: A foundation action model for generalist gui agents. *Preprint*, arXiv:2410.23218.
- Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and 1 others. 2025b. Os-atlas: Foundation action model for generalist gui agents. In *International Conference on Learning Representations*, volume 2025, pages 5090–5108.
- Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, and 1 others. 2026. Scaling computer-use grounding via user interface decomposition and synthesis. *Advances in Neural Information Processing Systems*, 38.
- Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, and 1 others. 2026. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*.
- Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2024. Aguis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*.
- Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, and 1 others. 2026a. Gta1: Gui test-time scaling agent. In *International Conference on Learning Representations*, volume 2026, pages 48295–48323.
- Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. 2026b. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems. *Advances in Neural Information Processing Systems*, 38:107309–107336.
- Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2025a. Aria-ui: Visual grounding for gui instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22418–22433.

- Zhen Yang, Zi-Yi Dou, Di Feng, Forrest Huang, Anh Nguyen, Keen You, Omar Attia, Yuhao Yang, Michael Feng, Haotian Zhang, and 1 others. 2025b. Ferret-ui lite: Lessons from building small on-device gui agents. *arXiv preprint arXiv:2509.26539*.
- Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, and 1 others. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*.
- Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, and 1 others. 2026. Se-gui: Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *Advances in Neural Information Processing Systems*, 38:127658–127679.
- Yunzhu Zhang, Zeyu Pan, Zhengwen Zeng, Shuheng Shen, Changhua Meng, and Linchao Zhu. 2026. Mvp: Multiple view prediction improves gui grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27482–27492.
- Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, and 1 others. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.
- Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, and 1 others. 2025. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*.

A Appendix

A.1	Method Details	14
A.1.1	Target-Region Proposal and Feature Retrieval . . .	14
A.1.2	Slot Separation Regularization	15
A.1.3	Training Objective	15
A.2	Implementation Details of Diagnostic Attention Intervention . . .	17
A.3	Experiment Details	17
A.4	More Experimental Results	17
A.4.1	Latency Measurement and TFLOPs Estimation. . . .	17
A.4.2	More Fine-grained Results.	18
A.4.3	Additional Qualitative Results.	19
A.4.4	Failure analysis.	19

A.1 Method Details

This section provides additional implementation details of our method. In particular, we elaborate on the details of locating text and visual tokens in the input sequence, constructing a heatmap from the text-to-vision response, selecting ROIs through connected-component analysis, and mapping the expanded regions back to token indices.

A.1.1 Target-Region Proposal and Feature Retrieval

Text-to-Vision Heatmap Construction. To obtain a target-region proposal from the model’s own intermediate responses, we first locate the user-instruction tokens and visual tokens in the input sequence. Specifically, we identify the user instruction span using the special tokens $\langle |im_start| \rangle$ and $\langle |im_end| \rangle$, and locate the visual-token span using $\langle |vision_start| \rangle$ and $\langle |vision_end| \rangle$. Since different prompt templates or preprocessing pipelines may place the visual tokens either before or after the user instruction, and batched inputs may be left- or right-padded depending on the training or inference setting, we employ the attention mask provided with the model inputs to identify the first valid token position. This mask distinguishes real input tokens from padding tokens, allowing us to shift the parsed relative spans to their absolute positions in the full input sequence. The parsed relative positions are then shifted to absolute indices in the full input sequence, ensuring that subsequent attention hooks access the correct text and visual-token ranges.

Our target region proposal only requires a text-to-vision heatmap from one selected intermediate decoder layer. Current decoder backbones often support eager attention (Wolf et al., 2019), PyTorch SDPA, and FlashAttention-style kernels (Dao et al., 2022), which differ in whether intermediate attention maps are exposed. Under the standard eager-attention output interface, enabling attention outputs typically materializes full sequence-to-sequence attention matrices across decoder layers, even though we only need the text-to-vision response from one layer. This causes unnecessary memory overhead and may slow down training by disabling more efficient fused attention kernels. In contrast, PyTorch SDPA returns only the attention output, while FlashAttention-style kernels avoid materializing the full attention matrix to reduce memory traffic. Thus, directly extracting the required intermediate attention evidence from the backbone is either memory-intensive or incompatible during training.

To preserve the benefits of optimized attention backends while still obtaining the required attention evidence, we recompute the attention response only during target-region proposal generation. For the selected intermediate decoder layer, we reuse its Q/K projections to compute a restricted attention response from the user-text tokens to the visual tokens. We first average the projected queries of all user-text tokens to obtain an instruction-level query for each attention head. To reduce memory usage, we split the visual-key sequence into 256-token chunks along the key dimension, compute the logits between the instruction-level query and each chunk of visual keys, concatenate the chunk-wise logits, and apply softmax over the visual-token dimension. The resulting response assigns each visual token a target-relevance score with respect to the user instruction. These scores are aggregated across attention heads, normalized, and reshaped into a two-dimensional visual grid to form the text-to-vision heatmap. The heatmap captures the target-region awareness that has already emerged before coordinate generation and serves as the basis for subsequent connected-component selection.

Connected Component Selection. Given the two-dimensional text-to-vision heatmap, we obtain coarse target-region proposals via connected-component analysis. Specifically, we first threshold the heatmap with a fixed quantile threshold $q_{thr} = 0.90$, retaining the top 10% most responsive positions as foreground pixels. We then apply the 8-

neighbor connected-component algorithm to group adjacent foreground pixels into candidate regions.

