RECURRENT LATENT VISUAL SEARCH FOR GUI GROUNDING

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI grounding is a critical capability for GUI agents powered by vision-language models, helping them execute user instructions by locating the corresponding elements in screenshots. Single-step grounding struggles with small elements and dense layouts, motivating multi-step visual search. However, existing approaches commonly rely on textual reasoning misaligned with visual space or costly multi-round interactions with external visual tools. To make multi-step visual search an explicit spatial process within the model, we propose ***ReLaViS***, which performs ***Re**current **La**tent **Vi**sual **S**earch* in a single interaction round. At each step, a spatial search head uses the hidden state to query the screenshot's visual tokens, producing a spatial search distribution that explicitly represents the search focus. This distribution then aggregates the visual tokens into latent visual evidence, which is recurrently fed back as the next input embedding to condition subsequent search. We further introduce a GUI-aware coarse-to-fine inductive bias through trajectories constructed from flat element annotations, supervising search from the global interface through intermediate element groups to the target. Built on Qwen2.5-VL-7B, ReLaViS improves ScreenSpot-Pro accuracy by 3.1 percentage points to 56.3% with only a 3.5% increase in inference FLOPs and outperforms the matched single-step baseline on all five benchmarks.

1 INTRODUCTION

Advances in large language models (LLMs) and vision-language models (VLMs) enable GUI agents to understand user instructions and operate software interfaces (Anthropic, 2024; Wang et al., 2024; OpenAI, 2025). GUI grounding, which locates the interactive element specified by an instruction, is a core capability underlying actions such as clicking (Yang et al., 2025; Yuan et al., 2025). Prevailing approaches formulate GUI grounding as single-step localization, directly predicting a coordinate, bounding box, or spatial distribution (Xie et al., 2025; Tang et al., 2026b; Wu et al., 2025b). Despite strong performance on standard interfaces, they still struggle with high-resolution screens, small targets, and dense layouts, where accurate grounding requires integrating global structure, local appearance, and instruction semantics (Li et al., 2025b; Feizi et al., 2026). To address these challenges, recent work improved grounding through multi-step visual search, including inspecting local regions (Tang et al., 2026a; Liu et al., 2026a) or revising predictions (Zhao et al., 2026; Wang et al., 2026b) before returning a final location.

A key design choice is how to represent search states and update them across steps. *Thinking about images* expresses target semantics and spatial reasoning through chain of thought, yet the textual tokens are not directly aligned with visual space (Shao et al., 2024; Chen et al., 2026c). *Thinking with images* makes visual search explicit through tools such as zoom-in or cropping, but incurs additional test-time computation through multiple interaction rounds and image re-encoding (Lei et al., 2025; Zhang et al., 2026b; Tang et al., 2026a). *Latent visual reasoning* instead propagates continuous visual representations internally, but existing formulations leave the multi-step spatial search process implicit (Li et al., 2026a; Qin et al., 2026; Wang et al., 2026a).

This raises a question: *Can multi-step visual search become an explicit spatial process within a model's internal computation?* Our key insight is to represent search as an evolving spatial focus over the screenshot's visual tokens. Each search state specifies where to search and carries the

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

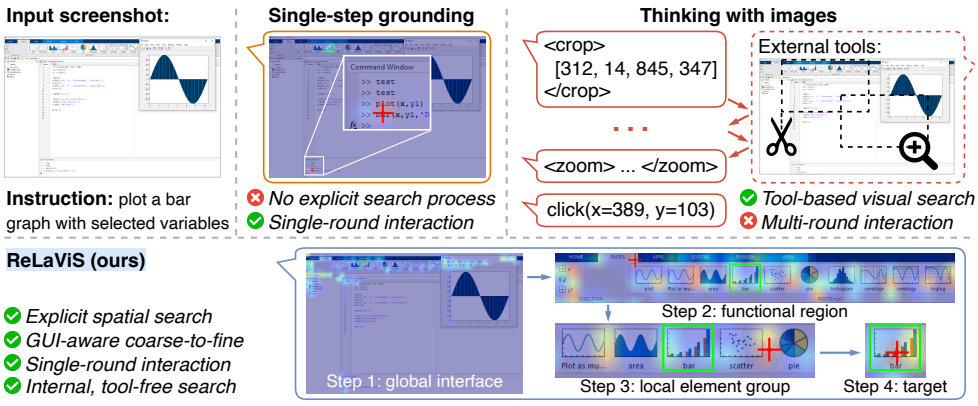


Figure 1: Conceptual comparison of grounding paradigms. ReLaViS performs GUI-aware coarse-to-fine visual search by updating explicit spatial search states within a single model interaction.

corresponding visual evidence forward to guide subsequent search. Search thus proceeds over the same encoded screenshot without additional interaction rounds or image re-encoding.

Beyond representing search states, we need to determine how their spatial focus should evolve toward the target. Human visual search in GUIs provides useful inspiration, as people often search through likely regions and element groups before identifying the target (Putkonen et al., 2025). This motivates using the GUI hierarchy as a natural inductive bias to organize multi-step visual search from coarse to fine. In practice, supervising this GUI-aware coarse-to-fine visual search remains challenging, as annotations of UI trees and search trajectories are scarce.

Building on these insights, we propose ***ReLaViS***, a framework that performs ***Re**current **La**tent **Vi**sual **S**earch* for GUI grounding (Fig. 1). Specifically, at each step, a spatial search head uses the hidden state to query the visual tokens, producing a spatial search distribution over them. This distribution represents the current search focus and then aggregates the visual tokens into latent visual evidence, which is recurrently fed as the next input embedding to condition subsequent search. We further supervise the search process with GUI-aware coarse-to-fine trajectories constructed from flat element annotations, progressing from the global interface through intermediate element groups to the target. After the final search step, a deterministic readout converts the distribution into a click coordinate.

Our contributions are threefold: (i) we formulate GUI grounding as recurrent latent visual search and propose ReLaViS, a framework that recurrently updates explicit spatial search states aligned with visual tokens within a single model interaction, without external tools; (ii) we construct stepwise supervisory trajectories from flat GUI element annotations to instantiate a GUI-aware coarse-to-fine inductive bias without requiring scarce ground-truth UI trees; and (iii) ReLaViS outperforms matched single-step baselines across all five GUI grounding benchmarks, including a 3.1-point accuracy gain on ScreenSpot-Pro (53.2% to 56.3%) with only 3.5% more inference FLOPs, while controlled analyses support its key design choices.

2 Related Work

GUI agents and GUI grounding. As GUI agents shift from structured metadata to direct screenshot perception, GUI grounding serves as the interface between instructions and actions (OpenAI, 2025; Anthropic, 2024). Existing methods differ in how they represent the target location. *Coordinate-based* methods generate textual coordinates or bounding boxes; prior work improved these predictions through supervised fine-tuning (SFT) (Wu et al., 2025c; Xie et al., 2025), reinforcement learning (RL) (Yuan et al., 2025; Tang et al., 2026b), and on-policy self-distillation (Zhang et al., 2026a; Huang et al., 2026). *Coordinate-free* methods represent the target location as a spatial distribution over visual tokens; prior work refined its construction and supervision (Wu et al., 2025b; Chen et al., 2026b; Zhou et al., 2026). Regardless of the output representation, high-resolution screens, small targets, and dense layouts pose challenges for GUI grounding (Li et al., 2025b; Feizi et al., 2026), motivating multi-step visual search. To this end, ReLaViS builds on coordinate-free grounding (Chen et al., 2026b) to perform supervised multi-step visual search.

Thinking about images and thinking with images. Existing multi-step approaches differ in how they represent and update search states. *Thinking about images* externalizes this process as textual chains of thought that describe target semantics and compare candidates (Tan et al., 2025; Chen et al., 2026c), yet remains confined to the textual space. *Thinking with images* instead updates visual evidence through adaptive zoom and region search (Park et al., 2025; Wu et al., 2025a; Tang et al., 2026a; Yao et al., 2026), multi-view aggregation (Zhang et al., 2026b; Luo et al., 2025), visual tool use (Lei et al., 2025; Chen et al., 2026a; Li et al., 2026b), and iterative visual feedback (Zhao et al., 2026; Li et al., 2025a; Wang et al., 2026b). It relies on external tools to produce new image views, which are re-encoded and processed in additional model interactions. This makes spatial search explicit at the observation level but incurs extra test-time computation. Such limitations motivate ReLaViS, which performs spatial search internally within a single model interaction.

Latent visual reasoning. Latent reasoning offers a mechanism for such internal search by propagating continuous representations rather than externalizing intermediate steps as textual tokens (Hao et al., 2026; Cheng & Durme, 2024). Recent work extended this paradigm to visual reasoning by reconstructing visual tokens from regions of interest (Li et al., 2026a), incorporating perceptual priors from external models (Qin et al., 2026), and using auxiliary image supervision to learn visual thoughts (Wang et al., 2026a; Yang et al., 2026). Together, these works established latent representations as viable internal visual reasoning states. However, they do not explicitly track the search focus over the visual token grid or supervise the progression of their latent states as multi-step visual search. In contrast, ReLaViS specializes latent visual reasoning for visual search by recurrently updating explicit spatial search states aligned with the visual token grid, and supervises the multi-step search process with a GUI-aware coarse-to-fine trajectory for GUI grounding.

3 METHODOLOGY

To make multi-step visual search explicit within the model, ReLaViS recurrently updates spatial search states that track the search focus and carry visual evidence across steps. Given a GUI screenshot I and an instruction q, the model predicts a click coordinate $\hat{\mathbf{c}} = (x, y)$ through T search steps within a single model interaction:

$$(I, q) \to S_1 \to \cdots \to S_T \to \hat{\mathbf{c}}, \tag{1}$$

where S_t denotes the *spatial search state* at step t. In this section, we describe how ReLaViS represents and recurrently updates S_t (Sec. 3.1), constructs a GUI-aware coarse-to-fine search trajectory (Sec. 3.2), and uses the resulting trajectory to supervise $S_{1:T}$ (Secs. 3.3 and 3.4).

3.1 RECURRENT LATENT VISUAL SEARCH

Fig. 2 shows the ReLaViS inference pipeline. Specifically, we embed latent visual search (LVS) into the response sequence with three control symbols, `LVS-START`, `LVS-SLOT`, and `LVS-END`, delimiting a span $\{\ldots, \texttt{LVS-START}, \texttt{LVS-SLOT}^{\times T}, \texttt{LVS-END}, \ldots\}$. When the model emits `LVS-START`, it switches from ordinary token decoding to recurrent latent visual search, with each `LVS-SLOT` reserving one of T search steps. After the final slot, `LVS-END` is inserted and token decoding resumes. A deterministic readout converts the final state S_T to the click coordinate.

To support explicit spatial representation and recurrent evidence propagation, we instantiate S_t as

$$S_t = (s_t, z_t), \quad s_t \in [0, 1]^N, \quad \sum_{i=1}^{N} s_{t,i} = 1, \quad z_t \in \mathbb{R}^d, \tag{2}$$

where N denotes the number of visual tokens and d their representation dimension. The *spatial search distribution* s_t assigns a normalized probability to each visual token and represents *where* the model currently searches. The *latent visual evidence* z_t represents *what* the model observes at those locations and replaces the input embedding of the next token, the t-th `LVS-SLOT`. The injected z_t participates in subsequent autoregressive computation and recurrently conditions the construction of the next state S_{t+1}. We next describe how ReLaViS constructs s_t and z_t.

In order to explicitly represent and directly supervise the spatial search distribution on the visual token grid, without relying on textual coordinates, we introduce an attention-based *spatial search*

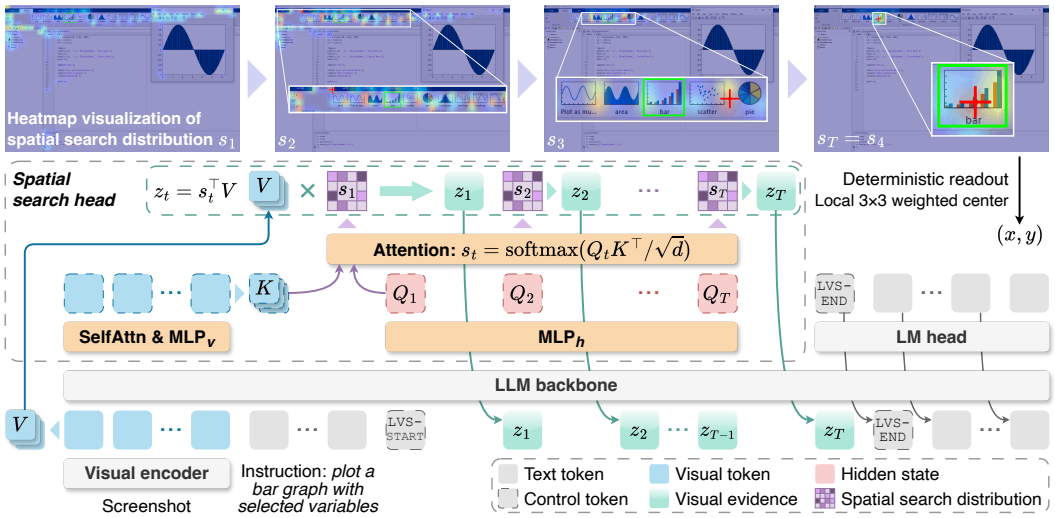


Figure 2: Inference pipeline of ReLaViS. At step t, the spatial search head attends to visual tokens V using query Q_t derived from the hidden state, producing the search distribution s_t. This distribution represents the search focus and aggregates V into latent visual evidence $z_t = s_t^\top V$, which serves as the next input embedding to condition subsequent search. After T steps, a deterministic readout maps s_T to the click coordinate. Within a single autoregressive sequence, the search focus evolves from the global interface through element groups to the target, as illustrated by the top heatmaps.

head. Let $V = [v_1, \ldots, v_N] \in \mathbb{R}^{N \times d}$ denote the visual tokens extracted from the screenshot by the visual encoder and multimodal merger. At search step t, let $h_t \in \mathbb{R}^d$ denote the last-layer hidden state at the position immediately before the t-th `LVS-SLOT`: `LVS-START` for $t = 1$ and the preceding `LVS-SLOT` with input z_{t-1} for $t > 1$. The spatial search head computes s_t as

$$\widetilde{V} = \mathrm{SelfAttn}(V) = \mathrm{SelfAttn}(v_1, \ldots, v_N),$$
$$Q_t = \mathrm{MLP}_h(h_t), \quad K = \mathrm{MLP}_v(\widetilde{V}), \quad s_t = \mathrm{softmax}\left(\frac{Q_t K^\top}{\sqrt{d}}\right). \tag{3}$$

Whereas prior coordinate-free methods used this attention distribution for single-step grounding, we predict s_t at every search step and use it to construct latent visual evidence for subsequent search.

Recycling hidden states as inputs can introduce a distributional mismatch with pretrained input embeddings (Deng et al., 2026). Motivated by this concern, we aggregate the original visual token embeddings according to s_t to carry evidence from the current search focus into subsequent search:

$$z_t = s_t^\top V = \sum_{i=1}^{N} s_{t,i} v_i. \tag{4}$$

Because s_t is normalized, z_t remains within the convex hull of the input-space visual embeddings while retaining an explicit spatial interpretation through s_t. Injecting z_t closes the recurrent loop by producing the contextual hidden state at the next search step: $h_{t+1} = \mathrm{VLM}(I, q, z_{\le t})$. Appendix A.1 provides additional details on recurrent latent visual search, including the task template and spatial search head architecture.

3.2 GUI-AWARE SEARCH TRAJECTORY CONSTRUCTION

Hierarchy-rich GUI corpora primarily cover mobile or web platforms (Li et al., 2024; Gurbuz et al., 2026), while desktop grounding data do not consistently pair UI trees with aligned instructions and target elements (Feizi et al., 2026). We therefore construct GUI-aware search trajectories from the flat GUI element annotations available in our training samples. Let E denote all GUI elements in the flat annotations, $b(e) = [x_{0,e}, y_{0,e}, x_{1,e}, y_{1,e}]$ the bounding box of element $e \in E$, and $e^\star$ the target. As shown in Fig. 5, we represent the trajectory as a nested sequence of candidate element sets:

$$E = C_1 \supseteq C_2 \supseteq \cdots \supseteq C_T = \{e^\star\}. \tag{5}$$

To construct this trajectory, we use the prior that GUI element groups tend to have compact enclosing rectangles. For a candidate set C, let $B(C)$ be its smallest enclosing rectangle and $\Pi(C)$ be the set of bipartitions induced by cuts between adjacent elements sorted by horizontal or vertical center coordinates. We select the partition that minimizes the total area of the two enclosing rectangles:

$$(C_a^\star, C_b^\star) = \underset{(C_a, C_b)\in\Pi(C)}{\arg\min} \left[\mathrm{Area}(B(C_a)) + \mathrm{Area}(B(C_b))\right]. \tag{6}$$

We then recurse into the partition branch containing the target, yielding a raw GUI-aware partition path from the full element set E through spatially compact element groups to $e^\star$.

In order to remove adjacent redundant steps from the raw partition path, we retain key steps based on cumulative reduction in candidate count from the last retained set and local reduction at the next partition, forming a T-step trajectory. The trajectory provides a spatial approximation to the GUI hierarchy, progressing from the global interface through local element groups to the target, as illustrated by trajectory visualizations (Appendix B.1) and supported by close agreement with ground-truth UI-tree hierarchies (Appendix C.4). We then map every element in C_t to its support on the visual token grid. Details on the trajectory construction are provided in Appendix A.2.

