ZOOMSD: LEARNING WHERE TO ZOOM IN SELF-DISTILLED GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI grounding—locating the target element of a natural-language instruction on a screenshot—is a core capability of GUI agents. While training algorithms such as on-policy self-distillation (OPSD) have greatly improved grounding, models still struggle on high-resolution screenshots with small icons. Zoom-in inference mitigates this problem by cropping around a first-pass prediction and re-encoding the enlarged crop, but the closer view comes at a price: the model must first decode a coarse coordinate in an extra round of inference, roughly doubling the latency. This raises a natural question: is explicit coordinate generation necessary for deciding where to zoom? We demonstrate that a few attention heads involved in coordinate decoding already provide a useful coarse localization signal. Building on this observation, we propose ZoomSD, with two components. *ILR Crop Selection* applies bounded low-rank corrections (ILR) to the selected heads' attention logits and selects a fixed-size crop from the refined attention map. Second, *Joint Training with OPSD and Containment Ranking* couples this selector with OPSD through a containment ranking loss, with gradients isolated from the grounding backbone. At inference time, one full-image prefill selects the crop without autoregressive coordinate decoding; the enlarged crop is then used for final click generation. On Qwen3-VL-4B and Qwen3-VL-8B, ZoomSD substantially improves over the vanilla OPSD baseline on ScreenSpot-Pro, UI-Vision, and OSWorld-G, matching or even exceeding two-stage zoom-in accuracy while reducing its inference latency by 12.4–32.5%.

1 INTRODUCTION

GUI grounding—locating the screen element that a natural-language instruction refers to (Cheng et al., 2024; Li et al., 2025)—is the most fundamental capability of GUI agents, as every subsequent action presupposes that the correct target has been identified. Existing work improves grounding along two complementary axes: training the model better, and giving it an enlarged view of the target region at inference time. On the training side, reinforcement learning with rule-based rewards has substantially improved grounding accuracy (Lu et al., 2026; Luo et al., 2025; Liu et al., 2025; Wang et al., 2025), but it optimizes only a sparse reward wand requires expensive online rollouts to do so. On-policy self-distillation (OPSD) alleviates both issues by replacing the sparse reward with dense token-level supervision from a teacher with privileged hints, surpassing RL-based training at a fraction of the cost (Zhang et al., 2026a; Huang et al., 2026).

On the inference side, zoom-in methods exploit the fact that GUI targets are often tiny on the model's visual grid: after image preprocessing, most targets in high-resolution benchmarks span less than a single visual token (Section 3), so the model may find the right region yet miss the exact point. Zoom-in methods address this with a two-stage procedure (Jiang et al., 2025; Tang et al., 2026a; Li et al., 2025; Zhou et al., 2025; Xu et al., 2026): the model first predicts a coarse coordinate on the full screenshot, then re-encodes an enlarged crop around it and predicts the final coordinate from this closer view. The enlarged view allocates more visual tokens to the region and recovers many of these errors. Because it improves the input rather than the model, zoom-in complements training-side advances and remains effective even on top of strongly trained models.

Existing zoom-in inference, however, has two limitations. First, it relies on an indirect control loop: before obtaining a closer view, the model must autoregressively decode an explicit coordinate, crop

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.

the corresponding region, and repeat this process if a still closer view is needed. Each zoom therefore requires an additional round of inference—zoom-in roughly doubles the latency of standard inference, and multi-crop variants that aggregate several sub-images cost several times more (Zhang et al., 2026b). Second, where to zoom is never learned directly. Training-side methods improve the model's predictions, and thereby indirectly improve the crops built on top of them, but the crop itself is never an optimization target: no objective rewards a window for containing the target. These limitations raise a natural question: *is explicit coordinate generation necessary for deciding where to zoom, or can this capability be obtained from within the model itself—and further improved through learning?*

We show that it can. A model's attention natively encodes which region it attends to: even when the model fails to ground the exact coordinate, its attention can still outline the approximate region, providing a hint for where to zoom. Using the identification method described in Section 4.1, we find a small set of *localization heads* whose attention over image tokens consistently concentrates on the instructed target. This signal is available during a single full-image prefill, before any coordinate token is generated. Building on these heads, we propose ZoomSD (Figure 2), which computes an attention score for each image token from the localization heads, aggregates the scores over candidate windows, and selects the highest-scoring window for zoom-in. To better select the instruction-relevant region, we train this selector jointly with OPSD: Isolated Localization-Head Residuals (ILR), bounded low-rank corrections to the selected heads' attention logits, are optimized with a containment ranking objective that encourages a target-containing window to outrank non-containing alternatives, while isolated updates keep the localization objective from interfering with the backbone's distillation updates. At inference, one full-image prefill selects the window, and the same model generates the final click from the enlarged crop, eliminating the preliminary coordinate decoding of conventional zoom-in.

We validate ZoomSD on Qwen3-VL-4B and Qwen3-VL-8B across multiple GUI grounding benchmarks, including ScreenSpot-Pro, UI-Vision, and OSWorld-G. ZoomSD substantially improves grounding accuracy over the vanilla OPSD baseline on all benchmarks, and matches or even exceeds two-stage zoom-in while eliminating its preliminary coordinate decoding. These findings establish native localization signals as a practical route to zooming without decoding a preliminary coordinate. Our contributions are:

- We identify a small set of localization heads that play an important role in GUI grounding: their attention over image tokens provides a crop-selection signal before any coordinate is decoded.
- We propose ZoomSD, which consists of two parts: an ILR-refined crop selector that selects zoom-in windows from the identified localization heads' attention scores, and a training algorithm that augments vanilla OPSD with a containment ranking loss supervising the selector.
- Experiments on Qwen3-VL-4B and Qwen3-VL-8B show that ZoomSD substantially outperforms the vanilla OPSD baseline across multiple benchmarks and matches or even exceeds two-stage zoom-in without decoding a preliminary coordinate.

2 Related Work

GUI grounding. Specialist grounding models are trained on large collections of annotated interfaces (Cheng et al., 2024; Wu et al., 2025; Lin et al., 2025). Recent work improves grounding with reinforcement learning, typically using rule-based rewards that check whether the predicted point falls inside the target box (Lu et al., 2026; Luo et al., 2025; Liu et al., 2025; Wang et al., 2025). Complementary lines improve reliability through uncertainty calibration (Zhang et al., 2025a), autonomous exploration (Fan et al., 2025), and tool-augmented perception (Chen et al., 2026). Closer to our training recipe, on-policy self-distillation replaces sparse rewards with dense token-level supervision from a teacher with privileged hints (Zhang et al., 2026a; Huang et al., 2026). These methods improve the model's answer distribution; our work instead changes the visual input from which the answer is generated.

Zoom-in inference. Revisiting image regions at a higher scale has been explored for general vision-language models through guided visual search and dynamic zooming (Wu & Xie, 2024; Zhang et al., 2025b). For GUI grounding, two-stage zoom-in first predicts a coarse point on the full screenshot and then re-encodes an enlarged crop around it (Li et al., 2025; Jiang et al., 2025;

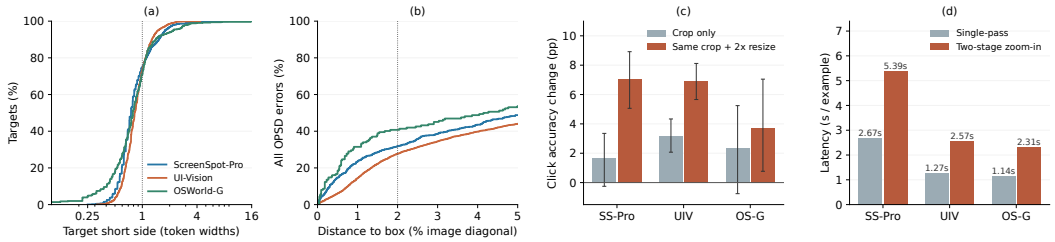


Figure 1: **Motivating a closer view of GUI targets.** **(a)–(c)** Analysis of a fixed OPSD-trained Qwen3-VL-4B model. **(a)** Cumulative target short-side extent on the merged visual-token grid after preprocessing; the dotted line marks one token width. **(b)** Fraction of all incorrect predictions within a given distance of the target box; the dotted line marks our 2% near-miss threshold. **(c)** Accuracy changes relative to full-image inference when revisiting the same predicted crop, re-encoded at its original pixel size or resized to the original image dimensions. **(d)** Per-example latency of single-pass inference and two-stage zoom-in on Qwen3-VL-4B, measured with HuggingFace Transformers on 6×H20 GPUs, batch size 1, and greedy decoding. Error bars are paired 95% bootstrap confidence intervals clustered by screenshot.