For each connected component, we compute its area, total heat response, mean response, and minimum enclosing bounding box. Then the components are ranked by a score that combines response strength and region size:

$$s_c = \sum_{p \in c} h_p \cdot |c|^\alpha,$$

where h_p is the heatmap value at position p , $|c|$ is the component area, and $\alpha = 0.7$. This score favors regions that are both highly responsive and spatially coherent. The highest-scoring component, or the top- k components when multiple regions are used, is selected as the target-region proposal.

Bounding Box Expansion and Token Index Mapping. After selecting a connected component, we compute its minimum enclosing rectangle and use it as the base bounding box for the target region. Since the attention heatmap often covers only the most responsive part of a target widget, such as the icon center, a text fragment, or a small region inside a button, directly using this box may miss useful boundary and layout cues. We therefore expand the box with a fixed ratio to include the target boundary and nearby context. Specifically, given a box with width w and height h , we expand it by $\lfloor \max(w, h) \cdot r_{\text{pad}} \rfloor$ patch positions on each side, where $r_{\text{pad}} = 0.30$ in our implementation. The expanded box is clipped to the valid visual grid to avoid out-of-bound indices.

This expansion is used only as a fixed feature-retrieval heuristic, not as an additional learned module. The same expansion ratio is used across all experiments rather than being tuned for individual datasets or benchmarks. After expansion, we map the box back to token indices. For each position (r, c) in the two-dimensional visual grid, its flattened visual-token index is $r \times W + c$. By adding the start position v_{start} of the visual-token span in the input sequence, we obtain the corresponding target-region positions in the language-model sequence. The resulting token set is used both to retrieve fine-grained pre-merge visual features from the vision encoder and to identify the positions where KV-only evidence injection is applied.

Top- k Region Union. In some cases, target-related responses may be split into multiple local components, for example, when a widget contains both an icon and a text label, or when high responses appear around different parts of the target boundary.

To improve coverage, we optionally keep the top- k connected components according to the component score. By default, we use $k = 1$. For each selected component, we compute and expand its bounding box as described above. We then collect all token indices inside the expanded boxes, remove duplicates, sort them, and discard indices outside the visual-token span $[v_{\text{start}}, v_{\text{end}}]$.

The final unioned token set serves as the target-region token set. It connects the proposal stage with the following evidence pathway in two ways: it specifies where to retrieve fine-grained visual features from the vision encoder, and it determines which language-model positions receive KV-only evidence injection during coordinate decoding.

A.1.2 Slot Separation Regularization

The dual-slot workspace is designed to maintain two complementary evidence states, namely a *focus slot* and a *context slot*. However, similar initial query directions may cause the two slots to attend to similar visual patches and gradually converge to redundant representations during training. This slot collapse degenerates the dual-slot workspace into two nearly identical evidence vectors, weakening its ability to separate click-relevant local cues from contextual information. To mitigate this issue, we encourage slot diversity from both initialization and optimization.

Orthogonal Initialization. We first break the symmetry between the two slots through orthogonal initialization. The learnable slot query embeddings are initialized with approximately orthogonal directions, allowing the focus and context slots to start from distinct query biases and induce different attention patterns from the first forward pass. The initial cross-layer slot states are also initialized as small orthogonal vectors rather than zeros. This avoids identical early gated updates while keeping the newly introduced evidence pathway nearly transparent at the beginning of training.

A.1.3 Training Objective

We train the model in two stages. The first stage uses supervised fine-tuning to learn coordinate generation and stabilize the dual-slot evidence pathway. The second stage applies GRPO to directly optimize point-level grounding accuracy.

Supervised Fine-Tuning. During supervised fine-tuning, we optimize the model with the standard autoregressive cross-entropy loss and an auxiliary slot-separation regularizer. For each adapter in-

Table 12: SFT training hyperparameters.

Hyperparameter	Stage 1	Stage 2	Stage 3
Purpose	Evidence warm-up	Joint training	Decoder adaptation
Duration	0.032 epoch	0.160 epoch	0.808 epoch
Learning rate	5×10^{-6}	$3 \times 10^{-6} \rightarrow 3 \times 10^{-7}$	$2 \times 10^{-5} \rightarrow 2 \times 10^{-6}$
LR schedule	Cosine	Cosine	Cosine
Warmup ratio	0.10	0.05	0.05
Weight decay	0.0	0.01	0.01
Trainable modules	Adapter only	Adapter + decoder layers	Decoder + MLP + adapter
Max grad norm	1.0	1.0	1.0
Effective batch size	256	256	256
Compute resources	256 NVIDIA H20 GPUs	256 NVIDIA H20 GPUs	256 NVIDIA H20 GPUs

Table 13: RL training hyperparameters.