3.3 Search process supervision

To guide the spatial search states $S_{1:T}$ along the constructed trajectory $C_{1:T}$, we introduce two complementary supervision objectives: the *candidate-wise coverage (CWC) loss* encourages each s_t to maintain coverage over each candidate element in C_t; the *Fitts-Gaussian peak (FGP) loss* shapes the final distribution s_T within the target element for coordinate readout.

Candidate-wise coverage loss. Let $\Omega(e)$ be the visual token support of e: tokens whose patches intersect with $b(e)$. We aggregate its probability mass as $m_{t,e} = \sum_{v_i\in\Omega(e)} s_{t,i}$ and define

$$\mathcal{L}_{\mathrm{CWC}}^{(t)} = \sum_{e\in C_t} \max\left(\frac{1}{|C_t|} - m_{t,e},\, 0\right). \tag{7}$$

By applying a lower bound to each candidate's aggregate probability mass, CWC encourages coverage of each candidate and discourages collapse onto a few. At the same time, this candidate-wise supervision preserves flexibility to emphasize grounding-relevant visual tokens within each candidate without prescribing a token-level target distribution. The comparison of supervision objectives in Appendix C.3 shows that CWC supervision outperforms token-level distributional supervision.

Fitts-Gaussian peak loss. To capture the center-biased nature of GUI clicking, we construct a normalized target distribution d_t on the visual token grid using 2D Gaussians centered on candidates in C_t and truncated at their boundaries. Their horizontal and vertical standard deviations match candidate widths and heights. At the final step, $C_T = \{e^\star\}$, and we apply d_T through the FGP loss:

$$\mathcal{L}_{\mathrm{FGP}}^{(T)} = D_{\mathrm{KL}}(d_T \,\|\, s_T) = \sum_{i=1}^{N} d_{T,i} \log \frac{d_{T,i}}{s_{T,i}}. \tag{8}$$

While the final-step $\mathcal{L}_{\mathrm{CWC}}^{(T)}$ places probability mass inside the target element, $\mathcal{L}_{\mathrm{FGP}}^{(T)}$ shapes the distribution to peak near the target center, facilitating coordinate readout for precise click localization.

The overall training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{NTP}} + \lambda_{\mathrm{FGP}}\mathcal{L}_{\mathrm{FGP}}^{(T)} + \lambda_{\mathrm{CWC}}^{(T)}\mathcal{L}_{\mathrm{CWC}}^{(T)} + \lambda_{\mathrm{CWC}}^{(1:T-1)}\,\mathrm{mean}(\mathcal{L}_{\mathrm{CWC}}^{(1:T-1)}). \tag{9}$$

Here, $\mathcal{L}_{\mathrm{NTP}}$ is the next-token prediction loss, and the three coefficients independently weight final peak shaping, final target coverage, and candidate-wise supervision at intermediate search steps. Appendix A.3 gives more details of the loss weighting and the construction of d_t.

3.4 Training and inference

The bottom panel of Fig. 5 summarizes our three-stage training protocol: Stage 1 (*spatial search head warm-up*) warms up the spatial search head for recurrent training through single-step localization ($T = 1$), with the VLM frozen. Stage 2 (*teacher-forced recurrent training*) jointly trains the

spatial search head and the VLM using teacher-forced latent visual evidence $z_t^{\mathrm{gt}} = d_t^\top V$ as input embeddings, allowing the model to learn from the ground-truth visual evidence without propagating unreliable early predictions. Stage 3 (*self-conditioned recurrent adaptation*) continues joint training with input embeddings $z_t = s_t^\top V$ constructed from predicted s_t through a self-conditioned rollout. This stage exposes the model to accumulated search errors and reduces the training–inference mismatch. No-gradient probes predict s_t for input construction, followed by one differentiable forward pass that recomputes $s_{1:T}$ and evaluates the objective in Eq. (9), avoiding retention of the recurrent computation graph through the LLM backbone and its prohibitive memory cost.

At inference, the model performs T search steps using only screenshot I and instruction q to produce s_T. To reduce patch-level quantization error, a deterministic readout converts s_T into $\hat{\mathbf{c}}$ by taking a probability-weighted average of token centers in the 3×3 neighborhood of the highest-probability token. No element annotations, UI tree, zoom-in or cropping tool, or additional interaction rounds are required. Appendix A.4 details the training procedures, cost analysis, and coordinate readout.

4 Experiments

Training data and configuration. Following the training-set size used in the controlled comparison in the GroundCUA study (Feizi et al., 2026), we randomly select 100K training samples and a disjoint 5K validation set from GroundCUA. Each sample contains a screenshot, an instruction, a target element, and flat GUI element annotations with non-nested bounding boxes for trajectory construction. We initialize ReLaViS-7B from Qwen2.5-VL-7B (Bai et al., 2025b) and train for one epoch per stage, with default settings $T = 4$, $\lambda_{\mathrm{FGP}} = 1$, and $\lambda_{\mathrm{CWC}}^{(T)} = \lambda_{\mathrm{CWC}}^{(1:T-1)} = 5$.

Benchmarks and metrics. Evaluation covers ScreenSpot-Pro (Li et al., 2025b), MMBench-GUI-L2 (Wang et al., 2026c), OSWorld-G-Refine (Xie et al., 2025), ScreenSpot-v2 (Wu et al., 2025c), and UI-Vision (Nayak et al., 2025) (in-domain), which together span high-resolution professional interfaces, standard GUIs, and cross-platform settings. We use point grounding accuracy (Lin et al., 2025): a prediction is correct when its click coordinate lies inside the ground-truth bounding box.

Baselines. Public baselines include zero-shot general VLMs and coordinate-based or coordinate-free GUI-specialized models evaluated without external tools. The latter are all based on Qwen2.5-VL-7B, a widely adopted backbone for training-based GUI grounding methods (Xie et al., 2025; Tang et al., 2026b; Wu et al., 2025b), reducing confounding from pretraining and scale differences. For controlled comparisons, we train two single-step-7B baselines using the same backbone and data as ReLaViS-7B, with matched training settings: the coordinate-based model generates textual click coordinates, while the coordinate-free $T = 1$ variant of ReLaViS predicts a spatial distribution over visual tokens. We also compare with thinking with images methods that use external tools to obtain new image views across multiple interaction rounds. Implementation details are in Appendix B.

4.1 Main results

As shown in Table 1, ReLaViS outperforms both matched single-step baselines on all five benchmarks. ReLaViS raises average accuracy to 68.1% from 61.6% for the coordinate-based baseline and 65.3% for the stricter coordinate-free baseline, gains of 6.5 and 2.8 percentage points (pp), respectively. Among the compared methods without external tools, ReLaViS achieves the highest five-benchmark average (68.1%) and average over the four out-of-domain benchmarks excluding UI-Vision (72.7%, versus 71.5% for the next-best InfiGUI-G1-7B). These results highlight its strong overall performance across benchmarks and platforms. Notably, on ScreenSpot-Pro, ReLaViS leads methods without external tools with 56.3% accuracy, outperforming the coordinate-free baseline by 3.1 pp with only 3.5% more inference FLOPs, demonstrating its advantage in professional and complex desktop interfaces at comparable inference cost (Table 5). On ScreenSpot-v2, however, the gain is only 0.3 pp, indicating that the improvement varies across benchmarks and motivating an analysis of its association with target size. Complete benchmark results are in Appendix C.6.

Where does ReLaViS help? Fig. 3 (left) pools instances from all five benchmarks and bins them by relative target bounding box area. ReLaViS performs comparably to both controlled single-step baselines in the three largest-target bins, while generally achieving larger gains in the smaller-target bins. Benchmark composition exhibits a corresponding pattern: ScreenSpot-v2 contributes mainly to the larger-target bins, whereas UI-Vision and ScreenSpot-Pro contribute predominantly to the

Table 1: Performance comparison across five GUI grounding benchmarks. For models trained in this work, results are reported as mean$_{\pm std}$ over three random seeds. "–" denotes unavailable results. Results with † are reproduced using official implementations; results for other baselines from prior work are taken from the corresponding papers. **Bold** values are best within and outside "thinking with images" separately; underlined values are second-best only outside it.

Model	Training data size	ScreenSpot-Pro	MMBench-GUI-L2	OSWorld-G-Refine	ScreenSpot-v2	UI-Vision	Average
Zero-shot general VLMs							
GPT-4o (OpenAI, 2024)	–	0.8	2.9	–	80.7	1.4	–
Operator (OpenAI, 2025)	–	–	–	40.6	70.5	–	–
Qwen2.5-VL-3B† (Bai et al., 2025b)	–	18.4	55.2	38.8	69.6	7.4	37.9
Qwen2.5-VL-7B† (Bai et al., 2025b)	–	28.8	71.9	44.0	88.4	14.7	49.5
GUI-specialized models (coordinate-based)							
SE-GUI-7B (Yuan et al., 2025)	3K	47.3	34.5	33.9	90.3	16.7	44.5
Jedi-7B (Xie et al., 2025)	4M	39.5	70.4	54.1	91.7	24.8	56.1
UI-TARS-1.5-7B (Qin et al., 2025)	–	49.6	64.3	**64.2**	89.7	20.8	57.7
GUI-G^2-7B (Tang et al., 2026b)	100K	47.5	79.5	61.9	93.3	25.6	61.6
InfiGUI-G1-7B (Liu et al., 2026b)	44K	51.9	80.8	59.9	**93.5**	26.1	62.4
GUI-specialized models (coordinate-free)							
GUI-Actor-7B (Wu et al., 2025b)	9.6M	44.6	70.9	47.0	92.1	21.9	55.3
V2P-7B (Chen et al., 2026b)	9.6M	52.5	79.9	52.5	92.4	28.8	61.2
Controlled comparison (identical training data)							
Single-step-7B (coordinate-based)	100K	46.3$_{\pm 1.40}$	78.7$_{\pm 0.30}$	59.6$_{\pm 0.20}$	90.0$_{\pm 0.75}$	33.4$_{\pm 0.35}$	61.6$_{\pm 0.38}$
Single-step-7B (coordinate-free)	100K	53.2$_{\pm 0.52}$	80.2$_{\pm 0.55}$	59.3$_{\pm 1.18}$	90.4$_{\pm 0.48}$	43.1$_{\pm 0.42}$	65.3$_{\pm 0.31}$
ReLaViS-7B	100K	**56.3**$_{\pm 0.29}$	**81.1**$_{\pm 0.08}$	62.5$_{\pm 1.01}$	90.7$_{\pm 0.21}$	**49.7**$_{\pm 0.35}$	**68.1**$_{\pm 0.25}$
Thinking with images (external tools, multi-round)							
MVP† (Zhang et al., 2026b) + Qwen2.5-VL-7B	–	46.7	55.0	51.2	73.3	19.6	49.2
UI-Zoomer† (Tang et al., 2026a) + Qwen2.5-VL-7B	–	39.9	74.2	54.9	90.7	23.9	56.7
GUI-Spotlight† (Lei et al., 2025)	18.5K	53.3	72.2	56.7	88.8	23.5	58.9
GUI-Cursor (Zhao et al., 2026)	8K	56.5	–	58.0	**93.9**	25.7	–
Zoom-in + Qwen2.5-VL-7B	–	49.8	77.3	59.2	88.1	23.9	59.6
Zoom-in + single-step-7B (coordinate-free)	100K	62.3$_{\pm 0.33}$	80.5$_{\pm 0.11}$	**68.3**$_{\pm 1.07}$	84.7$_{\pm 0.41}$	47.3$_{\pm 0.44}$	68.6$_{\pm 0.21}$
Zoom-in + ReLaViS-7B	100K	**63.5**$_{\pm 0.57}$	**80.9**$_{\pm 0.16}$	68.2$_{\pm 0.54}$	84.9$_{\pm 0.59}$	**52.3**$_{\pm 0.36}$	**70.0**$_{\pm 0.19}$

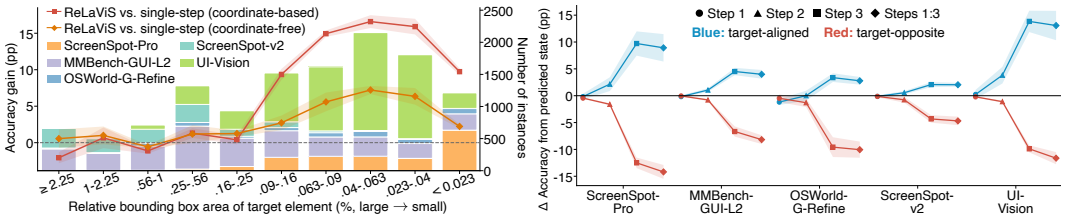


Figure 3: *Left*: pooled accuracy gains of ReLaViS-7B over single-step baselines across target area bins; stacked bars show instance counts and benchmark composition. *Right*: accuracy changes relative to predicted-state inference when target-aligned or target-opposite spatial search states are injected at $t = 1, 2, 3$ individually or jointly. Curves show multi-seed means with std shading.

smaller-target bins. This association is consistent with the smaller gain on ScreenSpot-v2 and larger gains on UI-Vision and ScreenSpot-Pro. Within-benchmark results in Appendix C.1 show a similar tendency, further supporting this association and suggesting greater gains on small targets.

How does ReLaViS compare to thinking with images? Across all five benchmarks, ReLaViS achieves an average accuracy of 68.1%, exceeding four thinking with images baselines: MVP (49.2%), UI-Zoomer (56.7%), GUI-Spotlight (58.9%), and zero-shot zoom-in (59.6%). ReLaViS requires one interaction round, compared with averages of 4.8, 2, 2.3, and 2 rounds for these baselines. On ScreenSpot-Pro, ReLaViS achieves 56.3% accuracy, outperforming these four baselines, while its latency is 2.19 s/sample, compared with 11.97, 2.34, 9.69, and 4.67 s/sample under the respective implementations, highlighting its favorable accuracy–efficiency trade-off. Furthermore, adding zoom-in to ReLaViS raises its five-benchmark average to 70.0%, exceeding zoom-in + single-step (coordinate-free) by 1.4 pp and leading the thinking with images group on three of five benchmarks. These gains support the complementarity of acquiring new image views externally and organizing visual evidence internally. However, on MMBench-GUI-L2 and ScreenSpot-v2, ReLaViS alone

Table 2: Ablation results on ScreenSpot-Pro. *Left*: search trajectory and latent visual evidence. *Right*: search steps and training stages. Empty cells inherit ReLaViS settings; ✓✓ denotes two epochs. Δ denotes the Avg. change from full ReLaViS (pp).

Group	Steps T	Search trajectory	Visual evidence z_t	Avg.	Δ
ReLaViS	4	GUI-aware	$s_t^\top V$	**56.3**$_{\pm.29}$	–
RQ1	1	Only target at T		53.2$_{\pm.52}$	-3.1
		Only target at T		54.0$_{\pm.48}$	-2.3
		Repeated target		53.9$_{\pm.96}$	-2.4
RQ2			LVS-SLOT	54.5$_{\pm.71}$	-1.8
			h_t	55.2$_{\pm.75}$	-1.1
RQ4		Geometric zoom		53.8$_{\pm.39}$	-2.5
		Shuffled GUI-aware		53.5$_{\pm.19}$	-2.8

Group	Steps T	Training stage 1	2	3	Avg.	Δ
ReLaViS	4	✓	✓	✓	**56.3**$_{\pm0.29}$	–
RQ5	3				54.8$_{\pm1.05}$	-1.5
	5				55.1$_{\pm0.93}$	-1.2
RQ6		✓	✓	×	53.0$_{\pm0.44}$	-3.3
		✓	×	✓	54.7$_{\pm1.22}$	-1.6
		×	✓	✓	54.9$_{\pm0.03}$	-1.4
		✓	✓✓	×	53.6$_{\pm1.20}$	-2.7
		✓	×	✓✓	55.2$_{\pm0.53}$	-1.1

achieves comparable or higher accuracy than its zoom-in variant while completing grounding in a single interaction round without external tools or additional image re-encoding.

4.2 Ablations and analyses

To evaluate the effectiveness of the key design choices of ReLaViS, we conduct ablations, interventions, and analyses organized around six research questions. Table 2 summarizes the ablation results. Appendix C provides further support through ablations and analyses, including studies of coordinate readout, trajectory construction, model scale, and search supervision.

RQ1: Does coarse-to-fine multi-step search improve grounding? To separate the effects of recurrent multi-step computation and supervision, we compare the coordinate-free single-step baseline, two 4-step variants supervised on the target element only at the final step T or at every step, and full ReLaViS with GUI-aware coarse-to-fine supervision. As shown in Table 2, increasing T from 1 to 4 with final-step supervision raises accuracy from 53.2% to only 54.0%, while repeating target-element supervision at every step yields 53.9%. ReLaViS reaches 56.3%, exceeding these 4-step variants by 2.3 and 2.4 pp, respectively. GUI-aware coarse-to-fine multi-step visual search thus provides gains beyond additional computation or supervision at every step alone.

RQ2: How should visual evidence be propagated? We next examine the contribution of recurrent latent visual evidence propagation and compare $z_t = s_t^\top V$ with direct feedback of the contextual hidden state. Propagating predicted latent visual evidence raises accuracy from 54.5% with a fixed LVS-SLOT embedding to 56.3%, indicating a benefit from propagating spatially selected visual evidence under the same trajectory supervision. Directly feeding back h_t reaches 55.2%, outperforming the fixed token but remaining 1.1 pp below $s_t^\top V$. These results support $s_t^\top V$ as an effective representation for recurrent propagation of latent visual evidence.