Tang et al., 2026a), a strategy also adopted by recent agent systems (Zhou et al., 2025; Xu et al., 2026). Other methods aggregate predictions from multiple sub-images (Zhang et al., 2026b). All of these approaches obtain the region through a preliminary decoded coordinate or additional inference passes. In contrast, ZoomSD reads the region directly from the model's native attention during a single full-image prefill.

Attention-based localization signals. Intrinsic attention has been used as a localization signal in several ways: a few attention heads in a frozen vision-language model already support grounding (Kang et al., 2025); GUI-Actor (Wu et al., 2026) and GUI-AIMA (Zhou et al., 2026) train attention-based modules to replace coordinate decoding for the final prediction; and Lin et al. (2026) show that the prefill stage largely determines the selected element, exploiting it with a training-free second prefill. ZoomSD differs from these uses in three ways: it keeps the generative answer pathway and uses attention only for the intermediate crop decision, rather than replacing coordinate decoding (Wu et al., 2026; Zhou et al., 2026); it learns to refine the native signal for crop containment instead of using it as-is (Kang et al., 2025; Lin et al., 2026); and its second view is an enlarged crop that adds detail resolution, rather than a re-encoded full image (Lin et al., 2026).

3 Motivating Analysis

We examine the behavior of a fixed OPSD-trained grounding model (Qwen3-VL-4B) and its existing predictions on ScreenSpot-Pro (Li et al., 2025), UI-Vision (Nayak et al., 2025), and OSWorld-G (Xie et al., 2025). Three observations follow.

Targets are small, and errors are local. After the model's image preprocessing, 71–74% of targets in ScreenSpot-Pro, UI-Vision, and OSWorld-G have a shorter side spanning less than one merged visual-token width (Figure 1a), and 31.8%, 27.7%, and 40.7% of errors on the three benchmarks fall within 2% of the image diagonal from the target box (Figure 1b). The model therefore often finds the approximate region of the target but fails to click the exact point.

A closer view helps only when the crop is enlarged. Re-encoding the predicted crop at its original pixel size improves over full-image inference by only 1.6 and 3.2 percentage points on ScreenSpot-Pro and UI-Vision, whereas resizing the same crop to the original image dimensions before re-encoding yields gains of 7.0 and 6.9 points (Figure 1c). Resizing allocates more visual tokens to the region, and it is this added resolution—rather than revisiting alone—that recovers many of the near-miss errors.

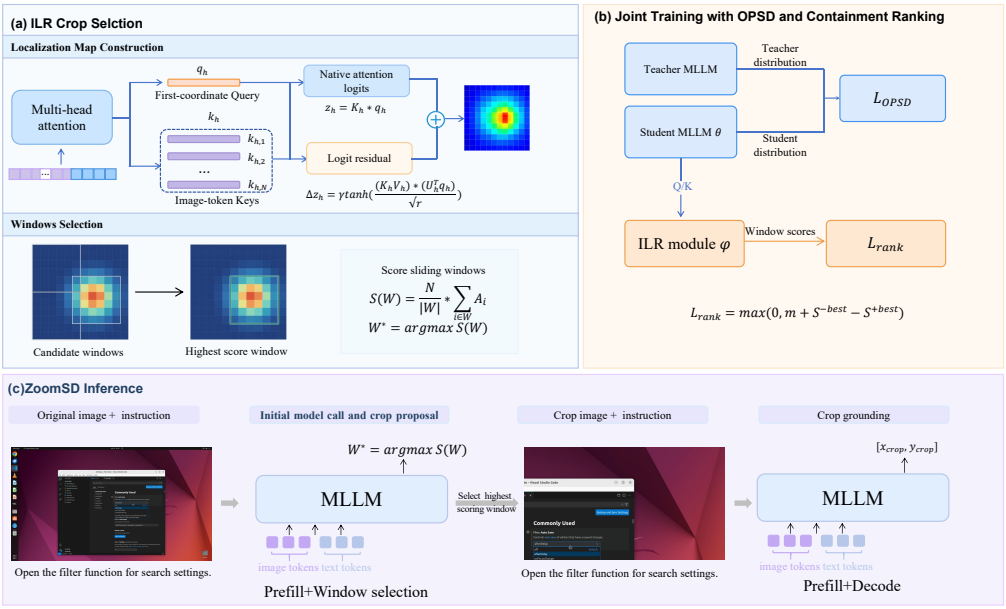


Figure 2: **Overview of ZoomSD. (a)** A full-image prefill selects a region before any coordinate is generated; the same model predicts the final click from the enlarged crop. **(b)** ILR refines the native localization heads' logits, and the resulting spatial map scores candidate crop windows. **(c)** OPSD trains the student from a teacher with privileged visual hints, while containment ranking trains the crop selector. The selector reads detached query and key features, so its loss updates only the residual parameters.

The closer view comes at a price. Obtaining the enlarged view conventionally requires an additional round of inference whose first stage autoregressively decodes a coarse coordinate, costing 102–103% more latency than single-pass inference across the three benchmarks (Figure 1d); variants that aggregate multiple sub-images cost several times more. Zoom-in is thus worth having, but its region should not have to be decoded.

4 METHOD

Given a screenshot I and an instruction u, GUI grounding predicts a click on the target element. Let θ denote the parameters of the grounding model and B the target bounding box available during training. ZoomSD uses the model's native attention to select a crop before generating any coordinate, then predicts the final click from the enlarged crop. After identifying localization heads on the training set, our method has two components: *ILR Crop Selection*, which learns to refine their spatial signal, and *Joint Training of OPSD and Containment Ranking*, which trains crop selection alongside on-policy self-distillation without passing localization gradients into the backbone. Figure 2 summarizes inference and training.

4.1 IDENTIFYING LOCALIZATION HEADS

A subset of native attention heads concentrates on the instructed target during coordinate prediction. We identify these *localization heads* by probing the pretrained model on annotated training examples. For each head, we measure the fraction of its image attention that falls within the target bounding box, accounting for partial overlap between visual tokens and the box. We average this score across the training examples and retain the top-K heads, denoted by $\mathcal{H}$; the full selection protocol, the selected heads, and a sensitivity analysis of K are given in Appendix D. Their identities remain fixed throughout joint training and evaluation. Two analyses in the experiments validate this choice: steering interventions show that these heads are causally necessary for grounding rather than a correlational artifact (Section E), and ablations show that their native signal alone already

yields substantial gains, on top of which the learned residual brings further consistent improvements (Section 5.3).

To use these heads without decoding a preliminary coordinate, we append a fixed prefix that ends immediately before the first coordinate token. A single prefill on the screenshot, instruction, and prefix provides a query $q_h \in \mathbb{R}^d$ at the final prefix position and image-token keys $\{k_{h,i}\}_{i=1}^N$ for each $h \in \mathcal{H}$. The prefix contains no coordinate values. The native image-token logits are

$$z_{h,i} = s\, k_{h,i}^\top q_h, \qquad h \in \mathcal{H}, \tag{1}$$

where s is the backbone's attention scaling and N is the number of visual tokens. These logits provide a coarse spatial prior for crop selection.