Hyperparameter	Value
Initialization	SFT checkpoint
Rollouts per prompt	8
	HuggingFace Transformers-
	based generation
Rollout engine	
Sampling temperature	1.0
Top- p	1.0
Actor micro-batch size	2 per GPU
Log-prob micro-batch size	2 per GPU
Actor strategy	FSDP2
Adapter LR multiplier	$1 \times$
Total epochs	1
Batch size	512
Compute resources	256 NVIDIA H20 GPUs

sertion layer $\ell \in \mathcal{L}_{\text{inj}}$, we apply the stop-gradient inner-product separation to the updated *focus* and *context* slots, denoted as $\mathcal{L}_{\text{z-sep}}^{(\ell)}$. The separation loss is averaged over all insertion layers:

$$\mathcal{L}_{\text{sep}} = \frac{1}{|\mathcal{L}_{\text{inj}}|} \sum_{\ell \in \mathcal{L}_{\text{inj}}} \mathcal{L}_{\text{z-sep}}^{(\ell)}. \quad (7)$$

Overall, the SFT objective is formulated as:

$$\mathcal{L}_{\text{SFT}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{sep}} \mathcal{L}_{\text{sep}}, \quad (8)$$

where $\mathcal{L}_{\text{CE}}$ is the autoregressive cross-entropy loss over the target sequence, and $\lambda_{\text{sep}} = 0.02$ controls the strength of the slot-separation regularization.

GRPO Optimization. After supervised fine-tuning, we further optimize the model with GRPO to directly improve point-level grounding accuracy as (Yang et al., 2026a; Zhou et al., 2025). For each input x , the policy samples a group of G responses $\{y_i\}_{i=1}^G$. Each response is parsed into a predicted coordinate (u_i, v_i) , and receives a binary grounding reward:

$$R_i = \mathbb{I}[(u_i, v_i) \in \mathcal{B}], \quad \hat{A}_i = \frac{R_i - \bar{R}}{\sigma_R + \epsilon_{\text{adv}}}, \quad (9)$$

Table 14: Fine-grained comparison on **OSWorld-G**. Green/purple indicates the best/second-best results within each scale group.

Agent Model	Text Matching	Element Recognition	Layout Understanding	Fine-grained Manipulation	Avg
2B/3B models					
Qwen3-VL-2B	61.7	45.8	54.2	39.6	45.9
Jedi-3B	67.4	53.0	53.8	44.3	50.9
MAI-UI-2B	62.8	56.7	59.3	40.3	52.0
InnerZoom-2B	62.1	59.7	59.3	39.5	53.7
Recent 4B/7B/8B models					
Qwen3-VL-8B	69.0	55.5	59.7	47.7	54.8
GTA1-7B	42.1	65.7	62.7	56.1	60.1
GUI-Owl-7B	64.8	63.6	61.3	41.0	55.9
UI-Venus-7B	74.6	60.5	61.5	45.5	58.8
MAI-UI-4B	78.8	68.6	68.6	57.8	60.3
InnerZoom-4B	84.2	75.5	74.1	64.4	64.7

where $\mathcal{B}$ denotes the ground-truth target bounding box, $\hat{A}_i$ is the group-normalized advantage, σ_R is the standard deviation of rewards within the group, and ϵ is a small constant for numerical stability. Responses that cannot be parsed into valid coordinates are assigned zero reward.

Following GRPO (Shao et al., 2024), we estimate group-wise advantages and optimize the policy with a clipped PPO-style objective (Schulman et al., 2017).

$$\mathcal{L}_{\text{GRPO}} = -\frac{1}{G} \sum_{i=1}^G \frac{1}{T_i} \sum_{t=1}^{T_i} \min \left(\rho_{i,t} \hat{A}_i, \text{clip}(\rho_{i,t}, 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}) \hat{A}_i \right). \quad (10)$$

where T_i is the response length, $\rho_{i,t} = \frac{\pi_{\theta}(y_{i,t}|x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t}|x, y_{i,<t})}$ is the token-level policy ratio, and ϵ_{clip} is the clipping threshold. This objective encourages responses whose predicted coordinates are more accurate than other samples in the same group, thereby aligning optimization with the final grounding outcome.

A.2 Implementation Details of Diagnostic Attention Intervention

We use attention intervention only for the diagnostic analysis in Fig. 1. The goal is to test whether the high-response regions identified from intermediate decoder layers contain decision-relevant evidence for final coordinate prediction. Given the original attention logits, we add a token-wise bias B before softmax. A positive bias increases the post-softmax attention weight of the corresponding tokens, while a negative bias suppresses it. Therefore, assigning positive bias to the selected region and negative bias to other visual tokens allows us to selectively amplify target-related evidence. We obtain the intervention region using the same ROI extraction procedure as in Sec. 3.1.

For the main intervention, we adopt a hard bias strategy. Tokens inside the selected target region receive $\beta_{\text{in}} > 0$, while tokens outside the region receive $\beta_{\text{out}} < 0$. In our diagnostic experiments, we set $\beta_{\text{in}} = 5.5$ and $\beta_{\text{out}} = -5.5$. To rule out the effect of generic attention perturbation, we further introduce a random attention intervention. Specifically, we randomly select the same number of visual tokens as the ROI tokens and treat them as a pseudo-target region, assigning them the same positive bias β_{in} while assigning β_{out} to the remaining tokens. Thus, the target-region-based intervention and the random intervention use the same amplification budget. The performance gap between them indicates whether the selected intermediate regions provide decision-relevant evidence rather than merely benefiting from arbitrary attention modulation.

A.3 Experiment Details

Data and Preprocessing. We follow the data construction, image preprocessing, coordinate normalization, and evaluation protocol of UI-Ins (Chen et al., 2026b) and MAI-UI (Zhou et al., 2025). All samples are converted into a unified GUI grounding format, where the model takes a GUI screenshot and a natural-language instruction as input and generates the target click coordinate.