RQ3: Do subsequent steps use preceding spatial search states? To test whether subsequent grounding depends on preceding spatial search states, we intervene on predicted s_t when computing the recurrent feedback $z_t = s_t^\top V$. We replace s_t at one or all preceding recurrent steps ($t = 1, 2, 3$) with either a Gaussian distribution aligned with the target element or a shape-matched Gaussian shifted to the opposite grid location. The model still predicts the final distribution $s_T = s_4$, from which the click coordinate is obtained. Across the five benchmarks, joint target-aligned injection at steps 1:3 raises accuracy by 2.0–13.1 pp, whereas target-opposite injection lowers it by 4.7–14.2 pp (Fig. 3, right). Individual-step effects are small at the first step, modest at the second, and largest at the third. These effects show that subsequent grounding functionally depends on the feedback derived from preceding spatial search states, with the strongest sensitivity at later recurrent steps.

RQ4: Does the GUI-aware spatial prior improve grounding? With four steps in both settings, we construct the GUI-aware trajectory by recursively partitioning the candidate set into spatially compact GUI element groups, whereas geometric zoom linearly contracts a box from the full screenshot to the target bounding box. Our GUI-aware trajectory reaches 56.3%, exceeding geometric zoom at 53.8% by 2.5 pp and showing that its spatial prior is more effective than generic contraction. In addition, shuffling the first three GUI-aware steps while fixing the final target reduces accuracy to 53.5%, 2.8 pp below the full model, highlighting the importance of coarse-to-fine order.

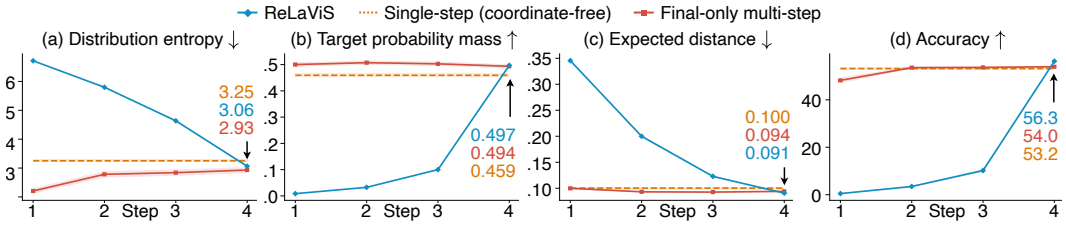


Figure 4: Stepwise search dynamics on ScreenSpot-Pro. Metric definitions are in Appendix B.5. Dashed lines indicate single-step references; curves show multi-seed means with std shading.

RQ5: How does the number of search steps affect grounding? Table 2 (right) shows that 4-step training and inference achieve 56.3%, compared with 54.8% for $T = 3$ and 55.1% for $T = 5$. These results highlight the importance of an appropriate number of search steps, as grounding accuracy does not increase monotonically with recurrent computation under the current setup.

RQ6: Does three-stage training matter? As shown in Table 2, removing Stage 3 reduces accuracy from 56.3% to 53.0%; replacing it with an additional Stage 2 epoch reaches only 53.6%. Thus, an extra teacher-forced epoch does not recover the benefit of adaptation to self-conditioned rollouts. Removing Stage 2 yields 54.7% with one Stage 3 epoch and 55.2% even after extending Stage 3 to two epochs, supporting the value of initial recurrent training with reliable ground-truth evidence. Omitting Stage 1 gives 54.9%, confirming that spatial search head warm-up provides a useful initialization. The three stages thus provide complementary benefits through initialization, stable recurrent learning, and training–inference adaptation, beyond additional training epochs alone.

4.3 Recurrent Search Dynamics

Fig. 4 compares the stepwise search dynamics of ReLaViS and two baselines from Table 2 (group RQ1, rows 1–2): the coordinate-free single-step baseline and the final-only multi-step variant, which receives supervision only at the final step. Across four search steps, the entropy of the ReLaViS spatial search distribution decreases to 3.06, target probability mass rises from near zero to 0.497, and expected distance to the target element decreases from 0.345 to 0.091. In contrast, the final-only variant starts with a highly concentrated search distribution, and its target probability mass and expected distance vary little across steps. These coordinated trends show that ReLaViS progressively concentrates its spatial search toward the target before the final readout.

The predicted heatmaps in Figs. 1 and 2 illustrate how ReLaViS progressively contracts its search focus from the global interface through local element groups (such as toolbars) to the target element; in contrast, the single-step variant incorrectly targets a non-interactive text element. Visualizations and qualitative analyses in Appendix C.5 illustrate how ReLaViS supports candidate disambiguation, semantic grounding, and fine-grained localization, and examine failure cases. Stepwise diagnostics in Appendix C.4 further show that ReLaViS follows the GUI-aware coarse-to-fine trajectory more closely than the single-step and final-only variants. With this GUI-aware search, ReLaViS reaches a better final spatial search state than these variants, with higher target probability mass, lower expected distance, and higher grounding accuracy (Fig. 4).

5 Conclusion and Limitations

We introduce ReLaViS, a novel framework that formulates GUI grounding as explicit multi-step visual search within a single model interaction. Guided by GUI-aware trajectories constructed from flat element annotations, its recurrent spatial search states carry visual evidence forward and progressively narrow the search toward the target without external tools. Comprehensive experiments show that ReLaViS outperforms matched single-step baselines on all five benchmarks, with its gains on ScreenSpot-Pro highlighting the value of recurrent latent visual search for complex GUI grounding at modest inference overhead. A current limitation is that ReLaViS mainly focuses on GUI grounding for click actions, leaving its extension to other actions, such as drag-and-drop gestures, for future work. More broadly, extending ReLaViS to general visual search tasks remains promising.

AI USE STATEMENT

In this work, we use generative AI tools to assist with implementing methods. We do not use generative AI tools to formulate or refine hypotheses, design or provide feedback on research methodology or experiments, or interpret results, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we use generative AI tools to assist with literature retrieval and discovery and to edit the manuscript to improve its readability and language. We review all AI-assisted work. In particular, we review and test AI-assisted code for correctness and verify the relevance of references identified with AI assistance. We also review all AI-assisted manuscript edits to preserve technical meaning and accurately reflect our work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We support reproducibility by documenting ReLaViS, the experimental protocol, and the supporting empirical evidence. Secs. 3 and 4 present the core method and experimental setup. Further information appears in Appendices A, B, and C, which respectively provide the detailed task and method specification, data preparation and experimental configurations, and complete benchmark results and additional analyses.

REFERENCES

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

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin 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, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

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 Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026a. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i35.40175. URL https://doi.org/10.1609/aaai.v40i35.40175.

Jikai Chen, Long Chen, Dong Wang, Qinglin Su, Zhixuan Chu, Bingguang Hao, Leilei Gan, Chenyi Zhuang, and Jinjie Gu. V2p: Visual attention calibration for gui grounding via background suppression and center peaking, 2026b. URL https://arxiv.org/abs/2508.13634.

Liangyu Chen, Hanzhang Zhou, chenglin cai, Jianan Zhang, Panrong Tong, Xu Zhang, Quyu Kong, Chen Liu, Yuqi Liu, Wenxuan Wang, Yue Wang, Qin Jin, and Steven HOI. Ui-ins: Enhancing gui grounding with multi-perspective instruction as reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 54418–54444, 2026c. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5965f3a748a8d41415db2bfa44635cc3-Paper-Conference.pdf.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations, 2024. URL https://arxiv.org/abs/2412.13171.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Llm latent reasoning as chain of superposition, 2026. URL https://arxiv.org/abs/2510.15522.

Aarash Feizi, Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Kaixin Li, Rabiul Awal, Xing Han Lu, Johan S Obando Ceron, Juan A. Rodriguez, Nicolas Chapados, David Vazquez, Adriana Romero-Soriano, Reihaneh Rabbany, Perouz Taslakian, Christopher Pal, Spandana Gella, and Sai Rajeswar Mudumba. Grounding computer use agents on human demonstrations. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 5181–5215, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0961fffb7adb81388b9b24ab6a3ff7ab-Paper-Conference.pdf.

A. Said Gurbuz, Sunghwan Hong, Ahmed Nassar, Marc Pollefeys, and Peter Staar. Screenparse: Moving beyond sparse grounding with complete screen parsing supervision, 2026. URL https://arxiv.org/abs/2602.14276.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2026. URL https://arxiv.org/abs/2412.06769.

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, 2021. URL https://arxiv.org/abs/2106.09685.

Siyuan Hu, Mingyu Ouyang, Difei Gao, and Mike Zheng Shou. The dawn of gui agent: A preliminary case study with claude 3.5 computer use, 2024. URL https://arxiv.org/abs/2411.10323.

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, 2026. URL https://arxiv.org/abs/2606.18101.

Bin Lei, Nuo Xu, Ali Payani, Mingyi Hong, Chunhua Liao, Yu Cao, and Caiwen Ding. GUI-SPOTLIGHT: Adaptive iterative focus refinement for enhanced gui visual grounding, 2025. URL https://arxiv.org/abs/2510.04039.

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

Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 148076–148090, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/efa0939caa8a906fd0a61a9c60e46e8b-Paper-Conference.pdf.

Junlong Li, Chao Hao, Lap-Pui Chau, and Yi Wang. Gui-c^2: Coarse-to-fine gui grounding via difficulty-aware reinforcement learning, 2026b. URL https://arxiv.org/abs/2605.30884.

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*, MM '25, pp. 8778–8786, New York, NY, USA, 2025b. Association for Computing Machinery. ISBN 9798400720352. doi: 10.1145/3746027.3755688. URL https://doi.org/10.1145/3746027.3755688.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 92130–92154. Curran Associates, Inc., 2024. doi: 10.52202/079017-2925. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/a79f3ef3b445fd4659f44648f7ea8ffd-Paper-Datasets_and_Benchmarks_Track.pdf.

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

Yichao Liu, Huawen Shen, Liu Yu, Shiyu Liu, Zeyu Chen, and Yu Zhou. Drs-gui: Dynamic region search for training-free gui grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 34606–34616, June 2026a.

Yuhang Liu, Zeyu Liu, Shuanghe Zhu, Pengxiang Li, Congkai Xie, Jiasheng Wang, Xueyu Hu, Xiaotian Han, Jianbo Yuan, Xinyao Wang, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-g1: advancing gui grounding with adaptive exploration policy optimization. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026b. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i38.40500. URL https://doi.org/10.1609/aaai.v40i38.40500.

Tiange Luo, Lajanugen Logeswaran, Justin Johnson, and Honglak Lee. Visual test-time scaling for gui agent grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19989–19998, October 2025.

Viktor Muryn, Marta Sumyk, Mariya Hirna, Sofiya Garkot, and Maksym Shamrai. Screen2ax: Vision-based approach for automatic macos accessibility generation, 2025. URL https://arxiv.org/abs/2507.16704.

Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, 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 Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45817–45851. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/nayak25a.html.

OpenAI. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

OpenAI. Computer-using agent: Introducing a universal interface for ai to interact with the digital world. 2025. URL https://openai.com/index/computer-using-agent.

Joonhyung Park, Peng Tang, Sagnik Das, Srikar Appalaraju, Kunwar Yashraj Singh, R. Manmatha, and Shabnam Ghadar. R-VLM: Region-aware vision language model for precise GUI grounding. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9669–9685, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.501. URL https://aclanthology.org/2025.findings-acl.501/.

Aini Putkonen, Yue Jiang, Jingchun Zeng, Olli Tammilehto, Jussi P.P. Jokinen, and Antti Oulasvirta. Understanding visual search in graphical user interfaces. *International Journal of Human-Computer Studies*, 199:103483, 2025. ISSN 1071-5819. doi: https://doi.org/10.1016/j.ijhcs.2025.103483. URL https://www.sciencedirect.com/science/article/pii/S1071581925000400.

Yiming Qin, Bomin Wei, Jiaxin Ge, Konstantinos Kallidromitis, Stephanie Fu, Trevor Darrell, and XuDong Wang. Chain-of-visual-thought: Teaching vlms to see and think better with continuous visual tokens. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 445–462, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37359-5.

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

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 8612–8642. Curran Associates, Inc., 2024. doi: 10.52202/079017-0275. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0ff38d72a2e0aa6dbe42de83a17b2223-Paper-Datasets_and_Benchmarks_Track.pdf.

Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Xiansheng Chen, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-rft: Reinforcement fine-tuning for visual reasoning of vision language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 5772–5822. Curran Associates, Inc., 2025. doi: 10.52202/085713-0204. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/08d70284b013c03ba89cd2b642bc864b-Paper-Conference.pdf.

Fei Tang, Bofan Chen, Zhengxi Lu, Tongbo Chen, Songqin Nong, Tao Jiang, Wenhao Xu, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Ui-zoomer: Uncertainty-driven adaptive zoom-in for gui grounding, 2026a. URL https://arxiv.org/abs/2604.14113.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. Gui-g2: Gaussian reward modeling for gui grounding. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026b. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i39.40606. URL https://doi.org/10.1609/aaai.v40i39.40606.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception, 2024. URL https://arxiv.org/abs/2401.16158.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond image and language. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12030–12040, June 2026a.

Wenkai Wang, Xiyun Li, Hongcan Guo, Wenhao Yu, Tianqing Fang, Haitao Mi, Dong Yu, and Shengyu Zhang. Measure twice, click once: Co-evolving proposer and visual critic via reinforcement learning for GUI grounding. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20964–20984, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.960. URL https://aclanthology.org/2026.acl-long.960/.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, Weiyun Wang, Xiangyu Zhao, Jixuan Chen, Haodong Duan, Tianbao Xie, Chenyu Yang, Shiqian Su, Yue Yu, Yanting Zhang, Xiangyu Yue, Weijie Su, Xizhou

Zhu, Wei Shen, Jifeng Dai, and Wenhai Wang. Mmbench-gui: A unified hierarchical evaluation framework for multi-platform gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6239–6248, June 2026c.

Hang Wu, Hongkai Chen, Yujun Cai, Chang Liu, Qingwen Ye, Ming-Hsuan Yang, and Yiwei Wang. DiMo-GUI: Advancing test-time scaling in GUI grounding via modality-aware visual reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26246–26256, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1334. URL https://aclanthology.org/2025.emnlp-main.1334/.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, Si Qin, Lars Liden, Qingwei Lin, Huan Zhang, Tong Zhang, Jianbing Zhang, Dongmei Zhang, and Jianfeng Gao. Gui-actor: Coordinate-free visual grounding for gui agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 15101–15128. Curran Associates, Inc., 2025b. doi: 10.52202/085713-0508. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/16130af940e9dabb43c726119bd3b42e-Paper-Conference.pdf.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. Os-atlas: Foundation action model for generalist gui agents. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 5090–5108, 2025c. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/0faa4bc5f522076947a030273629d4fe-Paper-Conference.pdf.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-0817. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/22c868099177ee278eb7baccec649f35-Paper-Datasets_and_Benchmarks_Track.pdf.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-UI: Visual grounding for GUI instructions. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22418–22433, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1152. URL https://aclanthology.org/2025.findings-acl.1152/.

Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 33510–33520, June 2026.

Ruilin Yao, Shegnwu Xiong, Tianyu Zou, Shili Xiong, and Yi Rong. Autofocus: Uncertainty-aware active visual search for gui grounding, 2026. URL https://arxiv.org/abs/2605.02630.

Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, and Bo Li. Se-gui: Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 127658–127679. Curran Associates, Inc., 2025. doi: 10.52202/085713-4251. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b95c7e24501f5d1dddbc5e8526cda7ae-Paper-Conference.pdf.

Yan Zhang, Daiqing Wu, Huawen Shen, Can Ma, and Yu Zhou. Learn where to click from yourself: On-policy self-distillation for gui grounding, 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 (CVPR)*, pp. 27482–27492, June 2026b.

Yu Zhao, Wei-Ning Chen, Huseyin Atahan Inan, Samuel Kessler, Lu Wang, Lukas Wutschitz, Fangkai Yang, Chaoyun Zhang, Pasquale Minervini, Saravan Rajmohan, and Robert Sim. Learning gui grounding with spatial reasoning from visual feedback, 2026. URL https://arxiv.org/abs/2509.21552.

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 Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 79–97, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37016-7.