4.2 ILR Crop Selection

Native localization signals do not always concentrate enough mass on a region containing the complete target. We therefore introduce *Isolated Localization-Head Residuals* (ILR) to adapt the crop selector while retaining the native attention prior. For each selected head, we add a bounded low-rank correction, in the spirit of low-rank adapters (Hu et al., 2022):

$$\Delta z_{h,i} = \gamma \tanh\left(\frac{(k_{h,i}^\top V_h)(U_h^\top q_h)}{\sqrt{r}}\right), \qquad U_h, V_h \in \mathbb{R}^{d\times r}, \tag{2}$$

The rank r controls the correction's capacity, and γ bounds its magnitude. We initialize one factor randomly and the other to zero so that crop selection initially follows the native prior. The residual parameters are $\phi = \{U_h, V_h\}_{h\in\mathcal{H}}$; during training they are the only parameters updated by the ranking loss, since the residual reads detached backbone features (Section 4.3). The corrected logits are used only by the crop selector; they do not alter the backbone's attention computation or answer generation.

We normalize the corrected logits over image tokens within each head and then average across heads:

$$a_{h,i} = \frac{\exp(z_{h,i} + \Delta z_{h,i})}{\sum_{j=1}^N \exp(z_{h,j} + \Delta z_{h,j})}, \qquad A_i = \frac{1}{K}\sum_{h\in\mathcal{H}} a_{h,i}. \tag{3}$$

Reshaping A to the visual-token grid yields a spatial map. Let $\mathcal{W}$ be the set of windows obtained by sliding a fixed-size rectangle over this grid, with its dimensions determined by a fixed fraction of the image size. We score each window by its normalized attention mass and select the highest-scoring one:

$$S_\phi(W) = \frac{N}{|W|}\sum_{i\in W} A_i, \qquad W^* = \arg\max_{W\in\mathcal{W}} S_\phi(W). \tag{4}$$

Here $|W|$ is the number of visual tokens in W; the normalization gives a uniform attention map a score of one across grid resolutions.

At inference, we map W^* to the original image, crop and enlarge the selected region, and use the same grounding model to generate the final click. The predicted coordinate is then mapped back to the original image. Thus, crop selection requires one full-image prefill, followed by a crop prefill and final answer generation, eliminating the initial coordinate decoding of conventional zoom-in inference.

4.3 Joint Training of OPSD and Containment Ranking

We train the grounding model and crop selector together using complementary objectives. On-policy self-distillation improves the model's answer distribution, while containment ranking teaches the selector to preserve the complete target within a fixed crop budget.

On-policy self-distillation. We follow GUI-SD (Zhang et al., 2026a). The student generates a response $y \sim p_\theta(\cdot \mid I, u)$ from the unmodified screenshot and instruction. A teacher with parameters $\bar{\theta}$ receives an additional privileged visual hint $v(I, B)$ that highlights the target during training. At each response position t, both models are conditioned on the same student-generated prefix $y_{<t}$:

$$p_t = p_\theta(\cdot \mid I, u, y_{<t}), \qquad \bar{p}_t = p_{\bar{\theta}}(\cdot \mid v(I, B), u, y_{<t}). \tag{5}$$

The teacher therefore supervises predictions along the student's own trajectories. The distillation objective averages the token-level divergence over the sampled responses:

$$\mathcal{L}_{\mathrm{OPSD}} = \frac{1}{T}\sum_{t=1}^{T} D_{\mathrm{KL}}(p_t \,\|\, \mathrm{sg}(\bar{p}_t))\,, \tag{6}$$

where T is the response length and $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation. The sampled responses and teacher distributions are held fixed when differentiating the loss. The teacher is maintained as an exponential moving average of the student, and privileged hints are used only during training.

Containment ranking. The crop selector is trained to prefer windows that contain the entire target. Let $C(W)$ denote the pixel region corresponding to window W. Using the same crop geometry as at inference, we partition the candidates into

$$\mathcal{W}^+ = \{W \in \mathcal{W} : B \subseteq C(W)\}, \qquad \mathcal{W}^- = \mathcal{W} \setminus \mathcal{W}^+. \tag{7}$$

For each example with both positive and negative candidates, we define

$$\ell_{\mathrm{rank}} = \left[m + \max_{W\in\mathcal{W}^-} S_\phi(W) - \max_{W\in\mathcal{W}^+} S_\phi(W)\right]_+, \tag{8}$$

where $m > 0$ is a margin and $[x]_+ = \max(0, x)$. The loss encourages the best containing window to outrank the strongest non-containing alternative, directly matching the selector's argmax decision. Examples without a positive–negative pair contribute zero, and $\mathcal{L}_{\mathrm{rank}}$ is the mean over the batch. Bounding boxes provide crop supervision but are never inputs to the selector.

Isolated updates. The two objectives are deliberately assigned to disjoint parameter sets, for two reasons. First, the ranking loss should not interfere with answer learning: the backbone's grounding ability is shaped by distillation alone, exactly as in vanilla GUI-SD. Second, if the ranking loss also updated the backbone, we would have to retain all intermediate activations of the backbone forward pass for backpropagation, which is prohibitively memory-intensive; with detached features, the selector's loss requires no stored backbone activations and its updates add negligible cost. Formally, let $F_\theta(I, u)$ denote the native Q/K features extracted with the fixed coordinate-free prefix. The two objectives update disjoint parameter sets:

$$g_\theta = \nabla_\theta \mathcal{L}_{\mathrm{OPSD}}, \qquad g_\phi = \nabla_\phi \mathcal{L}_{\mathrm{rank}}\big(\phi; \mathrm{sg}(F_\theta(I, u)), B\big). \tag{9}$$

We apply these gradients with separate optimizers. The stop-gradient interface ensures that containment supervision updates only ϕ, while the backbone receives only the OPSD gradient. Conversely, the residual is external to the answer model and receives no distillation gradient. As θ evolves, we recompute the selector features from the current student, allowing the residual to adapt to the changing localization prior. This jointly improves answer learning and crop selection while preserving the backbone's OPSD update rule.

5 Experiments

5.1 Experimental Setup

Models. We experiment with Qwen3-VL-4B-Instruct and Qwen3-VL-8B-Instruct (Qwen Team, 2025).

Training data. We directly use the GUI-SD training set (Zhang et al., 2026a), which is built on ScaleCUA (Liu et al., 2026b). Localization heads are identified once on 3,000 annotated training examples and remain fixed throughout training and evaluation.

Benchmarks and metric. We evaluate on ScreenSpot-Pro (Li et al., 2025), UI-Vision (Nayak et al., 2025), and OSWorld-G (Xie et al., 2025); benchmark details are provided in Appendix B. We report click accuracy: a prediction is correct if the predicted point falls inside the ground-truth box.

Table 1: **Results on ScreenSpot-Pro.** Results of existing models are taken from the MAI-UI report (Zhou et al., 2025) and are shown for reference; the bottom six rows are evaluated with our own protocol. ZoomSD rows are shaded; **bold** marks the best result in each column among the base model, the GUI-SD baseline, and ZoomSD within each model series.