Output Format. The model is trained to generate a complete grounding response, including grounding reasoning and the final coordinate answer. The final answer follows the JSON format $\{\text{"coordinate"}: [\text{x}, \text{y}]\}$, where the coordinate is normalized to the $[0, 1000]$ range along the image width and height.

SFT Training Framework. SFT is implemented

with HuggingFace Transformers and Accelerate (Wolf et al., 2019), together with DeepSpeed ZeRO-2 (Rajbhandari et al., 2020) for distributed training. We use bfloat16 precision and FlashAttention2 (Dao et al., 2022) to reduce memory cost under high-resolution GUI inputs, and enable gradient checkpointing for long-context training. During SFT, the proposed evidence pathway remains active, including the target-region proposal, iterative dual-slot evidence refinement, and KV-only evidence injection, allowing the model to learn the coordinate prediction with the same mechanism used at inference time.

Three-Stage SFT Schedule. To stabilize the newly introduced evidence pathway, we use a three-stage supervised fine-tuning schedule. First, we warm up the evidence pathway by mainly training the evidence adapter conditioned on the target-region proposal. This allows the model to adapt to target-region feature retrieval, dual-slot refinement, and KV-only evidence injection, while limiting perturbation to the backbone. Second, we jointly train the evidence adapter and selected decoder layers. This enables the decoder to better incorporate refined target-region evidence when generating grounding responses. In the third stage, we further adapt the decoder to improve its coordination prediction accuracy with the evidence pathway.

RL Training Framework. RL training is implemented with Verl (Sheng et al., 2025), Ray (Moritz et al., 2018), and FSDP2 (Zhao et al., 2023). For each prompt, the policy samples multiple rollouts and computes group-relative advantages from the binary click reward. We update the actor with the clipped GRPO objective, with the same evidence-injection pathway enabled during both rollout generation and policy updating. This stage directly optimizes point-level grounding under the target-box inclusion criterion, complementing the token-level imitation objective of SFT.

Hyperparameter Summary. Detailed hyperparameter settings for SFT and RL are summarized in Table 12 and Table 13, respectively.

A.4 More Experimental Results

A.4.1 Latency Measurement and TFLOPs Estimation.

For the accuracy–efficiency analysis, we compare Base SFT+RL, Base SFT+RL + Zoom-In, and InnerZoom under the same 4B scale-matched setting. Since different benchmarks have different

Table 15: Fine-grained comparison on **ScreenSpot-Pro**. Green/purple indicates the best/second-best results within each scale group. Larger models are shown in gray for reference.

Model	CAD		Dev.		Creative		Scientific		Office		OS		Avg.
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	
Larger models for reference													
Qwen3-VL-32B (Bai et al., 2025)	60.4	28.1	69.5	22.1	75.8	25.2	84.7	25.5	85.9	43.4	62.6	15.7	54.9
GUI-Owl-32B (Ye et al., 2025)	62.4	28.1	84.4	39.3	65.2	18.2	82.6	39.1	81.4	39.6	70.1	36.0	58.0
GTA1-32B (Yang et al., 2026a)	43.7	23.4	82.5	28.3	69.2	14.7	79.9	31.8	80.8	43.4	70.1	32.6	63.6
UI-Venus-72B (Gu et al., 2025)	66.5	29.7	84.4	33.1	73.2	30.8	84.7	42.7	83.1	60.4	75.7	36.0	61.9
Recent 4B/7B/8B models													
Qwen3-VL-8B (Bai et al., 2025)	46.7	10.9	79.2	23.4	68.2	14.0	73.6	30.0	76.3	30.2	65.4	21.3	49.9
GTA1-7B (Yang et al., 2026a)	53.3	17.2	66.9	20.7	62.6	18.9	76.4	31.8	82.5	50.9	48.6	25.9	50.1
UI-Venus-7B (Gu et al., 2025)	60.4	21.9	74.7	24.1	63.1	14.7	76.4	31.8	75.7	41.5	49.5	22.5	50.8
InfGUI-G1-7B (Liu et al., 2026b)	57.4	23.4	74.7	24.1	64.6	18.2	80.6	31.8	75.7	39.6	57.0	29.2	51.9
GUI-Owl-7B (Ye et al., 2025)	64.5	21.9	76.6	31.0	59.6	27.3	79.1	37.3	77.4	39.6	59.8	33.7	54.9
MAI-UI-4B (Zhou et al., 2025)	69.0	31.3	80.5	53.8	71.7	34.3	85.4	39.1	90.4	49.1	78.5	50.6	65.1
InnerZoom-4B	69.2	32.5	82.2	44.6	73.2	36.7	79.7	47.3	91.3	57.1	79.7	57.7	66.2

Table 16: Fine-grained comparison on **ScreenSpot-V2**. Green/purple indicates the best/second-best results within each scale group. Larger models are shown in gray for reference.