APPENDIX CONTENTS

A Additional details of ReLaViS **16**

A.1 Recurrent latent visual search 16

A.2 GUI-aware search trajectory construction 17

A.3 Search process supervision 19

A.4 Training and inference 19

B Implementation and reproducibility details **20**

B.1 Dataset and constructed trajectories 20

B.2 Training configuration 21

B.3 Comparison and ablation configuration 22

B.4 Training and inference cost 22

B.5 Evaluation benchmarks and metrics 23

C Additional results and analyses **24**

C.1 Target-size analysis within each benchmark 24

C.2 Results at a smaller model scale 24

C.3 Additional ablations 24

C.4 Search trajectory and state analyses 25

C.5 Visualizations and qualitative analyses 26

C.6 Complete benchmark results 31

A ADDITIONAL DETAILS OF RELAVIS

A.1 RECURRENT LATENT VISUAL SEARCH

Task template. We follow the tool-call template used in the official Qwen2.5-VL (Bai et al., 2025b) implementation for GUI interaction. Our system prompt exposes a `computer_use` tool whose `left_click` action identifies the click target through an LVS span in place of textual coordinates:

```
You are a helpful assistant. The user will give you an instruction, and you MUST left click on
↪  the corresponding UI element via tool call. If you are not sure about where to click,
↪  guess a most likely one.

# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name": "computer_use", "description": "Use a mouse to
↪  interact with a computer.
* Make sure to click any buttons, links, icons, etc with the cursor tip in the center of the
↪  element.
* You can only use the left_click action to interact with the computer.", "parameters":
↪  {"properties": {"action": {"description": "The action to perform. The available actions
↪  are:
* `left_click`: Click the left mouse button on the UI element specified by lvr tokens.",
↪  "enum": ["left_click"], "type": "string"}, "lvr": {"description":
↪  "<|object_ref_start|><|lvr|><|object_ref_end|>: Latent visual reasoning tokens identifying
↪  the UI element to left click. Required only by `action=left_click`.", "type":
↪  "special_token"}, "required": ["action"], "type": "object"}}}}
</tools>

For each function call, return a json object with function name and arguments within
↪  <tool_call></tool_call> XML tags:
```

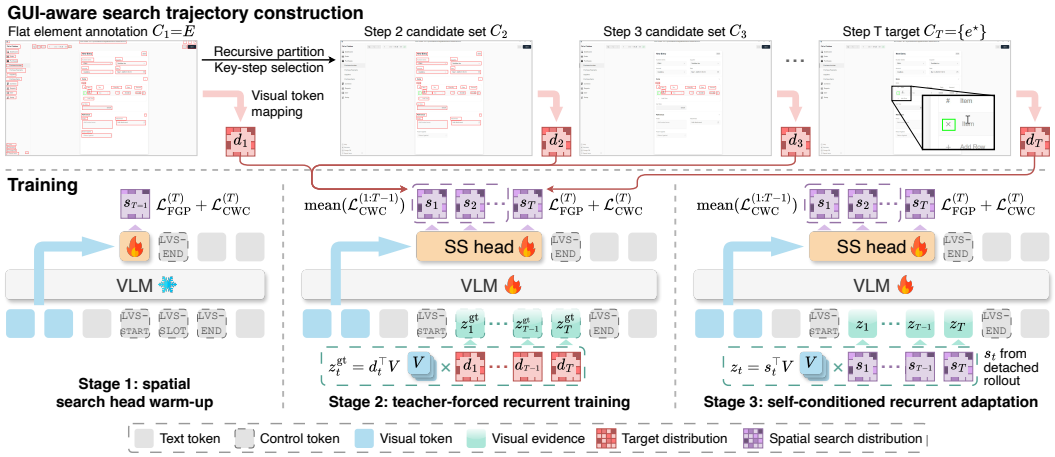


Figure 5: Overview of trajectory construction and training. *Top*: ReLaViS converts flat GUI element annotations E into a search trajectory $C_{1:T}$ and maps each candidate set C_t to a target distribution d_t over the visual token grid. *Bottom*: Stage 1 trains only the spatial search head with a frozen VLM and $T = 1$; Stage 2 uses teacher-forced inputs $z_t^{\mathrm{gt}} = d_t^\top V$; and Stage 3 uses self-conditioned inputs $z_t = s_t^\top V$ from a detached rollout. $\mathcal{L}_{\mathrm{CWC}}$ supervises all search steps, while $\mathcal{L}_{\mathrm{FGP}}$ additionally supervises the final step. "SS head" denotes the spatial search head.

```
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

The corresponding tool-call response is as follows:

```
<tool_call>
{"name": "computer_use", "arguments": {"action": "left_click", "lvr":
↪  <|object_ref_start|><|lvr|><|lvr|><|lvr|><|lvr|><|object_ref_end|>}}
</tool_call>
```

The exact tokens `<|object_ref_start|>`, `<|lvr|>`, and `<|object_ref_end|>` correspond to the semantic symbols `LVS-START`, `LVS-SLOT`, and `LVS-END`, respectively. The boundary tokens `<|object_ref_start|>` and `<|object_ref_end|>` already belong to the pretrained vocabulary, whereas we initialize `<|lvr|>` with the mean of the pretrained vocabulary embeddings. We exclude every `LVS-SLOT` and `LVS-END` position from the next-token prediction loss, while retaining supervision on `LVS-START` and the textual tool-call prefix and suffix.

Visual token grid. We use the default dynamic-resolution image preprocessing of Qwen2.5-VL, which resizes each screenshot while preserving its aspect ratio. The visual encoder partitions the resized screenshot into 14×14 pixel patches and initially produces a grid of size $G_h \times G_w$. The multimodal merger applies a spatial merge factor $M = 2$ to each 2×2 group of patch embeddings. The post-merge grid shared by the LLM backbone and the spatial search head therefore has dimensions $H_v = G_h/M$ and $W_v = G_w/M$, yielding $N = H_v W_v$ visual tokens, denoted as V. Each post-merge visual token corresponds to a 28×28 pixel region in the resized screenshot, and we constrain the resized image area to the range from 4×28^2 to $16{,}384 \times 28^2$, yielding 4 to 16,384 tokens per screenshot.

Spatial search head architecture. The SelfAttn block in Eq. (3) uses 8-head self-attention, a dropout rate of 0.1, a residual connection, and LayerNorm. The hidden state and visual token MLPs use independent `Linear-GELU-Linear` projections. Their intermediate width r equals the backbone hidden width d. The head contains $8d^2 + 10d$ parameters (102.8M with Qwen2.5-VL-7B), introducing only a lightweight parameter overhead.

A.2 GUI-AWARE SEARCH TRAJECTORY CONSTRUCTION

Fig. 5 summarizes GUI-aware search trajectory construction and the training procedure.

Algorithm 1 Constructing the target-containing recursive partition.

Require: Full element set E and target element $e^\star \in E$
Ensure: Raw partition path $(R_0, \ldots, R_L)$ with $R_0 = E$ and $R_L = \{e^\star\}$

$R_0 \leftarrow E$; $L \leftarrow 0$
while $|R_L| > 1$ **do**
  $\Pi(R_L) \leftarrow \emptyset$
  for all $\alpha \in \{x, y\}$ **do**
    $(e_1, \ldots, e_n) \leftarrow \mathrm{sort}(R_L, \mathrm{Center}_\alpha)$
    for $p = 1, \ldots, n-1$ **do**
      Add $(\{e_1, \ldots, e_p\}, \{e_{p+1}, \ldots, e_n\})$ to $\Pi(R_L)$
    end for
  end for
  $(C_a^\star, C_b^\star) \leftarrow \underset{(C_a, C_b) \in \Pi(R_L)}{\arg\min} [\mathrm{Area}(B(C_a)) + \mathrm{Area}(B(C_b))]$
  $R_{L+1} \leftarrow C_a^\star$ if $e^\star \in C_a^\star$, and $R_{L+1} \leftarrow C_b^\star$ otherwise
  $L \leftarrow L + 1$
end while
return $(R_0, \ldots, R_L)$

Algorithm 2 Selecting key steps and constructing a fixed-step trajectory.

Require: Raw partition path $(R_0, \ldots, R_L)$, number of steps $T \geq 2$, and local-reduction threshold k
Ensure: Fixed-step trajectory $P^\star = (C_1, \ldots, C_T)$, $C_1 = R_0 = E$, and $C_T = R_L = \{e^\star\}$

$\mathcal{R} \leftarrow \{1, 0\} \cup \left\{ \frac{|R_i| - |R_j|}{|R_i|} : 0 \leq i < j < L \right\}$
$P^\star \leftarrow ()$
for all $\rho \in \mathrm{sort}_\downarrow(\mathcal{R})$ **do**
  $P \leftarrow (R_0)$; $i \leftarrow 0$
  Define $\mathcal{J}(i, \rho) = \left\{ j : i < j < L,\ \frac{|R_i| - |R_j|}{|R_i|} \geq \rho,\ |R_j| - |R_{j+1}| > k \right\}$
  while $\mathcal{J}(i, \rho) \neq \emptyset$ **do**
    $j \leftarrow \min \mathcal{J}(i, \rho)$
    Append R_j to P; $i \leftarrow j$
  end while
  if $\mathrm{last}(P) \neq R_L$ **then**
    Append R_L to P
  end if
  if $|P| \geq T$ **then**
    $P^\star \leftarrow (R_0) \,\|\, \mathrm{suffix}_{T-1}(P)$
    return $P^\star$
  end if
  if $|P| > |P^\star|$ **then**
    $P^\star \leftarrow P$
  end if
end for
while $|P^\star| < T$ **do**
  Append R_L to $P^\star$
end while
return $P^\star$

Target-containing recursive partition. Alg. 1 starts from the full element set E, with $R_0 = E$, and recursively narrows the current element set R_i toward $e^\star$. To keep the partition search tractable, we restrict $\Pi(R_i)$ to bipartitions induced by cuts between adjacent elements sorted by horizontal or vertical center coordinates. Then the algorithm selects $(C_a^\star, C_b^\star)$ using the objective in Eq. (6) and retains the subset containing $e^\star$ as R_{i+1}. Ties in center coordinates are broken by original annotation order; for equal-area partitions, the algorithm favors the x-axis, then the first cut in sorted order. The recursion terminates at $R_L = \{e^\star\}$, producing the nested raw partition path $(R_0, \ldots, R_L)$.

Key-step selection and fixed-step trajectory. Alg. 2 removes redundant steps from the partition path and converts the result into a fixed-step trajectory. The cumulative-reduction condition $(|R_i| - |R_j|)/|R_i| \geq \rho$ requires sufficient reduction from the last retained step, while the local-reduction condition $|R_j| - |R_{j+1}| > k$ retains only steps followed by a substantial reduction in candidate count. For each $\rho \in \mathcal{R}$ in descending order, the algorithm constructs a new path P from R_0 by repeatedly selecting the earliest subsequent intermediate step satisfying both conditions. When no further intermediate step qualifies, it appends R_L. At the first threshold yielding $|P| \geq T$, it returns P unchanged if $|P| = T$; otherwise, it truncates P to R_0 and its last $T-1$ steps. If all paths are shorter than T, it pads the first longest path with copies of R_L to length T. The returned $P^\star = (C_1, \ldots, C_T)$ is the T-step trajectory used in the main text.

Visual token mapping. Let $b(v_i)$ be the rectangular patch corresponding to post-merge visual token v_i. For an element e, the corresponding visual token support is

$$\Omega(e) = \{v_i \in V : \mathrm{Area}(b(v_i) \cap b(e)) > 0\}. \tag{10}$$

The support includes every visual token whose patch has a positive-area intersection with the element bounding box. If $\Omega(e)$ is empty, we add the token whose patch contains the bounding box center, ensuring that every candidate element e has nonempty support $\Omega(e)$.

A.3 Search process supervision

Gaussian target distributions. Following V2P (Chen et al., 2026b), we construct the target distribution d_t using size-adaptive 2D Gaussians. For each element, we set $\mu_{x,e} = (x_{0,e} + x_{1,e})/2$, $\sigma_{x,e} = x_{1,e} - x_{0,e}$, $\mu_{y,e} = (y_{0,e} + y_{1,e})/2$, and $\sigma_{y,e} = y_{1,e} - y_{0,e}$. Let $\mathcal{G}(\cdot; \mu, \sigma)$ denote the density of a one-dimensional Gaussian distribution with mean μ and standard deviation σ. At step t, we integrate each candidate's Gaussian density over $b(v_i) \cap b(e)$, take the maximum across $e \in C_t$, and normalize over visual tokens:

$$\widetilde{d}_{t,i} = \max_{e \in C_t} \int_{(x,y) \in b(v_i) \cap b(e)} \mathcal{G}(x; \mu_{x,e}, \sigma_{x,e}) \mathcal{G}(y; \mu_{y,e}, \sigma_{y,e})\, dx\, dy, \quad d_{t,i} = \frac{\widetilde{d}_{t,i}}{\sum_j \widetilde{d}_{t,j}}. \tag{11}$$

The intersection truncates each density at its element boundary, and the maximum selects the largest contribution across candidates to define $\widetilde{d}_{t,i}$ for each visual token.

Loss weighting. Because the click coordinate directly depends on s_T, we assign equal weights to $\mathcal{L}_{\mathrm{CWC}}^{(T)}$ and the mean CWC loss over intermediate steps. The FGP weight controls the contribution of final peak shaping relative to CWC supervision. Validation-set selection yields $\lambda_{\mathrm{CWC}}^{(T)} = \lambda_{\mathrm{CWC}}^{(1:T-1)} = 5$ and $\lambda_{\mathrm{FGP}} = 1$. As shown in the ablation of loss weighting (Appendix C.3), the tested lower and higher intermediate CWC weights both reduce grounding accuracy relative to this setting.

A.4 Training and inference

Training procedure. In our implementation, we initialize the full response sequence with T `LVS-SLOT` tokens. Since s_T does not depend on z_T, we inject latent visual evidence only into the first $T-1$ slots, saving one no-gradient backbone forward pass in Stage 3. Stage 2 constructs the recurrent inputs directly from $d_{1:T-1}$ and V. Stage 3 fills the slots sequentially: for each $t < T$, a backbone probe and the spatial search head predict s_t without gradients using the previously filled slots, and the resulting z_t replaces the t-th slot embedding. The probe predictions are detached, while the aggregation $z_t = s_t^\top V$ remains differentiable with respect to V. After all inputs are prepared, one differentiable forward pass over the full sequence computes all T search distributions and the applicable losses in Eq. (9).

Forward-pass cost. We count LLM backbone forward passes, with visual tokens V computed once and reused during both training and inference. Stages 1 and 2 require one differentiable pass each, since their inputs require no backbone probes. Stage 3 requires $T-1$ no-gradient probes plus one differentiable pass, for T passes in total. Inference requires T backbone passes for search, one per predicted s_t, plus additional passes for autoregressive decoding of the tool-call prefix and suffix. Appendix B.4 reports training time, inference latency, FLOPs, and memory.

Coordinate readout. $\mathcal{L}_{\mathrm{FGP}}^{(T)}$ encourages s_T to concentrate around the target center, making the center of the highest-probability token a natural readout choice. However, each visual token covers

Table 3: Statistics of fixed-step trajectory construction over the 105,000 training and validation samples. Exact, truncated, and padded refer to mutually exclusive outcomes when adjusting the key-step-selected path to T steps. Candidate-set sizes are sample means in order from C_1 to C_T.

T	**Exact**	**Truncated**	**Padded**	**Mean candidate-set sizes,** $\lvert C_1\rvert \to \cdots \to \lvert C_T\rvert$
3	102,798 (97.90%)	0 (0.00%)	2,202 (2.10%)	$74.25 \to 11.35 \to 1.00$
4	94,159 (89.68%)	69 (0.07%)	10,772 (10.26%)	$74.25 \to 28.01 \to 10.11 \to 1.00$
5	78,449 (74.71%)	167 (0.16%)	26,384 (25.13%)	$74.25 \to 39.24 \to 19.19 \to 8.00 \to 1.00$

a patch of the screenshot, so selecting the peak-token center quantizes the predicted coordinate to the visual token grid. Even when the peak token contains the target center, its center can differ from the target center by up to half a patch width or height. For GUI elements smaller than or comparable to the patch size, this offset may place the predicted coordinate outside the target boundary. We therefore refine the coordinate using the local probability distribution around the peak. Based on validation performance, we use a 3×3 neighborhood around the peak token $v^\star = \arg\max_{v_i \in V} s_{T,i}$, clipped to the $H_v \times W_v$ grid. Let $\mathcal{N}(v^\star)$ denote the visual tokens in this neighborhood. Using zero-based column and row indices, we define the normalized patch center of v_i as

$$\mathrm{Center}(v_i) = \left((\mathrm{Col}(v_i) + \tfrac{1}{2})/W_v,\ (\mathrm{Row}(v_i) + \tfrac{1}{2})/H_v\right). \tag{12}$$

The final click coordinate is

$$\hat{\mathbf{c}} = \frac{\sum_{v_i \in \mathcal{N}(v^\star)} s_{T,i}\, \mathrm{Center}(v_i)}{\sum_{v_i \in \mathcal{N}(v^\star)} s_{T,i}}. \tag{13}$$

This local weighted average interpolates among the centers of the peak token and its adjacent tokens, mitigating patch-level quantization while excluding distant probability mass from the readout. Multiplying the two normalized components by the original image width and height gives pixel coordinates. The ablation in Appendix C.3 shows that this local readout outperforms both selecting the peak-token center and averaging over a 5×5 neighborhood.

B Implementation and reproducibility details

B.1 Dataset and constructed trajectories

Dataset sampling. We construct the sample pool from the 700K-instruction subset of GroundCUA (Feizi et al., 2026), whose authors collected expert demonstrations on desktop applications and deduplicated the resulting data to reduce interface redundancy. We discard samples with a missing screenshot, an unparseable target bounding box, or no valid click coordinate, leaving 697,892 valid grounding samples. Using random seed 42, we select 100,000 samples for training and 5,000 for validation.

Trajectory statistics. We report trajectory statistics over the 105,000 training and validation samples. Before key-step selection, the number of candidate sets in each target-containing recursive partition path, including both endpoints, has a mean of 9.36, a median of 9, and an interquartile range of 7–11. As illustrated by Alg. 2, we categorize fixed-step trajectory construction as exact, truncated, or padded. Based on validation performance, we set the local-reduction threshold to $k = 2$ for all settings and use $T = 4$ by default. Table 3 shows that truncation remains below 0.2% for $T \in \{3, 4, 5\}$, whereas padding increases with the target number of steps. At the default $T = 4$, the mean candidate-set size contracts from 74.25 to 28.01, 10.11, and 1.00. Moreover, 89.68% of samples provide exactly two intermediate key steps and require neither truncation nor padding. Compared with $T = 3$, four steps provide one additional intermediate step; compared with $T = 5$, they reduce padding from 25.13% to 10.26%, providing data-level context for the default $T = 4$.