	CAD		Dev.		Creative		Scientific		Office		OS		
Model	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Avg.
Existing open-source models													
InfiGUI-3B (Liu et al., 2025)	50.8	25.0	64.9	20.0	51.5	16.8	68.8	32.7	70.6	32.1	49.5	19.7	45.2
UI-TARS-7B (Qin et al., 2025)	20.8	9.4	58.4	12.4	50.0	9.1	63.9	31.8	63.3	20.8	30.8	16.9	35.7
GUI-Actor-7B (Wu et al., 2026)	47.7	9.4	59.1	15.9	59.6	16.1	70.1	25.5	69.5	41.5	55.1	19.1	44.6
GUI-G^2-7B (Tang et al., 2026b)	55.8	12.5	68.8	17.2	57.1	15.4	77.1	24.5	74.0	32.7	57.9	21.3	47.5
GTA1-7B (Yang et al., 2026)	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
InfiGUI-G1-7B (Liu et al., 2026a)	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-8B (Zhou et al., 2025)	72.6	35.9	83.8	52.4	76.3	33.6	79.9	37.3	88.7	60.4	76.6	49.4	65.8
Larger models for reference													
Qwen3-VL-32B (Qwen Team, 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., 2026)	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
Qwen3-VL-4B	52.8	21.9	82.5	31.0	71.2	21.7	77.1	34.6	83.1	39.6	**79.4**	36.0	56.7
+ GUI-SD	57.4	31.3	83.8	40.7	69.7	26.6	79.9	37.3	89.3	45.3	78.5	41.6	60.5
+ ZoomSD (Ours)	**75.1**	**45.3**	**89.6**	**49.0**	**73.2**	**37.1**	**84.0**	**45.5**	**92.1**	**64.2**	**79.4**	**60.7**	**69.0**
Qwen3-VL-8B	59.4	17.2	78.6	26.9	68.2	17.5	75.7	27.3	81.4	41.5	71.0	24.7	53.8
+ GUI-SD	68.5	26.6	83.1	42.8	68.7	23.8	79.9	35.5	85.3	52.8	75.7	36.0	60.6
+ ZoomSD (Ours)	**80.7**	**34.4**	**89.0**	**49.0**	**78.8**	**35.7**	**84.7**	**40.9**	**88.1**	**69.8**	**79.4**	**49.4**	**68.6**

Baselines. We compare with existing open-source grounding models, whose results are taken from the MAI-UI report (Zhou et al., 2025). For controlled comparisons, all of our rows are trained and evaluated under the same protocol: the base model with direct full-image inference; the GUI-SD model trained with vanilla OPSD; ZoomSD without ILR, which selects windows with uncorrected native attention; and the full ZoomSD. For the efficiency analysis, we additionally compare with conventional two-stage zoom-in at the same jointly trained checkpoint, which decodes a coarse coordinate before re-encoding an enlarged crop, using the same crop geometry and enlargement as ZoomSD.

Training details. All models are trained on 6×H20 GPUs, with one card dedicated to rollout generation and five to training; the effective batch size is 80 and training runs for 2 epochs. For ILR, we use $K{=}10$ localization heads, rank $r{=}8$, and bound $\gamma{=}2$. Candidate windows have side lengths equal to half of the image dimensions (about 25% of the image area), and the selected crop is resized to the original image size before re-encoding.

5.2 Main Results

5.2.1 Results on ScreenSpot-Pro

Table 1 reports ScreenSpot-Pro results. ZoomSD improves substantially over both the base model and the GUI-SD baseline at each scale. On Qwen3-VL-4B, ZoomSD raises the average accuracy from 56.7 to 69.0, an 8.5-point gain over the GUI-SD model trained with the same data and budget; on Qwen3-VL-8B, the average rises from 53.8 to 68.6, an 8.0-point gain over GUI-SD. At both scales, ZoomSD surpasses all existing models of comparable size, including the recent MAI-UI-8B (65.8), and also exceeds the larger 32B–72B reference models. The gains concentrate on the icon split, where targets are smallest: on Qwen3-VL-4B, icon accuracy improves from 30.0 (base) and 36.3 (GUI-SD) to 48.2, with the largest domain-level gain on CAD icons (21.9 to 45.3).

Table 2: **Results on UI-Vision and OSWorld-G.** Results of existing models are taken from the MAI-UI report (Zhou et al., 2025); our rows are evaluated with our own protocol. ZoomSD rows are shaded; **bold** marks the best result in each column among the base model, the GUI-SD baseline, and ZoomSD within each model series.

	UI-Vision				OSWorld-G				
Model	Basic	Func.	Spatial	Avg.	Text Match	Elem. Recog.	Layout Und.	Fine Manip.	Avg.
Existing open-source models									
UGround-7B (Gou et al., 2025)	15.4	17.1	6.3	12.9	51.3	40.3	43.5	24.8	36.4
UI-TARS-7B (Qin et al., 2025)	20.1	24.3	8.4	17.6	60.2	51.8	54.9	35.6	47.5
UI-TARS-1.5-7B (Seed, 2025)	28.8	27.5	10.7	22.3	36.8	62.7	62.2	50.8	52.8
UI-Venus-7B (Gu et al., 2025)	36.1	32.8	11.9	26.5	74.6	60.5	61.5	45.5	58.8
UI-TARS-72B (Qin et al., 2025)	31.4	30.5	14.7	25.5	69.4	60.6	62.9	45.6	57.1
UI-Venus-72B (Gu et al., 2025)	45.6	42.3	23.7	36.8	82.1	71.2	70.7	64.4	70.4
MAI-UI-8B (Zhou et al., 2025)	51.7	49.6	22.5	40.7	72.0	63.3	66.0	51.0	60.1
Qwen3-VL-4B	30.8	32.3	9.2	23.7	78.8	57.3	34.8	41.5	65.1
+ GUI-SD	41.4	40.6	16.8	32.4	80.8	60.7	30.4	31.7	66.5
+ ZoomSD (Ours)	**47.1**	**44.7**	**24.0**	**38.2**	**83.3**	**67.0**	**43.5**	**46.3**	**72.0**
Qwen3-VL-8B	30.4	30.6	13.7	24.6	78.3	54.4	30.4	46.3	63.9
+ GUI-SD	42.9	40.7	18.9	33.8	**80.8**	62.6	**43.5**	**48.8**	69.2
+ ZoomSD (Ours)	**48.1**	**46.8**	**26.5**	**40.1**	79.2	**68.4**	30.4	43.9	**69.8**

5.2.2 Results on UI-Vision and OSWorld-G

Table 2 reports results on UI-Vision and OSWorld-G. On UI-Vision, ZoomSD improves over GUI-SD by 5.8 and 6.3 points on average at the 4B and 8B scales, respectively. The largest gains appear on spatial instructions, which require resolving the target among visually similar distractors: spatial accuracy rises from 16.8 to 24.0 on Qwen3-VL-4B and from 18.9 to 26.5 on Qwen3-VL-8B. On OSWorld-G, ZoomSD improves over GUI-SD by 5.5 points on Qwen3-VL-4B, with gains on all four categories—most strongly on element recognition (+6.3) and layout understanding (+13.1)—and reaches the highest average (72.0) among all compared models. On Qwen3-VL-8B, where the GUI-SD baseline is already strong (69.2), ZoomSD remains comparable on average (69.8) while improving element recognition by 5.8 points.

5.3 Ablation Study

Figure 3 isolates the contribution of the ILR residual. Two observations stand out. First, uncorrected native attention already recovers most of the gain: on Qwen3-VL-4B, selecting crops with native heads alone lifts ScreenSpot-Pro from 60.5 to 67.0, confirming that the model's native localization signal carries a usable crop-selection prior before any coordinate is decoded. Second, the ILR residual improves crop containment on every benchmark at both scales—most notably on UI-Vision at 8B, where containment rises from 69.2 to 77.5—and these containment gains translate into click-accuracy gains in most cases. The exception is UI-Vision at 4B, where containment improves by 3.2 points while accuracy stays flat: inspection shows that many UI-Vision targets are small text labels whose crops, once correctly contained, are still misclicked within the enlarged view, so better selection alone cannot convert. Overall, the residual adds 0.1–3.1 accuracy points on top of the native signal with only about 20K additional parameters ($K{=}10$ heads, rank $r{=}8$) and no change to the backbone.