Model	Mobile		Desktop		Web		Avg.
	Text	Icon	Text	Icon	Text	Icon	
Larger models for reference							
GUI-Owl-32B	98.6	90.0	97.9	87.8	94.4	86.7	93.2
GTA1-32B	99.7	90.5	99.0	94.3	95.7	90.1	95.2
UI-Venus-72B	99.7	93.8	95.9	90.0	96.2	92.6	95.3
2B/3B models							
Qwen3-VL-2B	95.5	82.0	95.4	73.6	89.7	76.4	86.7
Jedi-3B	96.6	81.5	96.9	78.6	88.5	83.7	88.6
MAI-UI-2B	99.3	87.2	97.4	88.6	94.0	84.7	92.5
InnerZoom-2B	99.3	84.1	98.5	90.3	95.9	85.6	93.0
Recent 4B/7B/8B models							
Qwen3-VL-8B	97.9	84.8	95.9	87.9	95.7	83.7	91.7
GTA1-7B	99.0	88.6	94.9	89.3	92.3	86.7	92.4
GUI-Owl-7B	99.0	92.4	96.9	85.0	93.6	85.2	92.8
UI-Venus-7B	99.0	90.0	96.9	90.7	96.2	88.7	94.1
MAI-UI-4B	99.0	89.6	98.5	92.1	96.6	90.6	94.8
InnerZoom-4B	99.0	87.7	99.5	95.0	97.4	91.1	95.2

image resolutions, pixel budgets, and input/output token lengths, we report latency and TFLOPs as relative values normalized by the corresponding Base SFT+RL result on the same benchmark.

Latency measurement. Latency is measured as end-to-end milliseconds per sample, including image preprocessing, model generation, coordinate decoding, and output parsing. All timing results are obtained on a single NVIDIA H20 GPU with 96GB memory, using CUDA 12.3, PyTorch 2.3.0a0+ebedce2, and bfloat16 precision. We use the same decoding configuration as the main evaluation, with max_new_tokens=128, greedy decoding, and do_sample=False. Generation is performed sample by sample rather than with tensor-batched inference to ensure consistent per-sample timing.

For Base SFT+RL, latency corresponds to one forward generation pass. For Base SFT+RL + Zoom-In, latency includes the second zoom-in pass only when it is triggered, so the measured overhead reflects the actual trigger rate rather than a fixed

$2\times$ cost. For **InnerZoom**, latency includes all additional operations introduced by our method, including visual evidence extraction and cross-layer evidence adaptation. Relative latency is computed by dividing each method’s measured latency by the Base SFT+RL latency on the same benchmark.

TFLOPs estimation. TFLOPs are analytically estimated rather than measured with hardware profilers. We estimate the computational cost from the Qwen3-VL-4B architecture and the average vision pixels, input tokens, and output tokens of each benchmark. For Base SFT+RL, the estimate includes the vision encoder, language-model prefill, and autoregressive decoding. For Base SFT+RL + Zoom-In, we scale the single-pass cost by the expected number of forward passes, *i.e.*, one initial pass plus the measured zoom-trigger rate. For **InnerZoom**, the estimate includes the base forward computation and the additional costs of target region evidence extraction and the inserted cross-layer evidence adapters. Relative TFLOPs are computed by normalizing each method’s estimated TFLOPs by the Base SFT+RL TFLOPs on the same benchmark. This per-benchmark normalization accounts for dataset-specific differences in resolution, pixel budget, and sequence length.

A.4.2 More Fine-grained Results.

Tables 14, 15, and 16 provide additional fine-grained comparisons on OSWorld-G, ScreenSpot-Pro, and ScreenSpot-V2. These results further verify the effectiveness of **InnerZoom** across desktop interaction, high-resolution professional interfaces, and general GUI grounding scenarios.

On OSWorld-G, **InnerZoom-4B** achieves the best result across all four subcategories, including text matching, element recognition, layout understanding, and fine-grained manipulation. Compared with MAI-UI-4B, **InnerZoom** improves the

average score from 60.3 to 64.7, with clear gains in element recognition and layout understanding. This suggests that cross-layer evidence bridging helps preserve target-relevant visual evidence for both semantic recognition and layout-sensitive grounding. Under the 2B setting, **InnerZoom** also achieves the best average score, further showing that the proposed design remains effective under limited model capacity.

On ScreenSpot-Pro, **InnerZoom**-4B obtains the best average score of 66.2, surpassing MAI-UI-4B and larger reference models. It achieves the best result in 10 out of 12 text/icon subcategories and the best or second-best result in 11 out of 12 subcategories across CAD, development, creative, scientific, office, and OS. The improvements are especially clear on dense icon-based categories such as scientific and office, where precise grounding requires distinguishing visually similar neighboring elements. These results show that **InnerZoom** remains robust on high-resolution professional GUIs with dense layouts and small targets.

On ScreenSpot-V2, **InnerZoom**-4B achieves the best average result within the recent 4B/7B/8B group and remains comparable to much larger reference models. It performs particularly well on desktop and web splits, achieving the best results on Desktop Text, Desktop Icon, Web Text, and Web Icon. The 2B variant also achieves the best average score within the 2B/3B group, with strong performance on desktop and web categories. Overall, these results demonstrate that **InnerZoom** consistently improves GUI grounding across task types, resolutions, and interface styles by converting intermediate target-region evidence into more accurate point-level coordinate predictions.

A.4.3 Additional Qualitative Results.