Trajectory visualization. Fig. 6 presents constructed GUI-aware search trajectories for nine common applications spanning image editing, CAD, office productivity, video editing, scientific computing, reference management, media playback, and accounting. In these examples, the intermediate candidate sets narrow the search through broad functional regions and local element groups before reaching the target, illustrating a GUI-aware coarse-to-fine search process. Together with the close agreement with UI-tree hierarchies in Appendix C.4, these results illustrate how our trajectory construction can approximate the GUI hierarchy using flat GUI element annotations.

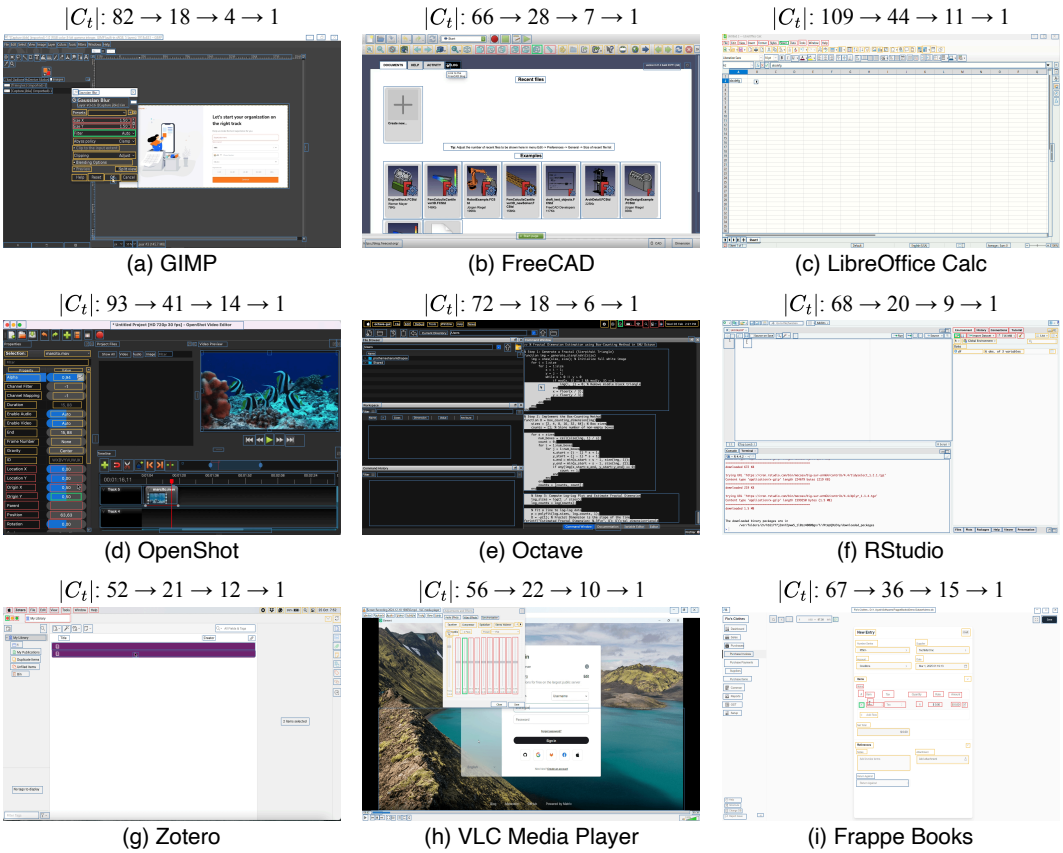


Figure 6: Constructed GUI-aware search trajectories with $T = 4$ across representative applications. The sequence above each panel reports $|C_1| \to \cdots \to |C_4|$, while blue, yellow, red, and green bounding boxes mark candidate elements in $C_1 \setminus C_2$, $C_2 \setminus C_3$, $C_3 \setminus C_4$, and $C_4 = \{e^\star\}$, respectively. The candidate sets contract from the full element set E through intermediate groups to the target.

B.2 Training Configuration

Model and trainable modules. We initialize ReLaViS-7B from Qwen2.5-VL-7B (Bai et al., 2025b) and randomly initialize the spatial search head. In Stages 2 and 3, we apply LoRA adapters (Hu et al., 2021) to all linear layers of the LLM backbone except the output head, using rank 32, $\alpha = 64$, and dropout 0.05. Stage 1 trains only the spatial search head with the VLM frozen. Stages 2 and 3 train the spatial search head, LoRA adapters, visual encoder, and multimodal merger while keeping the base weights of the LLM backbone and token embeddings frozen.

Optimization. All three stages use AdamW with weight decay 0.1 and a cosine learning-rate schedule with warm-up over the first 3% of steps in each stage. The spatial search head uses a learning rate of 10^{-4} in every stage; in Stages 2 and 3, the LoRA adapters in the LLM backbone, the multimodal merger, and the visual encoder use learning rates of 10^{-4}, 10^{-5}, and 2×10^{-6}, respectively. Training uses BF16 precision and a global batch size of 128 with 16 gradient-accumulation steps. Each stage runs for one epoch, corresponding to 782 optimization steps on the 100K training set.

Hyperparameters, random seeds, and checkpoint selection. We select k, T, and the FGP and CWC loss weights based on point grounding accuracy on the 5,000-sample GroundCUA validation set. The selected settings include $k = 2$, $T = 4$, and the loss weights described in Appendix A.3: unit weights for $\mathcal{L}_{\text{NTP}}$ and $\mathcal{L}_{\text{FGP}}^{(T)}$, and $\lambda_{\text{CWC}}^{(T)} = \lambda_{\text{CWC}}^{(1:T-1)} = 5$. To ensure the stability and reproducibility of results, we train each model with three random seeds (42, 43, and 44) in the main controlled comparisons, ablations, and analyses, and report the mean and standard deviation across runs. After each training stage completes all optimization steps, its final checkpoint initializes the

Table 4: Training cost. Times are approximate wall-clock hours on 4 NVIDIA L40-48G GPUs.

Model	Training time (h)			
	Stage 1	Stage 2	Stage 3	Total
Single-step-7B (coordinate-based)	∼30	∼30	∼30	∼90
Single-step-7B (coordinate-free)	∼11	∼29	∼29	∼69
ReLaViS-7B	∼11	∼29	∼48	∼88

next stage. We evaluate the final checkpoint of each run, without selecting checkpoints by validation or test performance.

B.3 Comparison and ablation configuration

Controlled comparisons. Single-step-7B (coordinate-based) directly generates a textual click coordinate. Single-step-7B (coordinate-free) sets $T = 1$ and uses the spatial search head to predict a spatial search distribution over visual tokens, followed by the same deterministic coordinate readout as ReLaViS-7B. All three models use the same 100K training set, evaluation protocol, and optimization settings described above. Since single-step-7B (coordinate-based) has no spatial search head, we train it with coordinate-generation SFT throughout all three stages, updating the same VLM modules as in Stages 2 and 3 of ReLaViS-7B.

Ablations and analyses. Except for the changes specified in each experiment, all ablation variants retain the ReLaViS-7B configuration. We evaluate these variants on ScreenSpot-Pro, whose high-resolution professional desktop interfaces and small targets provide a focused setting for examining complex grounding cases.

Thinking with images. The four baselines for thinking with images cover the mechanisms discussed in Sec. 2. MVP (Zhang et al., 2026b) employs multi-view aggregation, and UI-Zoomer (Tang et al., 2026a) utilizes adaptive zoom and region search. GUI-Spotlight (Lei et al., 2025) learns to use visual tools, and GUI-Cursor (Zhao et al., 2026) performs iterative refinement from visual feedback. MVP and UI-Zoomer are training-free; we reproduce them using their official implementations with Qwen2.5-VL-7B as the backbone. We directly evaluate the public GUI-Spotlight checkpoint, which is based on Qwen2.5-VL-7B. GUI-Cursor has no publicly released implementation at this time, so we use the results reported in its paper. We also implement a simple zoom-in baseline to explore the potential for combining ReLaViS with thinking with images. The baseline first grounds the instruction on the full screenshot and defines a window centered on the predicted click, with width and height each one-third of the corresponding screenshot dimension. If the window extends beyond the image boundaries, we shift it to fit within the screenshot while preserving its size. We then crop the selected region and resize it to the original screenshot resolution with bicubic interpolation, producing a $3\times$ linear zoom. The same model grounds the original instruction on this enlarged crop, and we map the prediction back to the original screenshot as the final output. We apply this identical implementation to Qwen2.5-VL-7B, single-step-7B (coordinate-free), and ReLaViS-7B.

B.4 Training and inference cost

Training cost. All training runs use 4 NVIDIA L40-48G GPUs with a per-device batch size of 2. Table 4 reports the approximate training time of each stage. ReLaViS-7B requires approximately 88 hours for the complete training procedure, comparable to the 90 hours required by single-step-7B (coordinate-based) and higher than the 69 hours required by single-step-7B (coordinate-free). The additional training time relative to single-step-7B (coordinate-free) occurs in Stage 3, where ReLaViS performs sequential no-gradient probes before the differentiable main pass, consistent with the forward-pass analysis in Appendix A.4.

Inference cost. We average interaction rounds across all five benchmarks and measure latency and peak allocated memory on ScreenSpot-Pro using a single L40 GPU, BF16 precision, a batch size of 1, and deterministic decoding. End-to-end latency includes image preprocessing, model generation, and prediction postprocessing. We analytically estimate FLOPs from each sample's actual token and patch counts and report the average in TFLOPs/sample. The estimate covers the visual encoder,

Table 5: Inference cost. Interaction rounds are averaged across all five benchmarks. Latency, FLOPs, and peak allocated GPU memory are reported for ScreenSpot-Pro.

Model	Interaction rounds	Latency (s/sample)	TFLOPs (/sample)	Peak memory (GiB)
Single-step-7B (coordinate-based)	1.0	2.29	172.00	18.98
Single-step-7B (coordinate-free)	1.0	2.06	173.51	18.78
ReLaViS-7B	1.0	2.19	179.65	19.54
Zoom-in + Qwen2.5-VL-7B	2.0	4.67	344.01	18.98
Zoom-in + single-step-7B (coordinate-free)	2.0	4.28	347.03	18.78
Zoom-in + ReLaViS-7B	2.0	4.57	359.31	19.54
MVP (Zhang et al., 2026b) + Qwen2.5-VL-7B	4.8	11.97	786.39	18.00
UI-Zoomer (Tang et al., 2026a) + Qwen2.5-VL-7B	2.0	2.34	148.70	39.67
GUI-Spotlight (Lei et al., 2025)	2.3	9.69	532.09	45.05

LLM backbone prefill and decoding, and method-specific operations, including the spatial search head and visual evidence propagation for ReLaViS. For thinking with images baselines, latency measurements and FLOPs estimates cover all interaction rounds. Peak allocated memory measures the maximum live tensor allocation over inference. Inference cost results for MVP, UI-Zoomer, and GUI-Spotlight serve as references under their official implementations.

As shown in Table 5, ReLaViS improves grounding accuracy while keeping inference cost close to the single-step baselines. Relative to single-step-7B (coordinate-free), ReLaViS-7B gains 3.1 pp in ScreenSpot-Pro accuracy (Table 1). The corresponding increases are 6.3% in mean latency, 3.5% in FLOPs, and 0.76 GiB in peak allocated memory. Compared with single-step-7B (coordinate-based), ReLaViS improves accuracy by 10.0 pp with 4.4% lower mean latency, 4.4% more FLOPs, and 0.56 GiB more peak allocated memory. Recurrent search thus improves grounding with modest increases in FLOPs and memory while keeping latency comparable to that of single-step grounding.

This favorable accuracy–efficiency trade-off also holds with external zoom-in. Zoom-in + ReLaViS achieves 63.5% ScreenSpot-Pro accuracy, compared with 62.3% for zoom-in + single-step-7B (coordinate-free), with 6.8% higher mean latency and 3.5% more FLOPs. Without zoom-in, ReLaViS already exceeds zoom-in + Qwen2.5-VL-7B in accuracy (56.3% versus 49.8%) while reducing latency by 53.1% and FLOPs by 47.8%.

B.5 Evaluation benchmarks and metrics

Benchmarks. We evaluate on ScreenSpot-Pro (Li et al., 2025b), MMBench-GUI-L2 (Wang et al., 2026c), OSWorld-G-Refine (Xie et al., 2025), ScreenSpot-v2 (Wu et al., 2025c), and UI-Vision (Nayak et al., 2025), spanning high-resolution professional desktop interfaces and general desktop, web, and mobile interfaces. As noted by the GroundCUA (Feizi et al., 2026) authors, UI-Vision overlaps with GroundCUA in terms of application coverage, and we therefore treat it as an in-domain benchmark. The cross-benchmark average is the arithmetic mean of the headline scores from the five benchmarks.

Search dynamics metrics. For a spatial search distribution s_t over N visual tokens, we measure its concentration using its entropy, defined as $\mathrm{Entropy}(s_t) = -\sum_{i=1}^{N} s_{t,i} \log s_{t,i}$. Thus, a lower value indicates a more concentrated spatial search distribution.

The probability mass of the target element $e^\star$ is $m_{t,e^\star} = \sum_{v_i \in \Omega(e^\star)} s_{t,i}$. A higher target probability mass indicates that more probability is assigned to the visual token support of the target element.

To measure proximity to the target element $e^\star$, let visual token v_i have coordinates (x_i, y_i) on an $H_v \times W_v$ grid, where $x_i \in \{0, \ldots, W_v - 1\}$ and $y_i \in \{0, \ldots, H_v - 1\}$. We define the distance from token v_i to the nearest target-support token in $\Omega(e^\star)$, with normalization by the grid diagonal, as

$$\mathrm{Distance}(v_i, e^\star) = \frac{\min_{v_j \in \Omega(e^\star)} \|(x_i, y_i) - (x_j, y_j)\|_2}{\sqrt{H_v^2 + W_v^2}}. \tag{14}$$

Table 6: Accuracy gains (pp) of ReLaViS-7B over matched single-step baselines across relative target area bins within each benchmark. Relative target area is the target bounding box area divided by the screenshot area. Gains are reported as mean$_{\pm std}$ over three random seeds; n is the instance count in each bin. Within each benchmark and baseline comparison, blue and red indicate the highest and lowest mean gains, respectively.

	vs. single-step (coordinate-based)			vs. single-step (coordinate-free)		
Benchmark	**Large** $\geq 0.25\%$	**Medium** $[0.09\%, 0.25\%)$	**Small** $< 0.09\%$	**Large** $\geq 0.25\%$	**Medium** $[0.09\%, 0.25\%)$	**Small** $< 0.09\%$
ScreenSpot-Pro	+13.9$_{\pm 4.81}$ $n = 36$	+4.5$_{\pm 3.81}$ $n = 273$	+11.1$_{\pm 0.94}$ $n = 1{,}272$	+0.9$_{\pm 4.24}$ $n = 36$	+2.6$_{\pm 1.68}$ $n = 273$	+3.3$_{\pm 0.69}$ $n = 1{,}272$
MMBench-GUI-L2	+0.1$_{\pm 0.72}$ $n = 1{,}649$	+1.2$_{\pm 0.47}$ $n = 857$	+6.6$_{\pm 0.74}$ $n = 1{,}088$	+0.8$_{\pm 0.53}$ $n = 1{,}649$	+0.7$_{\pm 0.34}$ $n = 857$	+1.0$_{\pm 0.69}$ $n = 1{,}088$
OSWorld-G-Refine	-0.5$_{\pm 2.20}$ $n = 131$	0.0$_{\pm 2.60}$ $n = 77$	+5.6$_{\pm 2.98}$ $n = 302$	+3.1$_{\pm 2.02}$ $n = 131$	-0.4$_{\pm 4.92}$ $n = 77$	+4.7$_{\pm 3.77}$ $n = 302$
ScreenSpot-v2	-0.3$_{\pm 0.52}$ $n = 1{,}019$	+2.8$_{\pm 2.02}$ $n = 200$	+11.9$_{\pm 2.88}$ $n = 53$	+0.3$_{\pm 0.34}$ $n = 1{,}019$	+0.5$_{\pm 2.18}$ $n = 200$	+0.6$_{\pm 2.18}$ $n = 53$
UI-Vision	-1.3$_{\pm 1.85}$ $n = 359$	+11.2$_{\pm 0.36}$ $n = 1{,}048$	+18.9$_{\pm 0.58}$ $n = 4{,}071$	+0.1$_{\pm 2.02}$ $n = 359$	+3.7$_{\pm 1.03}$ $n = 1{,}048$	+7.9$_{\pm 0.90}$ $n = 4{,}071$

The expected distance under the spatial search distribution s_t is

$$\text{ExpectedDistance}(s_t, e^\star) = \sum_{i=1}^{N} s_{t,i} \ \text{Distance}(v_i, e^\star). \tag{15}$$

Target-support tokens have zero distance; a lower expected distance indicates a search focus closer to the target element on the visual token grid.

C Additional results and analyses

C.1 Target-size analysis within each benchmark