5.4 Inference Cost

Table 3 compares inference cost and accuracy at the same Qwen3-VL-4B checkpoint. ZoomSD matches two-stage zoom-in on UI-Vision and OSWorld-G and exceeds it by 0.7 points on ScreenSpot-Pro, while being consistently faster: removing the first stage's autoregressive coordinate decoding saves 12.4–32.5% latency relative to two-stage zoom-in (per-stage token and latency breakdown in Appendix C). The saving grows as screenshots get smaller and the preliminary decoding dominates the first stage, reaching 32.5% on OSWorld-G. Both zoom-based methods are slower than single-pass inference, since they re-encode a second view; ZoomSD closes this overhead to 37–77% on top of single-pass while recovering nearly all of the zoom-in accuracy, and the margin over decoding-based selection would widen further for methods that aggregate multiple sub-images.

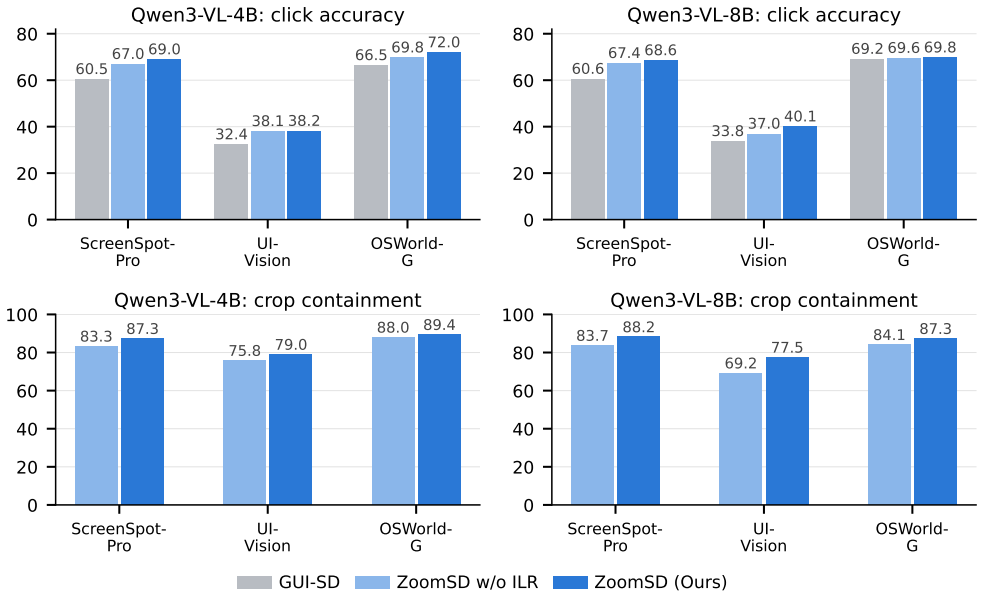


Figure 3: **Effect of the ILR residual.** Top: click accuracy; bottom: crop containment rate (the fraction of selected crops that fully contain the target; GUI-SD has no crop stage and is omitted). Selecting crops with uncorrected native attention already recovers most of the accuracy gain, and the ILR residual further improves containment on every benchmark at both scales.

Table 3: **Inference cost and accuracy on Qwen3-VL-4B**, all at the same jointly trained checkpoint. Latency is the average per-example wall-clock time, measured with HuggingFace Transformers inference on 6×H20 GPUs, batch size 1 (examples processed serially, data-parallel across cards), and greedy decoding; crop resizing time is included. *Saved* is the latency reduction of ZoomSD relative to two-stage zoom-in. **Bold** marks the best accuracy in each row.

	Single-pass		ZoomSD (Ours)		Two-stage zoom-in		
Benchmark	Acc.	Time	Acc.	Time	Acc.	Time	Saved
ScreenSpot-Pro	60.5	2.67s	**69.0**	4.72s (+77%)	68.3	5.39s (+102%)	12.4%
UI-Vision	32.4	1.27s	**38.2**	1.91s (+50%)	**38.2**	2.57s (+102%)	25.7%
OSWorld-G	66.5	1.14s	**72.0**	1.56s (+37%)	**72.0**	2.31s (+103%)	32.5%

Absolute latencies depend on the inference stack; the relative savings are inherent to the method, as they stem from eliminating one autoregressive stage rather than from implementation details.

6 Conclusion

We asked whether a GUI grounding model must decode an explicit coordinate before deciding where to zoom. Our analysis answers no: a small set of native localization heads already carries a usable crop-selection signal before any coordinate token is generated, and steering experiments show this signal is causally necessary for grounding rather than a correlational artifact. Building on it, ZoomSD selects zoom-in windows directly from the heads' attention over image tokens, refines the signal with lightweight ILR residuals trained under a containment ranking objective, and learns both objectives jointly with OPSD through isolated parameter updates. On Qwen3-VL-4B and Qwen3-VL-8B, ZoomSD substantially improves over the vanilla OPSD baseline on ScreenSpot-Pro, UI-Vision, and OSWorld-G, matching or exceeding two-stage zoom-in accuracy while eliminating its preliminary coordinate decoding and saving 12.7–34.4% inference latency. More broadly, our results suggest that native attention signals are a practical substrate for intermediate visual decisions in GUI agents, and that such decisions can be learned alongside—without interfering with—answer-level training.

Ethics Statement

Our work studies GUI grounding, the perceptual component of computer-use agents, using only publicly available benchmarks (ScreenSpot-Pro, UI-Vision, OSWorld-G) and publicly released training data (GUI-SD, built on ScaleCUA) and open-weight models (Qwen3-VL). We use each dataset and model in accordance with its license and intended research use, and our experiments involve no human subjects and collect no new personal data. GUI screenshots may occasionally contain incidental user content captured by the original dataset curators; we do not augment or re-distribute such content beyond the datasets' own distribution channels. More capable GUI agents could in principle be misused to automate unwanted interactions with software systems; our contribution concerns the accuracy and efficiency of localization on static screenshots and does not by itself enable new end-to-end automation capabilities, but we encourage downstream users to deploy grounding models with appropriate safeguards and human oversight.

Reproducibility Statement

The experimental setup, including models, training data, benchmarks, baselines, and hyperparameters, is described in Section 5. The benchmark details, including evaluation-set composition and metric conventions, are provided in Appendix B. We will release the training and evaluation code, the probed localization-head identities, and the trained model checkpoints upon publication.

AI Use Statement

We used AI assistants to support language polishing of the manuscript and to assist with the development and debugging of the training and evaluation code. No AI system was used to generate research ideas, design experiments, or produce results, and no AI-generated content is presented without human verification. The authors reviewed and edited all text and code and take full responsibility for the content of this paper.

References

Chen Chen, Jiawei Shao, Dakuan Lu, Haoyi Hu, Xiangcheng Liu, Hantao Yao, and Wu Liu. Gui-eyes: Tool-augmented perception for visual grounding in gui agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29350–29358, 2026.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.505/.

Yue Fan, Handong Zhao, Ruiyi Zhang, Yu Shen, Xin Eric Wang, and Gang Wu. Gui-bee: Align gui action grounding to novel environments via autonomous exploration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33249–33266, 2025.

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*, volume 2025, pp. 30851–30883, 2025.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. Ui-venus technical report: Building high-performance ui agents with rft. *arXiv preprint arXiv:2508.10833*, 2025.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jingyuan Huang, Zuming Huang, Yucheng Shi, Tianze Yang, Xiaoming Zhai, Wei Chu, and Ninghao Liu. Trust the right teacher: Quality-aware self-distillation for gui grounding. *arXiv preprint arXiv:2606.18101*, 2026.

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. *arXiv preprint arXiv:2512.05941*, 2025. URL https://arxiv.org/abs/2512.05941.

Seil Kang, Jinyeong Kim, Junhyeok Kim, and Seong Jae Hwang. Your large vision-language model only needs a few attention heads for visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2503.06287.

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 *Proceedings of the 33rd ACM International Conference on Multimedia*, 2025. doi: 10.1145/3746027.3755688. URL https://arxiv.org/abs/2504.07981.

Jiaping Lin, Fei Shen, Junzhe Li, Ping Nie, Fei Yu, Ming Li, and Haizhou Li. What happens before decoding? prefill determines gui grounding in vlms. *arXiv preprint arXiv:2605.12549*, 2026.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19498–19508. IEEE, 2025.

Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*, 2025.

Yuhang Liu, Zeyu Liu, Shuanghe Zhu, Pengxiang Li, Congkai Xie, Jiasheng Wang, Xueyu Hu, Xiaotian Han, Jianbo Yuan, Xinyao Wang, et al. Infigui-g1: Advancing gui grounding with adaptive exploration policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32267–32275, 2026a.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. Scalecua: Scaling open-source computer use agents with cross-platform data. In *International Conference on Learning Representations*, volume 2026, pp. 62707–62766, 2026b.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, et al. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17608–17616, 2026.

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. *arXiv preprint arXiv:2504.10458*, 2025.

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. UI-Vision: A desktop-centric GUI benchmark for visual perception and interaction. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2503.15661.

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

Qwen Team. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

ByteDance Seed. Ui-tars-1.5. https://seed-tars.com/1.5, 2025.

Fei Tang, Bofan Chen, Zhengxi Lu, Tongbo Chen, Songqin Nong, Tao Jiang, Wenhao Xu, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Ui-zoomer: Uncertainty-driven adaptive zoom-in for gui grounding. *arXiv preprint arXiv:2604.14113*, 2026a.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, et al. Gui-g^2: Gaussian reward modeling for gui grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33214–33222, 2026b.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13084–13094. IEEE, 2024.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *Advances in Neural Information Processing Systems*, 38:15101–15128, 2026.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: Foundation action model for generalist gui agents. In *International Conference on Learning Representations*, volume 2025, pp. 5090–5108, 2025.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Haoyuan Wu, Jixuan Chen, Wenjing Chen, 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*, 2025. URL https://arxiv.org/abs/2505.13227.

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

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, et al. Gta1: Gui test-time scaling agent. In *International Conference on Learning Representations*, volume 2026, pp. 48295–48323, 2026.

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

Shaojie Zhang, Pei Fu, Ruoceng Zhang, Jiahui Yang, Anan Du, Xiuwen Xi, Shaokang Wang, Ying Huang, Bin Qin, Zhenbo Luo, et al. Hyperclick: Advancing reliable gui grounding via uncertainty calibration. *arXiv e-prints*, pp. arXiv–2510, 2025a.

Xintong Zhang, Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaowen Zhang, Yang Liu, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, et al. Adaptive chain-of-focus reasoning via dynamic visual search and zooming for efficient vlms. *arXiv preprint arXiv:2505.15436*, 2025b.

Yan Zhang, Daiqing Wu, Huawen Shen, Yu Zhou, and Can Ma. Learn where to click from yourself: On-policy self-distillation for GUI grounding. *arXiv preprint arXiv:2605.00642*, 2026a. URL https://arxiv.org/abs/2605.00642.

Yunzhu Zhang, Zeyu Pan, Zhengwen Zeng, Shuheng Shen, Changhua Meng, and Linchao Zhu. Mvp: Multiple view prediction improves gui grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27482–27492, 2026b.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025.

Shijie Zhou, Viet Dac Lai, Hao Tan, Jihyung Kil, Wanrong Zhu, Changyou Chen, and Ruiyi Zhang. Gui-aima: Aligning intrinsic multimodal attention with a context anchor for gui grounding. In *European Conference on Computer Vision*, pp. 79–97. Springer, 2026.

A Additional Method Details

Training hyperparameters. Joint training uses two isolated optimizers, one for the backbone (OPSD) and one for the ILR parameters (containment ranking); their settings are listed in Table 4. The backbone optimizer follows the GUI-SD recipe: AdamW with $\beta{=}(0.9, 0.95)$, learning rate $2.5{\times}10^{-6}$, cosine decay with 5% warmup, and weight decay 0.1. The ILR optimizer is a separate AdamW over the residual parameters only, kept in FP32: learning rate 10^{-3} with linear warmup over 5% of the steps followed by cosine decay, weight decay 0.01, and default PyTorch momentum $(0.9, 0.999)$. Both use gradient clipping at norm 1.0. Because the residual reads detached query/key features (Section 4.3), the ranking loss never updates backbone parameters, and the OPSD loss never updates the residual.

Table 4: **Training hyperparameters.** Left: backbone optimizer for OPSD. Right: isolated ILR optimizer for containment ranking. Bottom: OPSD rollout/teacher settings and ILR objective settings, shared by both model scales.

	Backbone (OPSD)	ILR residual
Optimizer	AdamW	AdamW
Learning rate	2.5×10^{-6}	1×10^{-3}
Momentum β	$(0.9,\ 0.95)$	$(0.9,\ 0.999)$
Schedule	cosine	linear warmup + cosine
Warmup	5% of steps	5% of steps
Weight decay	0.1	0.01
Grad. clip	1.0	1.0
Precision	BF16	FP32
OPSD side		
Teacher update	EMA of the student, decay 0.95 per step	
Rollouts per prompt	8	
Sampling	temperature 0.9, top-p 0.9, top-k 50, max 128 tokens	
ILR side		
Margin m / bound γ / rank r	0.05 / 2.0 / 8	
Initialization	$U_h \sim \mathcal{N}(0,\ 1/\sqrt{d})$, $V_h = 0$ (exactly-zero residual)	
Trainable parameters	20,480 ($K{=}10$ heads, $d{=}128$, $r{=}8$)	

Compute. Training runs on 6×H20 GPUs, one dedicated to vLLM rollout generation and five to training. The effective batch size is 80 (4B: per-device 16; 8B: per-device 8 with 2 gradient-accumulation steps) over the 6,247 training examples for 2 epochs, i.e. 158 optimizer steps. Wall-clock time is approximately 7.4 hours for Qwen3-VL-4B (≈44 H20 GPU-hours) and 10.2 hours for Qwen3-VL-8B (≈61 H20 GPU-hours) per run, including on-policy rollouts. Inputs are capped at 10,000 merged image tokens and 12,288 total sequence tokens.

Joint versus sequential ILR training. In our main experiments the backbone (OPSD) and the ILR selector are trained jointly under isolated updates (Section 4.3). We also evaluated a sequential variant: take an already OPSD-trained GUI-SD checkpoint, freeze it, and train the ILR parameters alone with the same containment ranking objective, data, and hyperparameters. This post-hoc stage requires only a single detached-feature prefill per training example—no on-policy rollouts and no teacher forward passes. Table 5 shows that sequential training matches joint training within 0.7 accuracy points on every benchmark, with no consistent ordering between the two variants. This is consistent with the isolation design: because the ranking loss reads detached features and updates only the residual parameters, coupling its optimization with OPSD provides no measurable benefit, and ILR can be retrofitted onto an existing GUI-SD checkpoint without re-running the OPSD stage.

B Benchmark Details

ScreenSpot-Pro (Li et al., 2025) contains 1,581 instructions over high-resolution screenshots from professional applications, organized into six domains (CAD, Development, Creative, Scientific, Office, OS), each split into text and icon targets.

Table 5: **Joint versus sequential ILR training** (click accuracy). *Joint* is the main ZoomSD model; *sequential* trains ILR post-hoc on the frozen OPSD-trained checkpoint. The two variants are within 0.7 points on all benchmarks.

Model	ILR training	ScreenSpot-Pro	UI-Vision	OSWorld-G
Qwen3-VL-4B	Joint (main)	69.0	38.2	72.0
	Sequential	69.2	37.9	71.3
Qwen3-VL-8B	Joint (main)	68.6	40.1	69.8
	Sequential	68.5	40.2	69.6

UI-Vision (Nayak et al., 2025) contains 5,479 desktop examples with three instruction types: basic (1,772), functional (1,772), and spatial (1,935).

OSWorld-G (Xie et al., 2025) provides 564 desktop grounding examples. We evaluate on the 510 non-refusal examples, which fall into four categories: text matching (240), element recognition (206), layout understanding (23), and fine-grained manipulation (41). Four-number polygon annotations are interpreted as xyxy boxes; we use the corrected repository metric consistently across all views.