Figs. 5 and 6 provide additional qualitative examples on SS-Pro and UI-V. The SS-Pro examples cover high-resolution professional interfaces with dense layouts, small icons, tool panels, and domain-specific controls. These cases are challenging because the target is often surrounded by visually similar neighboring elements, and the instruction may refer to a fine-grained tool, button, menu item, or functional region. As shown in Fig. 5, **InnerZoom** produces accurate point-level predictions in these crowded regions, suggesting that cross-layer evidence bridging helps preserve and reuse local visual cues. We also visualize the dilated union of the estimated target regions with red boxes. These

regions remain spatially compact while preserving sufficient local context around the predicted target, providing effective visual evidence for subsequent point-level grounding.

The UI-V examples further demonstrate the generality of **InnerZoom** across broader UI grounding scenarios. The targets include menu entries, text-formatting controls, debugging buttons, file-related operations, terminal regions, and layout-related elements. As shown in Fig. 6, **InnerZoom** generalizes to diverse grounding intents and interface layouts, indicating that the proposed evidence-bridging method is not limited to a specific UI type.

A.4.4 Failure analysis.

Fig. 7 summarizes representative failure cases. We observe three main types of remaining errors.

Type I: Semantically challenging instructions. Some failures arise from domain-specific or abstract instructions, such as locating software-specific options or modifying specialized tool settings. In these cases, the main difficulty is not only point-level localization, but also understanding the operational meaning of the instruction within a particular application. When the model lacks sufficient domain knowledge about the tool hierarchy or function semantics, the extracted local evidence may still be associated with an incorrect region.

Type II: Ambiguous text-to-image grounding. Another common failure occurs when the instruction is understandable, but the corresponding visual target is ambiguous. This often happens in dense professional interfaces, where multiple regions may partially match the instruction, or the target is small, low-contrast, or visually similar to nearby elements. Such ambiguity makes it difficult to determine which local evidence should be emphasized during coordinate decoding.

Type III: Misinterpreting on-screen text as user instructions. The model may also be distracted by salient on-screen text, especially when the interface contains instruction-like labels, placeholders, or menu items. In such cases, the model may incorrectly treat visible UI text as part of the user command, leading to attention on task-irrelevant regions. This suggests that future work may benefit from stronger disentanglement between user instructions and interface text, as well as more explicit modeling of actionable UI elements.