To reduce confounding from benchmark composition in the pooled analysis (Fig. 3, left), we further examine the association between accuracy gains and target size within each benchmark. Table 6 reports the accuracy gains of ReLaViS-7B over the two matched single-step baselines across all five benchmarks, using the same three relative target area bins ordered from large to medium to small. Against the coordinate-based baseline, the small-target bin has the largest mean gain and the large-target bin has the smallest mean gain in four of the five benchmarks. Similarly, against the coordinate-free baseline, the small-target bin has the largest mean gain in all five benchmarks, and the large-target bin has the smallest mean gain in ScreenSpot-Pro, ScreenSpot-v2, and UI-Vision. These within-benchmark results further support the association observed in the pooled analysis and suggest greater gains on small targets.

C.2 Results at a smaller model scale

We evaluate ReLaViS on Qwen2.5-VL-3B to examine whether the gains from recurrent latent visual search persist at a smaller model scale. We compare ReLaViS-3B with single-step-3B (coordinate-free), its $T = 1$ variant. Both models use the same settings as their corresponding 7B models except for the smaller model scale and halved learning rates. Table 7 shows that ReLaViS-3B outperforms the single-step baseline on all five benchmarks, improving average accuracy by 3.1 pp, from 58.1% to 61.2%. Together with the 7B results in Table 1, these results demonstrate the effectiveness of ReLaViS across different model scales within the Qwen2.5-VL family.

C.3 Additional ablations

Coordinate readout. We compare four deterministic coordinate readouts on ScreenSpot-Pro while keeping the ReLaViS-7B model and evaluation setting fixed. Selecting the center of the highest-

Table 7: Performance comparison of models based on Qwen2.5-VL-3B across five GUI grounding benchmarks. For models trained in this work, accuracy (%) is reported as mean$_{\pm std}$ over three random seeds. † denotes a result reproduced using the official implementation. **Bold** values indicate the best results.

Model	ScreenSpot-Pro	MMBench-GUI-L2	OSWorld-G-Refine	ScreenSpot-v2	UI-Vision	Average
Zero-shot general VLMs						
Qwen2.5-VL-3B† (Bai et al., 2025b)	18.4	55.2	38.8	69.6	7.4	37.9
Controlled comparison (identical training data)						
Single-step-3B (coordinate-free)	$46.0_{\pm 0.89}$	$73.6_{\pm 0.41}$	$53.4_{\pm 0.10}$	$85.8_{\pm 0.28}$	$31.4_{\pm 0.06}$	$58.1_{\pm 0.17}$
ReLaViS-3B	$\mathbf{48.3}_{\pm 0.33}$	$\mathbf{75.6}_{\pm 0.12}$	$\mathbf{55.6}_{\pm 0.82}$	$\mathbf{87.3}_{\pm 0.33}$	$\mathbf{39.1}_{\pm 0.38}$	$\mathbf{61.2}_{\pm 0.18}$

Table 8: Agreement with reference trajectories derived from UI trees on Screen2AX-Tree. IoUs are averaged over targets within each screenshot and then across the 1,125 screenshots, which contain 29,901 targets in total. The two intermediate candidate sets are jointly matched to reference steps in coarse-to-fine order.

Search trajectory	C_2 IoU (%) ↑	C_3 IoU (%) ↑
GUI-aware	**75.1**	**74.7**
Geometric zoom	62.4	57.9

probability visual token yields $46.7_{\pm 0.48}\%$ accuracy. Region readout groups tokens whose probabilities exceed 30% of the highest token probability into connected regions and selects the region with the highest mean token probability. Taking its probability-weighted center improves accuracy to $53.5_{\pm 0.10}\%$. Local probability-weighted averaging around the peak performs better: a 5×5 neighborhood achieves $55.1_{\pm 0.40}\%$, whereas the default 3×3 neighborhood achieves the highest accuracy of $56.3_{\pm 0.29}\%$. The 9.6 pp gain over the peak-token center supports the use of local interpolation to mitigate patch-level quantization error. The 1.2 pp advantage over the broader 5×5 neighborhood suggests that a compact window limits the contribution of less relevant probability mass.

Intermediate CWC loss weight. We vary $\lambda_{\text{CWC}}^{(1:T-1)}$ while keeping all other training and evaluation configurations unchanged. This experiment examines how the weight on intermediate-step candidate-wise supervision affects point grounding accuracy. On ScreenSpot-Pro, the default weight of 5 yields $56.3_{\pm 0.29}\%$ accuracy, compared with $54.3_{\pm 1.27}\%$ at 15 and $52.9_{\pm 0.63}\%$ at 1. The tested lower and higher weights both reduce grounding accuracy relative to the default weight.

Intermediate supervision objective. We replace the intermediate CWC term in Eq. (9) with a direct token-level distributional objective that combines a KL loss and a global negative loss. The KL loss $\mathcal{L}_{\text{KL}}^{(t)} = D_{\text{KL}}(d_t \,\|\, s_t)$ matches s_t to the Gaussian target distribution d_t. The global negative loss $\mathcal{L}_{\text{neg}}^{(t)} = \sum_{v_i \notin \bigcup_{e \in C_t} \Omega(e)} s_{t,i}$ penalizes probability mass outside the visual token support of the candidate elements. We average each loss over the $T-1$ intermediate steps and include these averages in the overall objective with weights 1 and 5, respectively, and keep all other training and evaluation configurations unchanged. On ScreenSpot-Pro, this alternative achieves $54.8_{\pm 0.86}\%$ accuracy, 1.5 pp below CWC at $56.3_{\pm 0.29}\%$. These results favor candidate-wise coverage supervision over the tested token-level distributional objective.

C.4 Search trajectory and state analyses

Agreement with UI-tree hierarchies. We evaluate how closely our constructed trajectories approximate a GUI hierarchy using 1,125 macOS screenshots with nonempty UI-tree annotations from Screen2AX-Tree (Muryn et al., 2025). We treat the leaf nodes of the UI tree as a flat set of elements E. For each target in E, we follow its parent links to recover the root-to-target ancestor chain, which determines the reference trajectory steps. Starting from E, we intersect the current candidate set with the elements enclosed by the bounding box of each successive node in this chain (e.g., windows, groups, and toolbars) and retain only steps that strictly reduce the set, ending with the target element. For each target, we measure agreement with the reference GUI hierarchy by

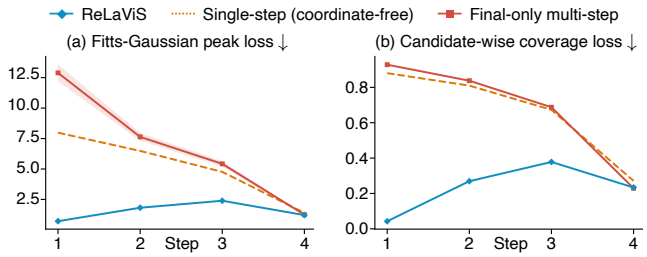


Figure 7: Stepwise diagnostic $\mathcal{L}_{\mathrm{FGP}}$ (a) and $\mathcal{L}_{\mathrm{CWC}}$ (b) on the 5,000-sample GroundCUA validation set. Curves show multi-seed means with std shading.

matching the two intermediate candidate sets C_2, C_3 of each evaluated four-step search trajectory to reference steps. We select the pair of reference steps that maximizes the sum of the two matched set IoUs while preserving coarse-to-fine order, with IoU computed over element ID sets. As shown in Table 8, the GUI-aware trajectory improves set IoU by 12.7 pp at C_2 and by 16.8 pp at C_3 over geometric zoom, indicating a closer approximation to the GUI hierarchy.

Stepwise search-state diagnostics. At each step, we evaluate $\mathcal{L}_{\mathrm{CWC}}^{(t)}$ against the candidate set C_t and the KL divergence used by $\mathcal{L}_{\mathrm{FGP}}$ against the Gaussian target distribution d_t. The intermediate-step KL serves only as a diagnostic and does not contribute to training. For the single-step model, we score its single predicted distribution against each of the four step-specific targets. Fig. 7 shows that ReLaViS yields substantially lower losses than the single-step (coordinate-free) and final-only multi-step variants at the first three search steps, while all three models have comparable losses at the final step. Lower intermediate $\mathcal{L}_{\mathrm{CWC}}$ indicates better candidate coverage, while lower diagnostic KL indicates closer agreement with d_t. These diagnostics show that ReLaViS's intermediate spatial search distributions align more closely with the constructed GUI-aware trajectories.

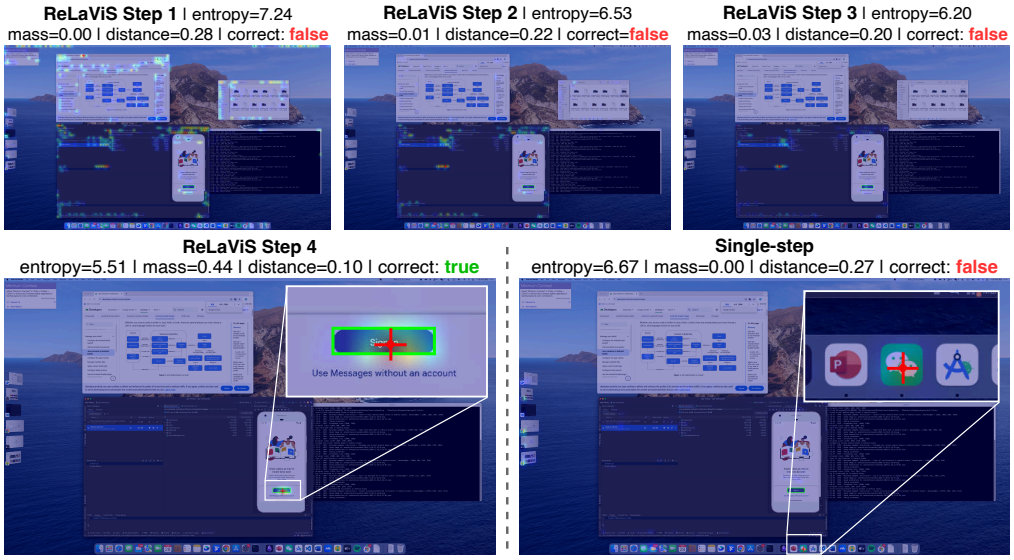
C.5 Visualizations and Qualitative Analyses

Figs. 8 and 9 present six successful cases across all five benchmarks and desktop, mobile, and web platforms. Fig. 10 presents three failure cases from ScreenSpot-Pro and ScreenSpot-v2. For each case, the figures visualize the spatial search distribution across four ReLaViS steps and report its entropy, target probability mass, expected distance to the target element, and click correctness, together with the matched single-step (coordinate-free) result. Across all six successful cases, entropy and expected distance decrease monotonically, target probability mass increases monotonically, and ReLaViS produces a correct final click where single-step (coordinate-free) fails. These case-level trajectories qualitatively complement the aggregate dynamics in Sec. 4.3.

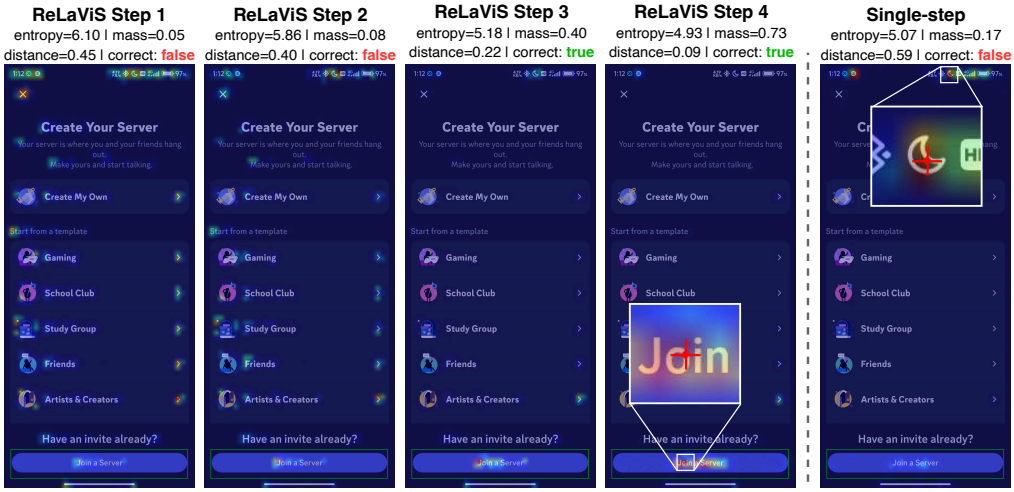
The successful cases illustrate how recurrent search supports candidate disambiguation, instruction-conditioned semantic grounding, and fine-grained localization. The ScreenSpot-Pro and UI-Vision cases in Figs. 8(a) and 9(c) show search states consistent with the GUI-aware trajectory: the former first emphasizes Android Studio over other windows; the latter reduces the probability mass assigned to text-related regions before selecting the target toolbar icon. Both examples illustrate candidate disambiguation as probability mass progressively shifts away from irrelevant regions. In Fig. 8(c), the search focus narrows from the full form to the Visibility Level group and then to the Private radio button. This progression shows that recurrent search grounds a compositional instruction by resolving both the relevant setting group and the required control role. In Fig. 8(b), "Twitch" does not match any visible interface label, yet the search focus converges on the interactive "Join a Server" control. This case shows that recurrent search remains grounded in the corresponding interactive control when the instruction is noisy. In Fig. 9(a), the search focus narrows to the toolbar and then to the connector drawing button adjacent to the referenced curve drawing button. In Fig. 9(b), it narrows to the dialog and then to the small checkbox beside "Blending Options". By resolving the enclosing functional region before distinguishing nearby controls, this coarse-to-fine progression supports fine-grained localization in dense layouts and for small targets.

In the failure case shown in Fig. 10(a), the search focus approaches the extremely small Format painter control by step 3. The deterministic final click lies visually closer to the target than the single-step (coordinate-free) prediction but remains outside the target boundary, consistent with

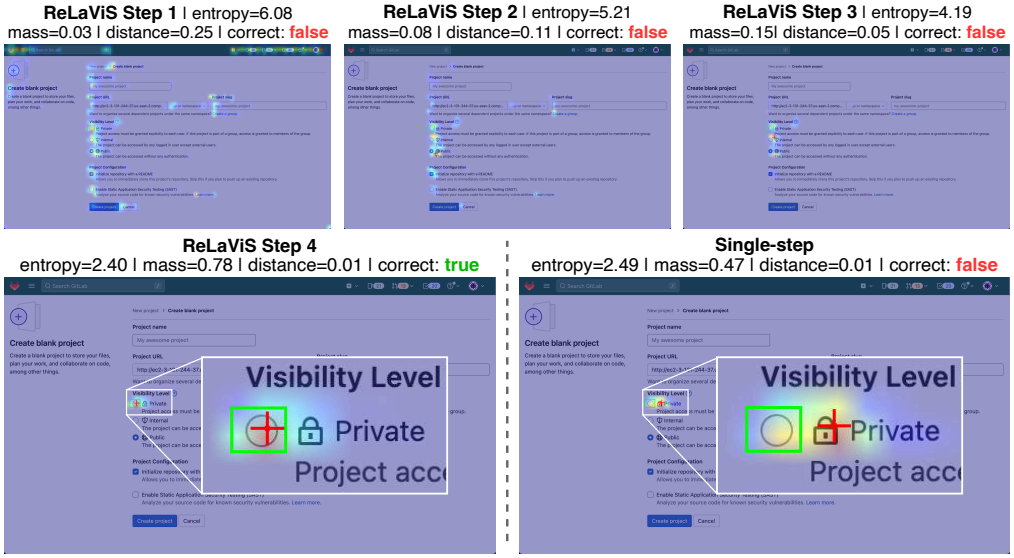
limited spatial resolution on the visual token grid. In Fig. 10(b), ReLaViS shifts from the annotated "Add Sources" control at step 3 to the plus icon in the Sources panel at step 4. Both controls are plausible entry points for "add sources", suggesting instruction–annotation ambiguity in this case. In Fig. 10(c), the search contracts to the brush toolbar but ultimately selects the color control instead of the annotated brush. This failure to associate the abstract visual style "watercolor" with the annotated control reveals a limited ability to distinguish abstract, non-standard visual semantics. These failure cases show that ReLaViS can narrow the search focus to the relevant region, while the final click still drifts under limited visual token grid resolution, instruction–annotation ambiguity, or a limited ability to distinguish abstract visual semantics.



(a) sign in google message in virtual android machine in android studio

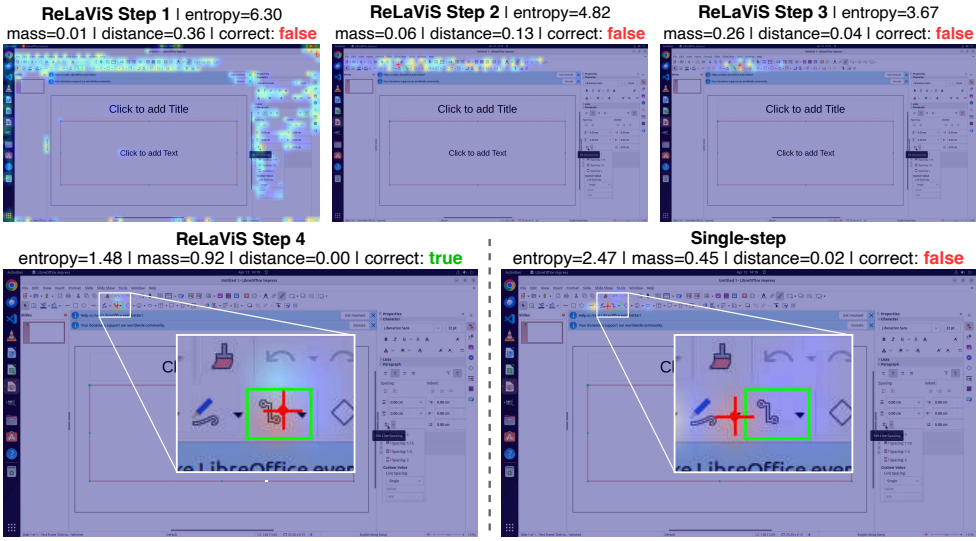


(b) join a twitch server

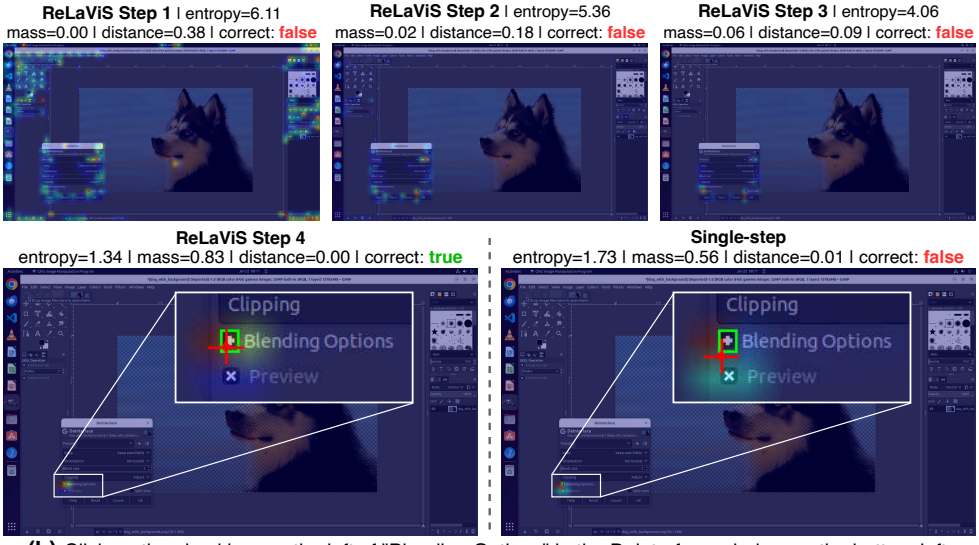


(c) switch to private visibility level

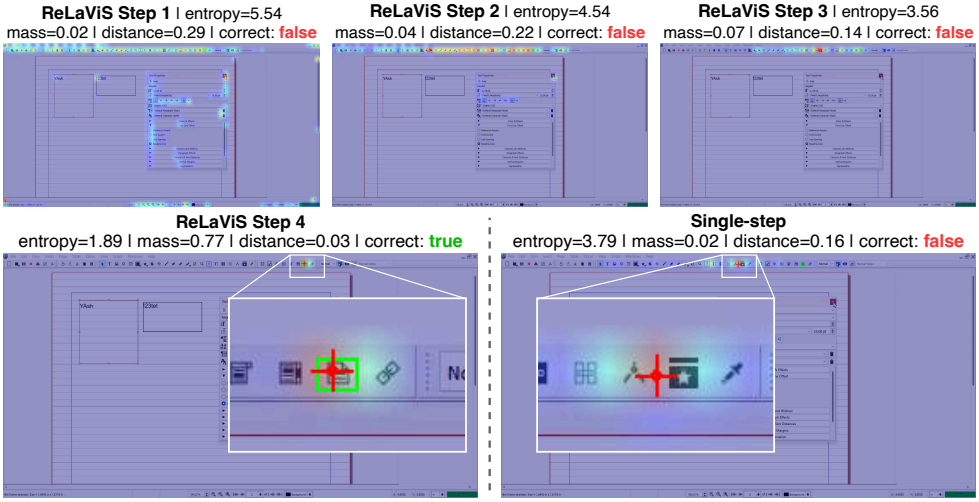
Figure 8: Successful cases from ScreenSpot-Pro (panel (a)) and ScreenSpot-v2 (panels (b) and (c)).



(a) A connector drawing button in the central toolbar of LibreOffice Impress, right to curve drawing button

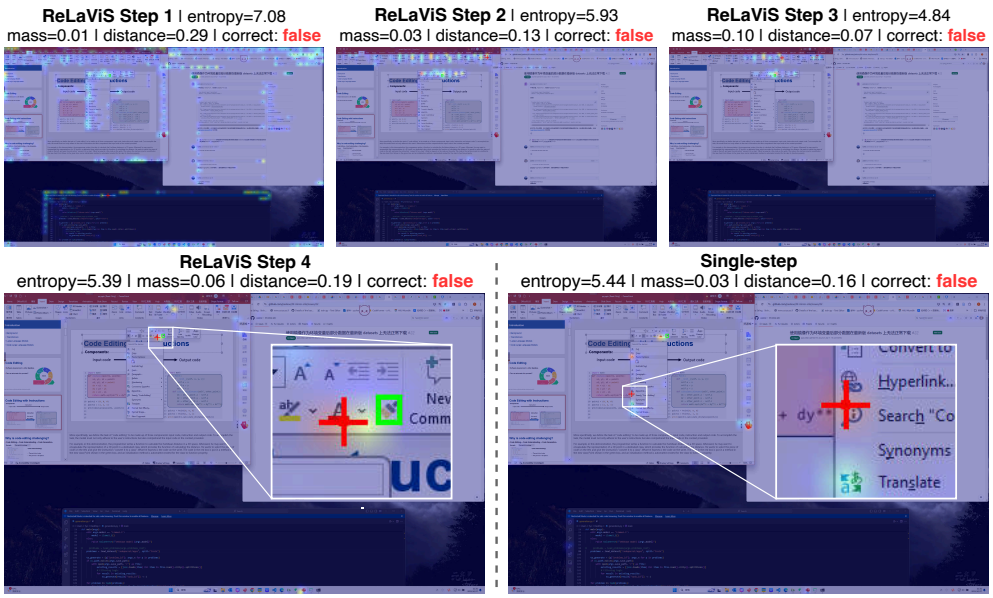


(b) Click on the checkbox on the left of "Blending Options" in the Deinterface window on the bottom left.

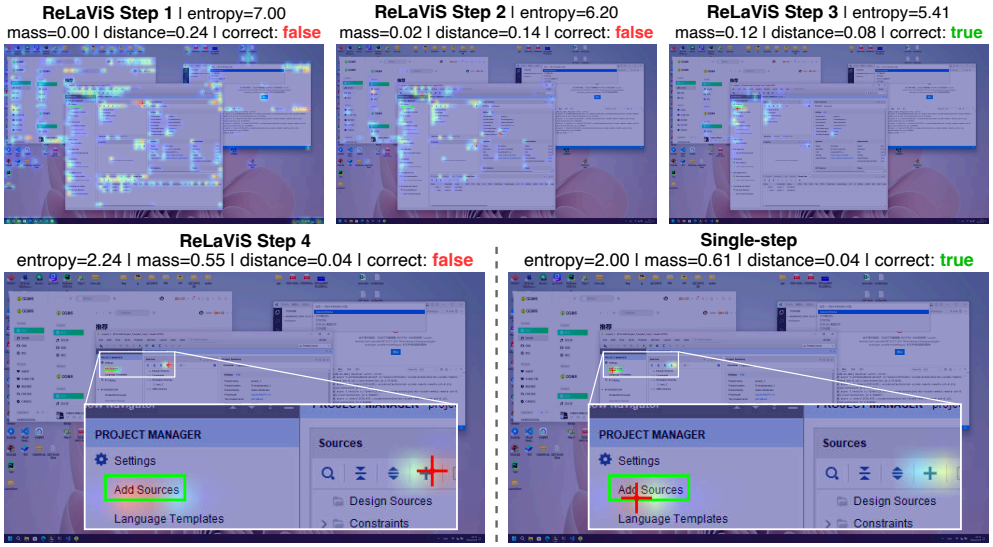


(c) Text Annotation

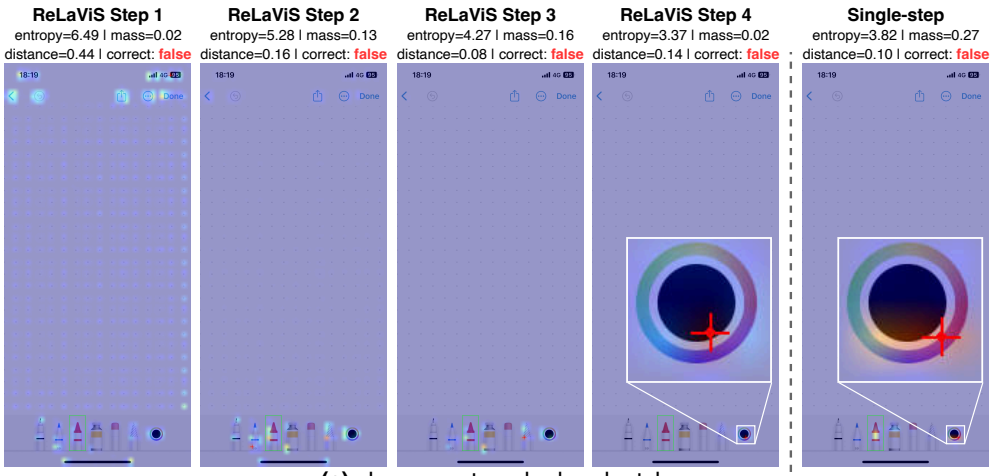
Figure 9: Successful cases from MMBench-GUI-L2 (panel (a)), OSWorld-G-Refine (panel (b)), and UI-Vision (panel (c)).



(a) Format painter



(b) add sources in vivado



(c) choose watercolor brush style

Figure 10: Failure cases from ScreenSpot-Pro (panels (a) and (b)) and ScreenSpot-v2 (panel (c)).

Table 9: Performance comparison on ScreenSpot-Pro.

Model	CAD		Dev		Creative		Scientific		Office		OS		Average		
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Avg
Zero-shot general VLMs															
GPT-4o (OpenAI, 2024)	2.0	0.0	1.3	0.0	1.0	0.0	2.1	0.0	1.1	0.0	0.0	0.0	1.3	0.0	0.8
Claude Computer (Hu et al., 2024)	14.5	3.7	22.0	3.9	25.9	3.4	33.9	15.8	30.1	16.3	11.0	4.5	23.4	7.1	17.1
Qwen2.5-VL-3B† (Bai et al., 2025b)	14.7	1.6	24.0	2.1	28.3	2.1	39.6	8.2	40.7	18.9	12.2	1.1	27.0	4.5	18.4
Qwen2.5-VL-7B† (Bai et al., 2025b)	21.8	1.6	46.8	4.8	36.4	7.0	56.3	8.2	55.9	20.8	39.3	10.1	41.9	7.8	28.8
Qwen2.5-VL-72B (Bai et al., 2025b)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	53.3
Qwen3-VL-8B (Bai et al., 2025a)	56.9	10.9	75.3	22.8	68.2	16.1	78.5	32.7	**80.8**	39.6	71.0	20.2	**71.1**	22.8	52.7
Qwen3-VL-30B-A3B (Bai et al., 2025a)	51.8	15.6	**76.0**	24.8	**69.2**	20.3	76.4	27.3	**80.8**	37.7	**75.7**	38.2	70.6	26.3	53.7
GUI-specialized models (coordinate-based)															
SE-GUI-7B (Yuan et al., 2025)	51.3	**42.2**	68.2	19.3	57.6	9.1	75.0	28.2	78.5	43.4	49.5	25.8	63.5	21.0	47.3
Jedi-7B (Xie et al., 2025)	38.0	14.1	42.9	11.0	50.0	11.9	72.9	25.5	75.1	47.2	33.6	16.9	52.6	18.2	39.5