C Inference Cost Breakdown

Table 6 breaks the inference cost of Table 3 into prefill and decode tokens per stage, averaged over the same 100-example subsets used for the latency measurements (Qwen3-VL-4B, greedy decoding). Prefill tokens are exact prompt lengths recorded during the measured runs; decode tokens are the exact lengths of the recorded responses. Since the selected crop is resized to the original image size before re-encoding, the crop prompt has the same visual-token count and text as the first-stage prompt.

Table 6: **Per-stage token breakdown of inference** (mean per example, Qwen3-VL-4B). Stage 1 is the full-image pass; stage 2 is the enlarged-crop pass. Prefill tokens are processed in parallel in one forward pass, whereas each decode token requires its own sequential forward pass. *Decode %* is the share of decode tokens in the stage's total tokens.

		Stage 1 (full image)		Stage 2 (crop)		
Benchmark	Method	Prefill	Decode	Prefill	Decode	Decode %
ScreenSpot-Pro	Single-pass	5,877	33.7	—	—	0.57
	ZoomSD (Ours)	5,900	0	5,877	33.6	0.28
	Two-stage zoom-in	5,877	33.7	5,877	33.8	0.57
UI-Vision	Single-pass	2,399	33.5	—	—	1.38
	ZoomSD (Ours)	2,422	0	2,399	33.8	0.70
	Two-stage zoom-in	2,399	33.5	2,399	33.9	1.39
OSWorld-G	Single-pass	1,916	33.7	—	—	1.73
	ZoomSD (Ours)	1,939	0	1,916	33.8	0.87
	Two-stage zoom-in	1,916	33.7	1,916	33.9	1.73

Three observations explain where the latency saving of Table 3 comes from. First, inference is overwhelmingly prefill-dominated: decode tokens are only about 34 per generation, 0.3–1.7% of all tokens. Second, ZoomSD and two-stage zoom-in process nearly identical token counts—ZoomSD's first stage decodes nothing, while two-stage decodes about 34 coordinate tokens—so the saving does not come from processing fewer tokens. It comes from replacing the first stage's $\approx$34 *serial* autoregressive steps, each a full forward pass over the KV cache, with attention readout inside the single *parallel* prefill that ZoomSD already performs. Third, the relative saving is largest where the first-stage prompt is smallest (OSWorld-G, 1,916 prefill tokens, 34.4% saved), because the eliminated serial decoding constitutes a larger fraction of the first stage's latency.

Table 7: **Per-stage latency** (mean seconds per example, Qwen3-VL-4B, same protocol and subsets as Table 3). *Stage 1* produces the crop decision from the full image; *stage 2* generates the final answer from the enlarged crop. ZoomSD's stage 1 is one prefill with attention readout; two-stage zoom-in's stage 1 additionally decodes ≈34 coordinate tokens. The stage-1 difference (Δ) is the coordinate-decoding phase; the crop stages are identical up to noise.

	ZoomSD (Ours)			Two-stage zoom-in			
Benchmark	Stage 1	Stage 2	Total	Stage 1	Stage 2	Total	Δ Stage 1
ScreenSpot-Pro	1.98	2.74	4.72	2.67	2.72	5.39	0.69
UI-Vision	0.57	1.34	1.91	1.27	1.30	2.57	0.70
OSWorld-G	0.40	1.16	1.56	1.14	1.17	2.31	0.74

Per-stage timing (Table 7) confirms this decomposition directly. ZoomSD's first stage—a single prefill with attention readout—takes 1.98s, 0.57s, and 0.40s on the three benchmarks, while two-stage zoom-in's first stage takes 2.67s, 1.27s, and 1.14s. The difference is almost exactly the coordinate-decoding phase: 0.69s, 0.70s, and 0.74s for ≈34 tokens (≈21 ms per serial step), nearly constant across benchmarks despite a 3× difference in prompt length, because each decode step re-reads the full model weights regardless of context size. The crop stages match on both sides (within 0.03s), as both pipelines re-encode the same enlarged crop. In other words, coordinate-based selection adds 26–65% latency on the selection stage itself—a share that grows as screenshots shrink—and ZoomSD removes it entirely.

We emphasize that the eliminated decoding is the *entire* selection-specific overhead: the full-image prefill and the enlarged-crop pass are mandatory for any zoom-in method and are identical in both pipelines, so the ≈34 serial decode steps are the only latency that two-stage zoom-in pays for coordinate-based crop selection. Relative to that overhead, ZoomSD's reduction is total—attention-based selection adds no serial step on top of the mandatory prefill. The 12.7–34.4% end-to-end figure of Table 3 is this complete elimination normalized by the full two-stage latency, and the gap would widen for agents that decode longer reasoning before producing a coordinate, whose selection-stage decoding is correspondingly longer.

D Localization-Head Selection Protocol

This appendix specifies the complete protocol used to identify the localization heads $\mathcal{H}$ (Section 4.1): the probe data, the exact scoring rule, the choice of K, the selected heads, and the layer-wise distribution of the scores.

Probe data. Heads are probed on the small-target subset of the GUI-SD training set that is also used for joint training: the 6,247 of 6,990 training examples whose ground-truth box covers less than 1% of the image area. The subset contains 3,850 icon and 2,397 text instructions over Windows (4,845), Ubuntu (1,238), and macOS (164) screenshots. We draw 3,000 examples uniformly at random (seed 0); the realized draw contains 1,876 icon (62.5%) and 1,124 text (37.5%) instructions, with 2,349 Windows, 585 Ubuntu, and 66 macOS screenshots. No benchmark data is used for selection at this stage (see the remark on the 4B list below).

Scoring rule. For each probed example, we teacher-force the ground-truth tool-call response and let $\mathcal{D}$ be the response positions that decode the coordinate digits (the two numbers inside `"coordinate": [x, y]`; 3–6 tokens in practice). Let $a^{(h)}_{d,\cdot}$ be the attention row of head h at position d, renormalized over the N image tokens: $\tilde{a}^{(h)}_{d,i} = a^{(h)}_{d,i} / \sum_{j=1}^{N} a^{(h)}_{d,j}$. Each image token i corresponds to a square cell c_i of $\tau \times \tau$ pixels in the resized image, with τ=32 (patch size 16, spatial merge 2). Rather than a binary in/out test or an IoU, we weight each token by the fraction of its cell covered by the ground-truth box B,

$$w_i = \frac{|c_i \cap B|}{\tau^2} \in [0, 1], \tag{10}$$

because targets on high-resolution GUIs are frequently smaller than a single 32-pixel cell, where a center-in-box rule would assign zero credit to every token. The score of head $h = (l, h')$ is the

Table 8: **Attention steering on the base Qwen3-VL-4B** (ScreenSpot-Pro, 1,500 evaluated examples, greedy decoding; the 81 examples with prompts over 9,000 tokens are skipped, and the 95 responses per condition without a parseable coordinate are counted as incorrect, matching the main evaluation protocol). We intervene on the top-10 probed localization heads during coordinate-token decoding: *knockout* zeros their image attention; *knockout (random)* applies the same operation to 10 random heads; *sharpen* sharpens their image-attention distribution (τ=0.5) without using ground truth; *oracle* adds ground-truth-box mass to estimate the headroom of the attention-to-coordinate pathway.