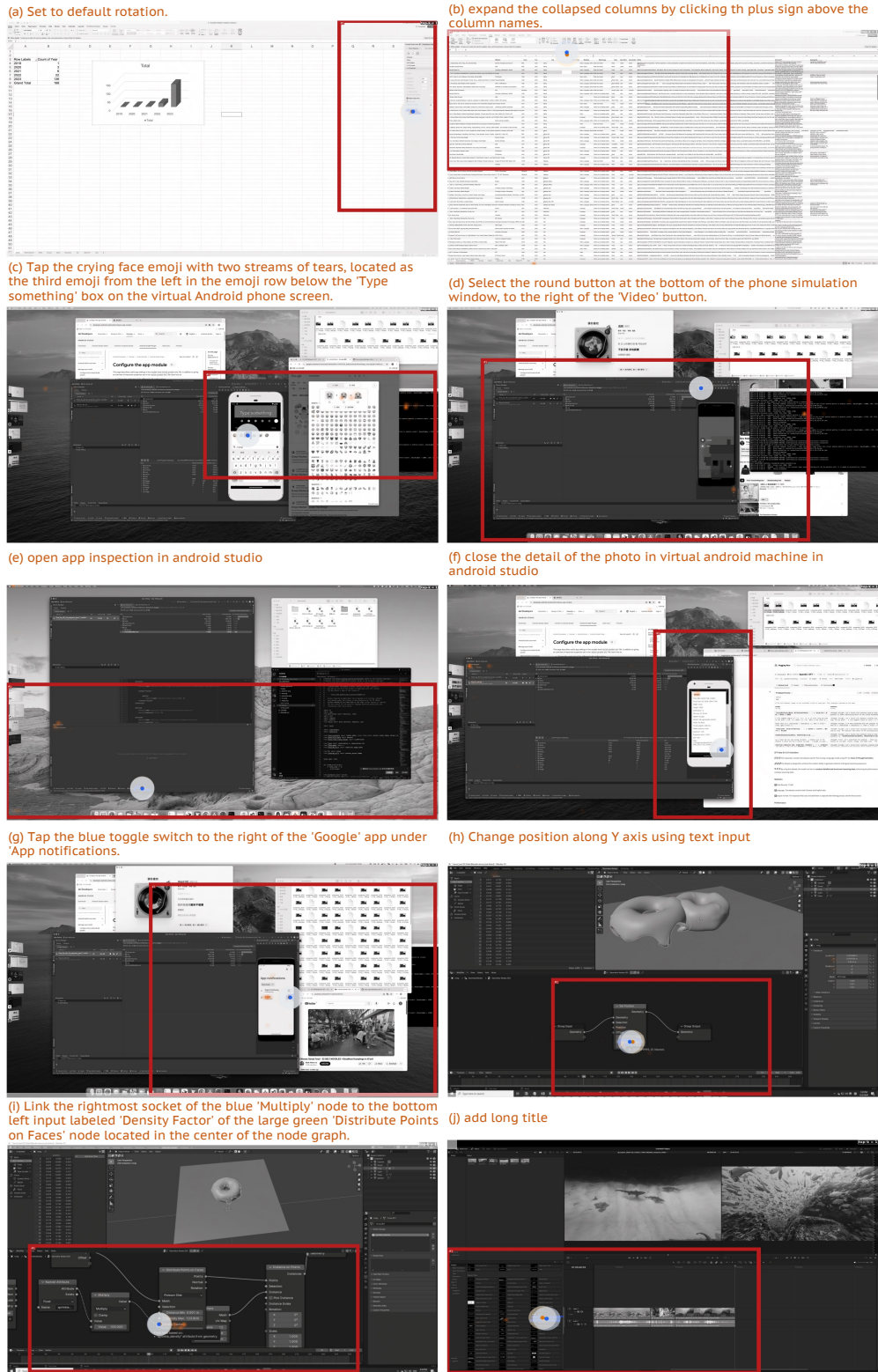


Figure 5: More visualizations on ScreenSpot-Pro. For better readability, we convert the visualization images to grayscale. The red boxes indicate the target regions identified by our method, the orange points denote the predicted results, and the blue points denote the ground-truth points.

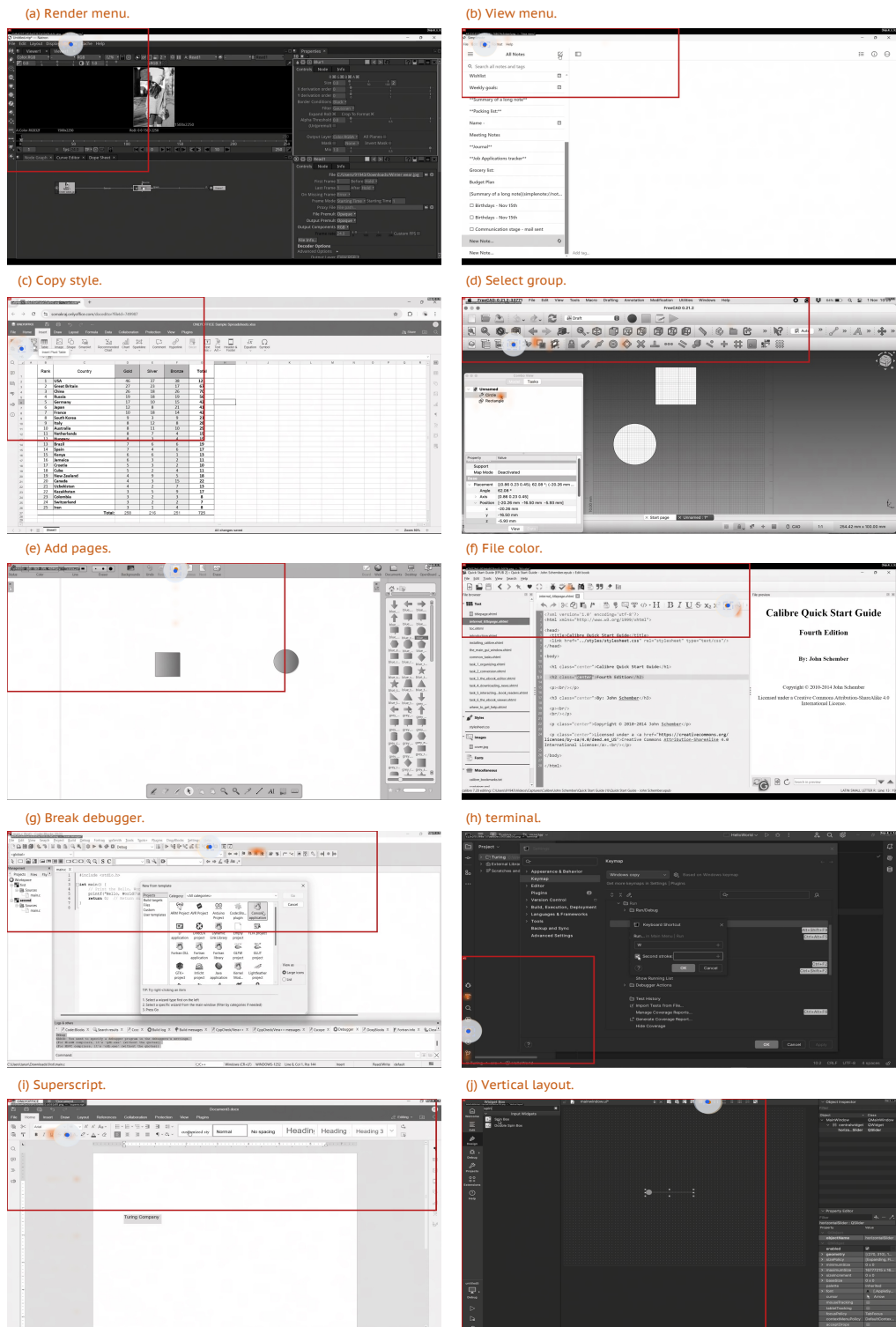
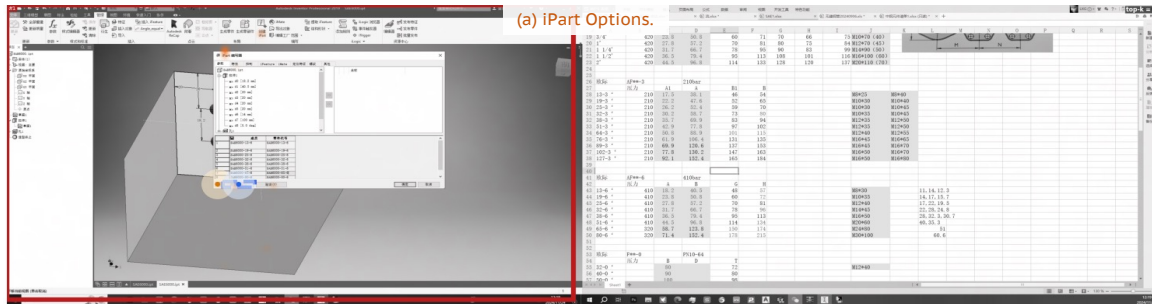
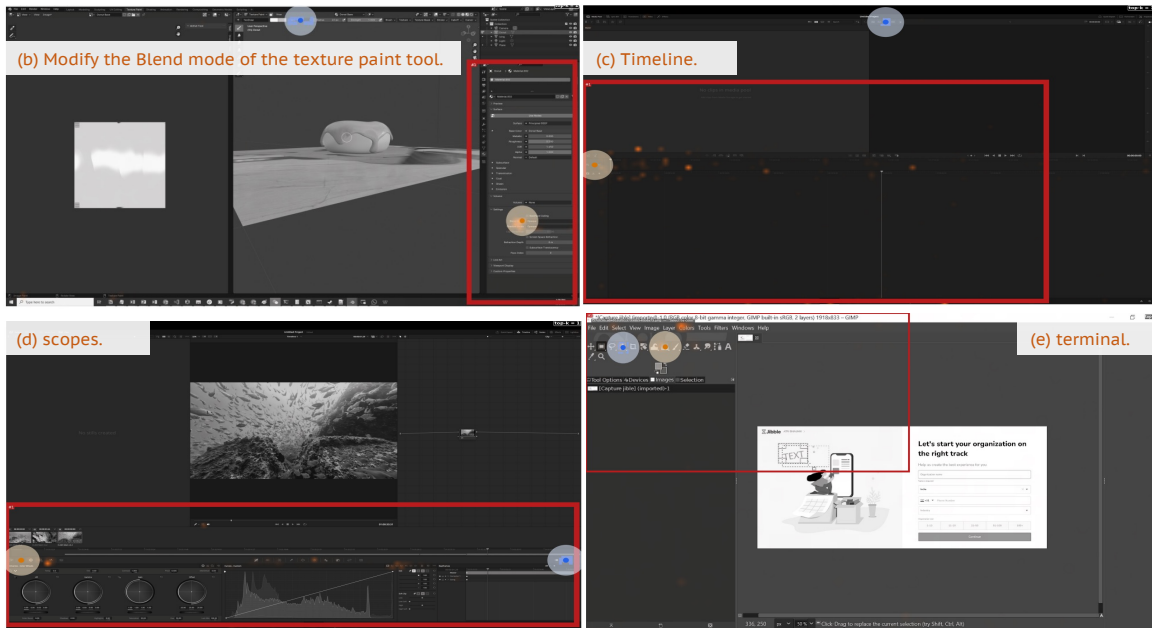