UI-TARS-1.5-7B (Qin et al., 2025)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	49.6
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	64.7	19.6	47.5
InfiGUI-G1-7B (Liu et al., 2026b)	57.4	23.4	74.7	24.1	64.6	15.4	80.6	31.8	75.7	39.6	57.0	29.2	68.4	25.2	51.9
UI-Ins-7B (Chen et al., 2026c)	**60.9**	20.3	75.3	18.6	65.2	18.9	**81.3**	29.1	79.7	37.7	57.0	25.8	–	–	52.2
GroundNext-7B (SFT) (Feizi et al., 2026)	46.2	32.8	68.2	38.6	54.5	20.3	70.8	37.3	76.8	49.1	45.8	33.7	59.9	33.6	50.2
GroundNext-7B (RL) (Feizi et al., 2026)	50.2	34.3	73.4	**40.0**	59.6	23.8	70.1	42.7	74.6	54.7	53.3	30.3	60.5	33.6	52.9
GUI-specialized models (coordinate-free)															
GUI-Actor-7B (Wu et al., 2025b)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	44.6
V2P-7B (Chen et al., 2026b)	58.4	12.5	67.5	24.8	62.6	16.1	73.6	33.6	75.7	43.4	56.1	32.6	65.8	25.8	52.5
Controlled comparison (identical training data)															
Single-step-7B (coordinate-based)	$49.2_{\pm 3.66}$	$21.4_{\pm 1.81}$	$68.0_{\pm 0.99}$	$23.9_{\pm 1.05}$	$57.4_{\pm 0.77}$	$14.5_{\pm 1.76}$	$63.2_{\pm 6.02}$	$30.9_{\pm 0.91}$	$75.3_{\pm 2.55}$	$40.3_{\pm 5.76}$	$46.1_{\pm 0.54}$	$20.6_{\pm 3.44}$	$60.3_{\pm 2.04}$	$23.6_{\pm 0.42}$	$46.3_{\pm 1.40}$
Single-step-7B (coordinate-free)	$58.2_{\pm 1.47}$	$26.6_{\pm 1.56}$	$65.2_{\pm 4.32}$	$36.1_{\pm 3.11}$	$62.5_{\pm 0.58}$	$24.7_{\pm 1.76}$	$60.7_{\pm 1.75}$	$41.8_{\pm 1.58}$	$74.6_{\pm 1.96}$	$57.9_{\pm 4.75}$	$62.0_{\pm 2.16}$	$39.3_{\pm 3.37}$	$63.9_{\pm 0.39}$	$35.8_{\pm 1.33}$	$53.2_{\pm 0.52}$
ReLaViS-7B	$57.9_{\pm 2.03}$	$30.2_{\pm 2.39}$	$69.9_{\pm 2.28}$	$\mathbf{40.0}_{\pm 1.83}$	$67.2_{\pm 0.87}$	$\mathbf{25.6}_{\pm 0.81}$	$65.5_{\pm 2.89}$	$\mathbf{43.0}_{\pm 1.39}$	$78.3_{\pm 1.18}$	$\mathbf{62.3}_{\pm 4.99}$	$63.2_{\pm 1.08}$	$\mathbf{45.3}_{\pm 1.72}$	$67.1_{\pm 0.30}$	$\mathbf{38.9}_{\pm 0.82}$	$\mathbf{56.3}_{\pm 0.29}$
Thinking with images (external tools, multi-round)															
MVP† (Zhang et al., 2026b) + Qwen2.5-VL-7B	49.8	12.5	70.1	25.5	58.6	11.9	68.1	22.7	69.5	41.5	58.9	27.0	62.0	22.0	46.7
UI-Zoomer† (Tang et al., 2026a) + Qwen2.5-VL-7B	47.2	12.5	58.4	21.4	46.5	13.3	48.6	22.7	64.4	26.4	45.8	29.2	52.0	20.4	39.9
GUI-Spotlight† (Lei et al., 2025)	64.5	17.2	**75.3**	24.1	66.2	19.6	78.5	20.9	80.8	35.9	**65.4**	29.2	71.7	23.5	53.3
GUI-Cursor (Zhao et al., 2026)	**80.5**	33.1	65.7	18.2	62.4	25.0	**83.3**	32.7	84.2	43.4	**65.4**	31.5	73.3	29.3	56.5
Zoom-in + Qwen2.5-VL-7B	50.3	21.9	73.4	20.7	64.1	21.0	76.4	19.1	77.4	41.5	55.1	29.2	66.0	23.7	49.8
Zoom-in + single-step-7B (coordinate-free)	$70.7_{\pm 0.59}$	$32.3_{\pm 2.39}$	$73.6_{\pm 1.35}$	$\mathbf{49.4}_{\pm 3.80}$	$68.7_{\pm 2.81}$	$31.7_{\pm 0.40}$	$73.6_{\pm 0.70}$	$\mathbf{51.2}_{\pm 2.93}$	$81.7_{\pm 1.82}$	$66.7_{\pm 4.75}$	$64.5_{\pm 3.74}$	$53.2_{\pm 3.24}$	$72.5_{\pm 0.52}$	$45.8_{\pm 0.39}$	$62.3_{\pm 0.33}$
Zoom-in + ReLaViS-7B	$69.9_{\pm 1.06}$	$\mathbf{34.4}_{\pm 7.16}$	$75.1_{\pm 1.63}$	$\mathbf{49.4}_{\pm 3.11}$	$\mathbf{69.7}_{\pm 0.51}$	$\mathbf{33.6}_{\pm 2.52}$	$75.2_{\pm 0.40}$	$49.4_{\pm 1.89}$	$\mathbf{84.4}_{\pm 0.86}$	$\mathbf{75.5}_{\pm 1.89}$	$\mathbf{65.4}_{\pm 3.37}$	$\mathbf{55.4}_{\pm 1.72}$	$\mathbf{73.6}_{\pm 0.36}$	$\mathbf{47.2}_{\pm 1.54}$	$\mathbf{63.5}_{\pm 0.57}$

Table 10: Performance comparison on MMBench-GUI-L2.

Model	Windows		macOS		Linux		iOS		Android		Web		Avg
	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	
Zero-shot general VLMs													
GPT-4o (OpenAI, 2024)	1.5	1.1	8.7	4.3	1.1	1.0	5.1	3.3	2.5	1.4	3.2	2.9	2.9
Qwen2.5-VL-3B† (Bai et al., 2025b)	57.2	26.5	62.3	31.5	49.2	23.5	81.2	56.4	83.7	57.2	70.3	42.5	55.2
Qwen2.5-VL-7B† (Bai et al., 2025b)	68.6	45.2	70.7	54.6	48.7	35.7	94.9	82.4	93.8	80.9	86.5	70.8	71.9
GUI-specialized models (coordinate-based)													
SE-GUI-7B (Yuan et al., 2025)	–	–	–	–	–	–	–	–	–	–	–	–	34.5
Jedi-7B (Xie et al., 2025)	–	–	–	–	–	–	–	–	–	–	–	–	70.4
UI-TARS-1.5-7B (Qin et al., 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
GUI-G^2-7B (Tang et al., 2026b)	–	–	–	–	–	–	–	–	–	–	–	–	79.5
InfiGUI-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
GroundNext-7B (SFT) (Feizi et al., 2026)	83.8	60.7	**86.7**	69.9	75.4	61.2	94.3	83.3	94.9	79.4	91.0	70.5	80.4
GUI-specialized models (coordinate-free)													
GUI-Actor-7B (Wu et al., 2025b)	–	–	–	–	–	–	–	–	–	–	–	–	70.9
V2P-7B (Chen et al., 2026b)	–	–	–	–	–	–	–	–	–	–	–	–	79.9
Controlled comparison (identical training data)													
Single-step-7B (coordinate-based)	$82.9_{\pm 1.18}$	$55.4_{\pm 2.03}$	$82.0_{\pm 1.33}$	$66.5_{\pm 0.87}$	$71.6_{\pm 0.31}$	$49.0_{\pm 0.51}$	$\mathbf{95.0}_{\pm 0.36}$	$83.4_{\pm 0.93}$	$\mathbf{96.0}_{\pm 0.32}$	$81.3_{\pm 0.43}$	$91.2_{\pm 0.37}$	$71.3_{\pm 1.85}$	$78.7_{\pm 0.30}$
Single-step-7B (coordinate-free)	$82.2_{\pm 0.85}$	$57.0_{\pm 1.28}$	$84.9_{\pm 1.00}$	$\mathbf{72.1}_{\pm 0.88}$	$74.0_{\pm 1.84}$	$55.4_{\pm 2.12}$	$94.2_{\pm 1.21}$	$86.7_{\pm 0.81}$	$93.5_{\pm 0.74}$	$83.4_{\pm 1.23}$	$88.7_{\pm 1.41}$	$73.4_{\pm 1.17}$	$80.2_{\pm 0.55}$
ReLaViS-7B	$\mathbf{85.6}_{\pm 0.98}$	$58.8_{\pm 0.37}$	$84.4_{\pm 0.89}$	$71.5_{\pm 2.69}$	$\mathbf{76.1}_{\pm 1.09}$	$\mathbf{61.7}_{\pm 0.88}$	$93.7_{\pm 0.74}$	$85.4_{\pm 0.17}$	$93.0_{\pm 0.85}$	$83.6_{\pm 0.32}$	$89.4_{\pm 0.85}$	$76.1_{\pm 0.82}$	$\mathbf{81.1}_{\pm 0.08}$
Thinking with images (external tools, multi-round)													
MVP† (Zhang et al., 2026b) + Qwen2.5-VL-7B	63.1	41.9	54.2	45.7	44.5	39.3	69.4	63.6	56.7	61.4	54.5	54.6	55.0
UI-Zoomer† (Tang et al., 2026a) + Qwen2.5-VL-7B	77.1	50.0	81.7	60.1	75.4	49.5	90.8	73.6	**91.9**	72.7	89.0	65.3	74.2
GUI-Spotlight† (Lei et al., 2025)	77.5	50.4	77.4	53.8	63.9	45.9	91.7	75.2	85.7	70.4	88.7	70.1	72.2
GUI-Cursor (Zhao et al., 2026)	–	–	–	–	–	–	–	–	–	–	–	–	–
Zoom-in + Qwen2.5-VL-7B	81.9	58.8	81.2	67.9	69.6	52.6	90.1	**77.9**	90.2	**77.8**	91.0	72.4	77.3
Zoom-in + single-step-7B (coordinate-free)	$\mathbf{87.0}_{\pm 0.42}$	$\mathbf{64.6}_{\pm 1.81}$	$88.4_{\pm 1.00}$	$\mathbf{75.7}_{\pm 0.77}$	$81.0_{\pm 1.99}$	$61.9_{\pm 0.78}$	$88.6_{\pm 0.48}$	$77.8_{\pm 1.75}$	$89.5_{\pm 0.59}$	$76.9_{\pm 1.47}$	$90.8_{\pm 0.98}$	$74.8_{\pm 1.84}$	$80.5_{\pm 0.11}$
Zoom-in + ReLaViS-7B	$86.0_{\pm 1.33}$	$64.0_{\pm 1.32}$	$\mathbf{89.7}_{\pm 1.17}$	$75.4_{\pm 0.29}$	$\mathbf{82.2}_{\pm 1.05}$	$\mathbf{64.1}_{\pm 1.06}$	$\mathbf{91.8}_{\pm 0.49}$	$76.4_{\pm 1.09}$	$90.5_{\pm 0.65}$	$75.4_{\pm 0.59}$	$\mathbf{91.9}_{\pm 0.97}$	$\mathbf{75.3}_{\pm 0.56}$	$\mathbf{80.9}_{\pm 0.16}$

C.6 COMPLETE BENCHMARK RESULTS

Tables 9, 10, 11, 12, and 13 provide category-level results across the five benchmarks, supplementing the benchmark-level summary in Table 1. Following each benchmark's official taxonomy, the tables break down point grounding accuracy (%) by target type, platform, software domain, or task category. For models trained in this work, results are reported as mean$_{\pm std}$ over three random seeds. In these tables, "–" denotes unavailable results. Results marked with † are reproduced using official implementations. Results for other baselines from prior work are taken from the corresponding papers. Within each column, **bold** values are the best results within and outside "thinking with images" separately; underlined values are the second-best results only outside it.

Table 11: Performance comparison on OSWorld-G-Refine.

Model	Text matching	Element recognition	Layout understanding	Fine-grained manipulation	Overall
Zero-shot general VLMs					
Operator (OpenAI, 2025)	51.3	42.4	46.6	31.5	40.6
Qwen2.5-VL-3B† (Bai et al., 2025b)	52.1	41.5	46.6	22.4	38.8
Qwen2.5-VL-7B† (Bai et al., 2025b)	57.5	46.1	51.0	31.6	44.0
GUI-specialized models (coordinate-based)					
SE-GUI-7B (Yuan et al., 2025)	–	–	–	–	33.9
Jedi-7B (Xie et al., 2025)	65.9	55.5	57.7	46.9	54.1
UI-TARS-1.5-7B (Qin et al., 2025)	52.6	**75.4**	**72.4**	**66.7**	**64.2**
GUI-G^2-7B (Tang et al., 2026b)	–	–	–	–	61.9
InfiGUI-G1-7B (Liu et al., 2026b)	**72.0**	63.6	66.8	46.3	59.9
GUI-specialized models (coordinate-free)					
GUI-Actor-7B (Wu et al., 2025b)	–	–	–	–	47.0
V2P-7B (Chen et al., 2026b)	–	–	–	–	52.5
Controlled comparison (identical training data)					
Single-step-7B (coordinate-based)	68.2$_{\pm 1.01}$	64.7$_{\pm 0.17}$	65.2$_{\pm 1.05}$	47.6$_{\pm 0.38}$	59.6$_{\pm 0.20}$
Single-step-7B (coordinate-free)	67.4$_{\pm 1.01}$	66.9$_{\pm 1.75}$	68.4$_{\pm 2.20}$	38.4$_{\pm 0.38}$	59.3$_{\pm 1.18}$
ReLaViS-7B	68.8$_{\pm 0.80}$	71.4$_{\pm 1.43}$	71.2$_{\pm 0.00}$	40.4$_{\pm 2.01}$	62.5$_{\pm 1.01}$
Thinking with images (external tools, multi-round)					
MVP† (Zhang et al., 2026b) + Qwen2.5-VL-7B	61.7	53.9	60.9	36.8	51.2
UI-Zoomer† (Tang et al., 2026a) + Qwen2.5-VL-7B	61.7	60.9	61.3	40.4	54.9
GUI-Spotlight† (Lei et al., 2025)	68.2	60.6	63.6	45.4	56.7
GUI-Cursor (Zhao et al., 2026)	70.1	58.5	62.1	52.3	58.0
Zoom-in + Qwen2.5-VL-7B	67.1	66.4	70.0	40.1	59.2
Zoom-in + single-step-7B (coordinate-free)	**73.9**$_{\pm 1.01}$	75.0$_{\pm 0.93}$	77.3$_{\pm 1.60}$	**53.7**$_{\pm 1.66}$	**68.3**$_{\pm 1.07}$
Zoom-in + ReLaViS-7B	**73.9**$_{\pm 1.33}$	**75.3**$_{\pm 0.46}$	**77.6**$_{\pm 1.60}$	52.4$_{\pm 1.37}$	68.2$_{\pm 0.54}$

Table 12: Performance comparison on ScreenSpot-v2.

Model	Mobile		Desktop		Web		Avg
	Text	Icon	Text	Icon	Text	Icon	
Zero-shot general VLMs							
GPT-4o (OpenAI, 2024) + OmniParser-v2	95.5	74.6	92.3	60.9	88.0	59.6	80.7
Operator (OpenAI, 2025)	47.3	41.5	90.2	80.3	92.8	84.3	70.5
Qwen2.5-VL-3B† (Bai et al., 2025b)	83.3	64.9	76.1	45.2	73.2	62.0	69.6
Qwen2.5-VL-7B† (Bai et al., 2025b)	98.8	87.4	90.0	73.8	92.9	78.9	88.4
Qwen3-VL-8B (Bai et al., 2025a)	**99.7**	87.7	94.8	83.6	95.3	85.7	92.1
Qwen3-VL-30B-A3B (Bai et al., 2025a)	99.0	87.7	95.4	82.9	95.3	83.7	91.7
GUI-specialized models (coordinate-based)							
SE-GUI-7B (Yuan et al., 2025)	–	–	–	–	–	–	90.3
Jedi-7B (Xie et al., 2025)	96.9	87.2	95.9	87.9	94.4	84.2	91.7
UI-TARS-1.5-7B (Qin et al., 2025)	95.9	84.8	94.9	80.7	90.6	86.2	89.7
GUI-G^2-7B (Tang et al., 2026b)	98.3	**91.9**	95.4	89.3	94.0	87.7	93.3
InfiGUI-G1-7B (Liu et al., 2026b)	99.0	**91.9**	94.3	82.1	**97.9**	89.2	93.5
UI-Ins-7B (Chen et al., 2026c)	99.0	90.5	**97.9**	81.4	97.4	**91.6**	**94.0**
GroundNext-7B (SFT) (Feizi et al., 2026)	97.2	84.8	94.3	90.0	91.5	74.9	89.3
GroundNext-7B (RL) (Feizi et al., 2026)	96.6	88.2	95.4	87.9	94.9	75.9	90.4
GUI-specialized models (coordinate-free)							
GUI-Actor-7B (Wu et al., 2025b)	97.6	88.2	96.9	85.7	93.2	86.7	92.1
V2P-7B (Chen et al., 2026b)	98.1	88.0	96.1	89.7	95.4	84.4	92.4
Controlled comparison (identical training data)							
Single-step-7B (coordinate-based)	98.3$_{\pm 0.45}$	89.5$_{\pm 0.52}$	97.0$_{\pm 0.32}$	83.9$_{\pm 1.99}$	91.1$_{\pm 0.36}$	78.1$_{\pm 2.92}$	90.0$_{\pm 0.75}$
Single-step-7B (coordinate-free)	97.8$_{\pm 0.44}$	89.4$_{\pm 0.80}$	94.1$_{\pm 0.32}$	90.5$_{\pm 1.37}$	92.4$_{\pm 0.54}$	78.1$_{\pm 1.46}$	90.4$_{\pm 0.48}$
ReLaViS-7B	96.1$_{\pm 0.67}$	87.3$_{\pm 0.80}$	94.8$_{\pm 0.65}$	**92.3**$_{\pm 0.46}$	93.1$_{\pm 0.54}$	80.9$_{\pm 0.24}$	90.7$_{\pm 0.21}$
Thinking with images (external tools, multi-round)							
MVP† (Zhang et al., 2026b) + Qwen2.5-VL-7B	86.8	61.3	84.4	51.6	81.4	61.6	73.3
UI-Zoomer† (Tang et al., 2026a) + Qwen2.5-VL-7B	**98.5**	**85.9**	**97.2**	87.3	91.1	82.7	90.7
GUI-Spotlight† (Lei et al., 2025)	97.7	79.1	95.0	76.2	**93.2**	**84.0**	88.8
GUI-Cursor (Zhao et al., 2026)	–	–	–	–	–	–	**93.9**
Zoom-in + Qwen2.5-VL-7B	89.5	83.3	**97.2**	83.3	90.0	83.5	88.1
Zoom-in + single-step-7B (coordinate-free)	84.2$_{\pm 2.37}$	77.8$_{\pm 2.88}$	92.8$_{\pm 1.47}$	88.6$_{\pm 2.00}$	88.1$_{\pm 0.83}$	78.3$_{\pm 0.87}$	84.7$_{\pm 0.41}$
Zoom-in + ReLaViS-7B	84.2$_{\pm 0.98}$	76.4$_{\pm 1.39}$	93.3$_{\pm 0.00}$	**90.7**$_{\pm 2.42}$	87.5$_{\pm 0.71}$	79.7$_{\pm 0.43}$	84.9$_{\pm 0.59}$

Table 13: Performance comparison on UI-Vision.

Model	Basic	Functional	Spatial	Overall
Zero-shot general VLMs				
GPT-4o (OpenAI, 2024)	1.6	1.5	1.0	1.4
Qwen2.5-VL-3B† (Bai et al., 2025b)	5.3	11.9	5.2	7.4
Qwen2.5-VL-7B† (Bai et al., 2025b)	17.4	19.1	7.4	14.7
Qwen3-VL-8B (Bai et al., 2025a)	27.8	29.6	9.4	21.9
Qwen3-VL-30B-A3B (Bai et al., 2025a)	31.2	31.9	14.6	25.6
GUI-specialized models (coordinate-based)				
SE-GUI-7B (Yuan et al., 2025)	–	–	–	16.7
Jedi-7B (Xie et al., 2025)	32.3	30.5	12.8	24.8
UI-TARS-1.5-7B (Qin et al., 2025)	–	–	–	20.8
GUI-G^2-7B (Tang et al., 2026b)	–	–	–	25.6
InfiGUI-G1-7B (Liu et al., 2026b)	36.2	31.9	11.5	26.1
GUI-specialized models (coordinate-free)				
GUI-Actor-7B (Wu et al., 2025b)	–	–	–	21.9
V2P-7B (Chen et al., 2026b)	–	–	–	28.8
Controlled comparison (identical training data)				
Single-step-7B (coordinate-based)	$40.5_{\pm 0.30}$	$38.4_{\pm 0.70}$	$21.3_{\pm 0.30}$	$33.4_{\pm 0.35}$
Single-step-7B (coordinate-free)	$\underline{54.7}_{\pm 0.41}$	$\underline{50.9}_{\pm 0.67}$	$\underline{23.8}_{\pm 1.62}$	$\underline{43.1}_{\pm 0.42}$
ReLaViS-7B	$\mathbf{62.6}_{\pm 0.42}$	$\mathbf{57.9}_{\pm 0.55}$	$\mathbf{28.7}_{\pm 0.98}$	$\mathbf{49.7}_{\pm 0.35}$
Thinking with images (external tools, multi-round)				
MVP† (Zhang et al., 2026b) + Qwen2.5-VL-7B	24.3	24.0	10.4	19.6
UI-Zoomer† (Tang et al., 2026a) + Qwen2.5-VL-7B	31.0	28.4	12.5	23.9
GUI-Spotlight† (Lei et al., 2025)	31.6	30.2	8.8	23.5
GUI-Cursor (Zhao et al., 2026)	35.0	31.2	12.1	25.7
Zoom-in + Qwen2.5-VL-7B	28.1	28.7	14.7	23.9
Zoom-in + single-step-7B (coordinate-free)	$57.2_{\pm 0.57}$	$53.6_{\pm 0.34}$	$31.1_{\pm 0.54}$	$47.3_{\pm 0.44}$
Zoom-in + ReLaViS-7B	$\mathbf{62.5}_{\pm 0.09}$	$\mathbf{57.1}_{\pm 0.75}$	$\mathbf{37.3}_{\pm 0.80}$	$\mathbf{52.3}_{\pm 0.36}$