Intervention	Text	Icon	Overall	Δ
None	72.6	28.1	55.9	–
Knockout	56.8	16.0	41.5	−14.4
Knockout (random heads)	72.4	28.1	55.8	−0.1
Sharpen	72.7	29.9	56.7	+0.7
Oracle	79.0	38.4	63.8	+7.9

overlap-weighted in-box attention ratio, averaged over digit positions and over the M=3,000 probe examples:

$$\mathrm{score}(h) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{|\mathcal{D}_m|}\sum_{d\in\mathcal{D}_m}\sum_{i=1}^{N_m}\tilde{a}_{d,i}^{(h)}\,w_i. \tag{11}$$

Under a uniform attention map this score equals the mean box coverage $\bar{w} = \frac{1}{M}\sum_m \frac{1}{N_m}\sum_i w_i \approx 0.0011$ on our probe set; the best heads reach ≈ 0.19, more than $170\times$ the uniform baseline. Equation equation 11 yields a 36×32 score matrix per model (text-decoder layers only, 0-indexed), from which the top-K heads are taken.

E Analysis: Localization Heads Are Causally Necessary

We further analyze whether the localization signal our method exploits is causally load-bearing. We intervene on the top-10 probed heads of the base model—the same heads that ZoomSD reads from—during coordinate decoding on ScreenSpot-Pro (Table 8). Three findings emerge. First, the heads are necessary and specific: zeroing their image attention drops accuracy by 14.4 points (p=4×10^{-49}, two-sided exact McNemar test on paired per-example outcomes), while the same intervention on ten random heads has no significant effect (-0.1, p=0.73). Second, the readout pathway from attention to coordinates is precise: biasing these heads toward the ground-truth box raises accuracy by 7.9 points (p=2×10^{-24}) and pulls predictions to within a median of 4/1000 normalized units of the target center, indicating that substantial headroom exists for supervision that corrects *where* these heads look. Third, the bottleneck is location rather than sharpness: sharpening the existing distribution without ground truth yields only a non-significant 0.7-point gain (p=0.19). Together with the probing observation, these results motivate our design: refining the native localization signal with containment supervision (ILR) addresses exactly the failure mode—looking at the wrong place—that limits the pretrained model.

Choice of K. We set K=10 based on two observations, summarized in Figure 4. First, the sorted head scores decay smoothly without a sharp cut-off, but the head at rank 10 still scores $10.5\times$ above the all-head average (0.126 vs. 0.012 on Qwen3-VL-4B), so K=10 captures the heads that are clearly separated from the bulk. Second, downstream accuracy is insensitive to K in this regime: evaluating the jointly trained model with $K \in \{10, 20, 30\}$ changes click accuracy by at most 0.7 points on each benchmark, with no monotone trend. We therefore keep the smallest K past the score elbow, which also keeps the ILR parameter count at about 20K.

Selected heads. Table 9 lists the top-10 heads of each model with their scores. Two properties are worth noting. First, the two models agree closely: 8 of the 10 (layer, head) pairs are shared, and the full 36×32 score matrices correlate at Spearman ρ=0.89. Second, all selected heads lie in layers 17–23 of 36 (relative depth 0.47–0.64), a middle-layer band consistent with the heatmaps in Figure 5.

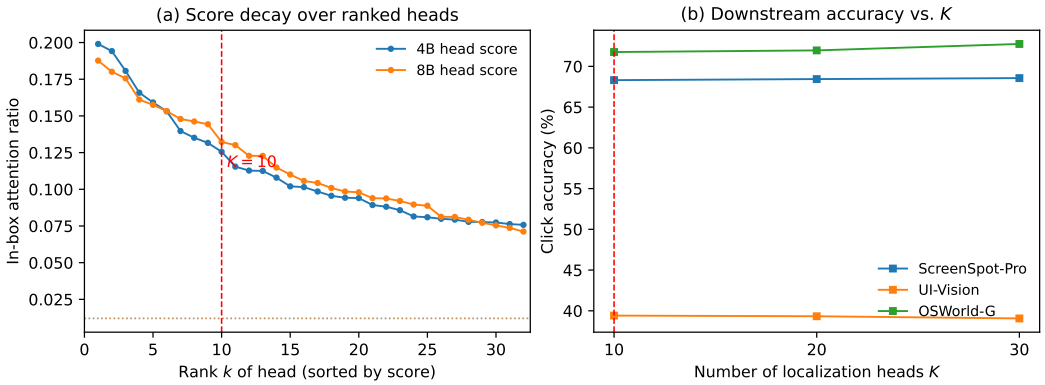


Figure 4: **Sensitivity to the number of localization heads K.** (a) Overlap-weighted in-box attention ratio (Eq. equation 11) of the k-th ranked head for Qwen3-VL-4B and Qwen3-VL-8B; dotted lines mark the all-head averages. The decay is smooth and the rank-10 head remains an order of magnitude above the average head. (b) Click accuracy of the jointly trained model as a function of K on the three benchmarks; accuracy varies by at most 0.7 points with no monotone trend, so we fix K=10.

Table 9: **Selected localization heads** (0-indexed text-decoder layers and heads) with their probe scores (Eq. equation 11), for Qwen3-VL-4B and Qwen3-VL-8B.

	Qwen3-VL-4B		Qwen3-VL-8B	
Rank	(layer, head)	Score	(layer, head)	Score
1	(21, 8)	0.199	(21, 10)	0.188
2	(21, 10)	0.194	(21, 11)	0.180
3	(21, 11)	0.181	(21, 8)	0.176
4	(20, 15)	0.166	(20, 15)	0.161
5	(17, 4)	0.159	(18, 15)	0.158
6	(23, 30)	0.153	(17, 4)	0.153
7	(18, 15)	0.140	(22, 4)	0.148
8	(22, 4)	0.135	(23, 13)	0.146
9	(23, 13)	0.132	(23, 30)	0.144
10	(18, 19)	0.126	(21, 18)	0.132

Layer-wise distribution. Figure 5 visualizes the full score matrices. Localization heads are not scattered uniformly: high scores concentrate in a contiguous middle-layer band (roughly layers 16–24, relative depth 0.44–0.67) in both models, while early layers (0–11) and late layers (28–35) show near-baseline scores. Within the band, high-scoring heads spread across head indices rather than aligning to a particular key–value group, and the band location is nearly identical across the two model scales, echoing the cross-model agreement of the selected heads.

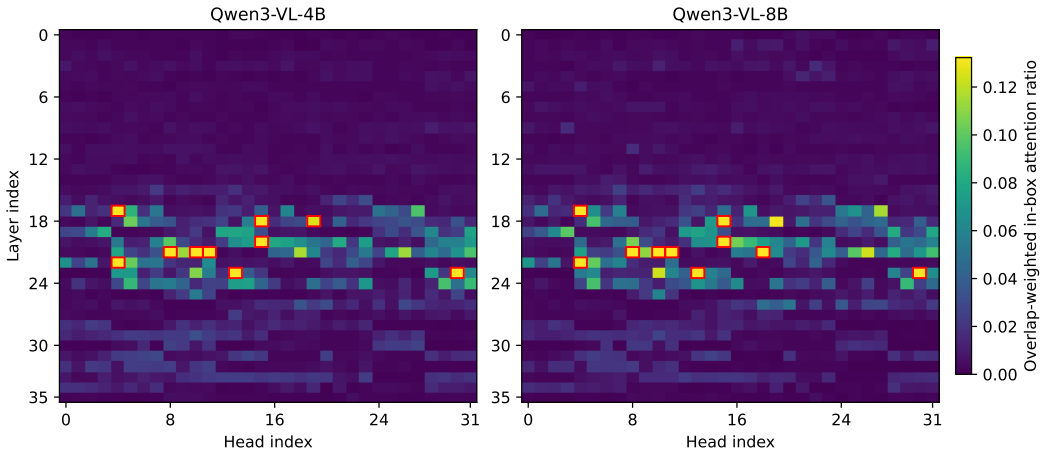


Figure 5: **Probe score of every (layer, head)** for Qwen3-VL-4B (left) and Qwen3-VL-8B (right), measured by the overlap-weighted in-box attention ratio of Eq. equation 11 on 3,000 training examples. Red boxes mark the top-10 heads of each model. High scores concentrate in a shared middle-layer band (layers 17–23 of 36, relative depth 0.47–0.64); the two models' score matrices correlate at Spearman ρ=0.89.