Figure 6: More visualizations on UI-Vision. For better readability, we convert the visualization images to grayscale. The red boxes indicate the target regions identified by our method, the orange points denote the predicted results, and the blue points denote the ground-truth points.

Error Type I: Semantically Challenging Instructions.



Error Type II: Ambiguous Text-to-Image Grounding.



Error Type Type III: Misinterpreting On-screen Text as Instructions.

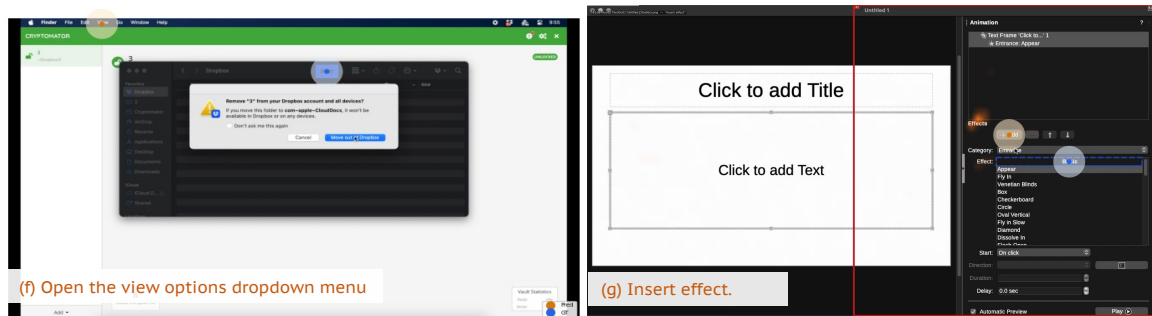


Figure 7: Failure case analysis. We identify three representative error types: **Type I: Semantically Challenging Instructions**, where abstract or indirect instructions make the intended target difficult to infer; **Type II: Ambiguous Text-to-Image Grounding**, where the instruction is understandable but cannot be clearly mapped to a unique visual element; and **Type III: Misinterpreting On-screen Text as Instructions**, where textual content in the screenshot distracts the model and is incorrectly treated as part of the user instruction. For better readability, screenshots are shown in grayscale. Red boxes indicate the target regions identified by our method, orange points denote predicted click locations, and blue points denote ground-truth points.