WHAT HAPPENS BEFORE DECODING? PREFILL DETERMINES GUI GROUNDING IN VLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing training-free approaches for GUI grounding often rely on multiple inference runs, such as iterative cropping or candidate aggregation, to identify target elements. Despite this additional computation, each forward pass still independently interprets the instruction and parses the visual layout, without enabling progressive interaction among visual tokens. In this paper, we study what happens during GUI grounding in Vision-Language Models (VLMs) and identify a previously overlooked bottleneck. We show that grounding follows a two-stage paradigm: the *prefill stage* determines candidate UI elements, while the *decoding stage* subsequently refines the final coordinates. This asymmetry establishes prefill as the critical step, as errors in candidate selection cannot be effectively corrected during decoding. Based on this observation, we propose **Re-Prefill**, a training-free method that revisits inference by introducing an attention-guided second prefill stage to refine target selection. Specifically, visual tokens that consistently receive high attention from the query position, *i.e.*, the final token, across layers are extracted as a preliminary target hypothesis and appended to the input, together with the instruction hidden states, enabling the model to deeply re-think its decision before coordinate generation. Experiments across four VLMs and five benchmarks, including ScreenSpot-Pro, ScreenSpot-V2, OSWorld-G, UI-Vision, and MMBench-GUI, demonstrate consistent improvements without additional training, with gains of up to 4.3% on ScreenSpot-Pro.

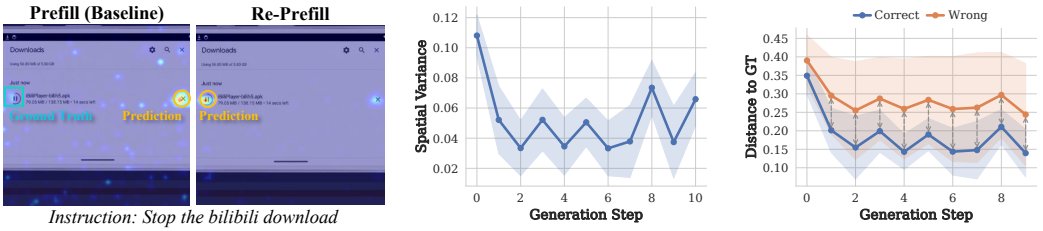
1 INTRODUCTION

GUI grounding aims to predict the coordinates of a target UI element from a screenshot guided by a natural language instruction, and is a fundamental capability for autonomous agents Nguyen et al. (2025); Zhang et al. (2025); Wang et al. (2024). Vision-Language Models (VLMs) have become the dominant approach to this task, achieving strong performance through large-scale supervised and reinforcement training (Yang et al., 2026; Tang et al., 2026b; Zhou et al., 2025; Xu et al., 2026). However, such performance relies on extensive annotation and substantial computing, leading to high labor and computational costs Hong et al. (2024); Wu et al. (2025b); Gou et al. (2025).

To reduce these costs, recent work has explored training-free methods that enhance grounding at inference time Lee et al. (2025); Luo et al. (2025b). These approaches typically rely on multiple inference runs, such as iterative cropping to refine visual resolution (Luo et al., 2025b; Jiang et al., 2025; Li et al., 2025a) or aggregating predictions across runs (Tang et al., 2026a; Zhang et al., 2026; Wu et al., 2025a). For example, ZoomClick (Jiang et al., 2025) characterizes key properties of zooming and performs iterative crop-and-zoom to increase the effective resolution around targets. CoG (Li et al., 2025a) uses context to refine iterative cropping, thereby improving grounding precision. MVP (Zhang et al., 2026) generates predictions from multiple cropped views and aggregates them to achieve more stable grounding accuracy. While effective, they incur substantial computational overhead. More importantly, our analysis reveals that stacking multiple runs does not enable interaction between visual tokens across runs, and therefore cannot correct errors introduced in earlier passes.

In this work, we attempt to address a more fundamental question: *where do grounding errors originate within inference?* We investigate this question by asking: *what happens before decoding?* We perform a thorough experimental analysis by tracing the query-position attention, *i.e.*the attention from the final token to the visual tokens, at each generation step. We then compute both its spatial

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.



(a) (b) (c)

Figure 1: **Prefill *vs.* Re-Prefill *vs.* Decoding.** *(a)* Query-position attention heatmaps over visual tokens. Re-Prefill produces a sharper, more focused distribution that disambiguates the correct target from other candidates. Additional visualizations are provided in Appendix D.1. *(b)* Spatial variance of query-position attention across generation steps. The sharp drop after the first generated token shows that target selection is largely completed during the prefill stage. *(c)* Deviation of the attention centroid from the ground-truth, separated into correct (blue) and incorrect (orange) predictions. The persistent gap in incorrect cases indicates that prefill errors cannot be corrected during decoding. Extended analysis across multiple models and benchmarks is provided in Appendix D.2.

variance, reflecting the focus of attention over image regions, and the deviation of its centroid from the ground-truth target. In Figure 1, we show the visual-token attention distribution from the query position at the prefill stage and the spatial variance of attention, the deviation of the attention centroid from the ground truth *w.r.t.* the decoding step. Step $t = 0$ on the horizontal axis represents the prefill stage, while $t = 1$ indicates the start of decoding. The results show that in prefill, attention is scattered on several candidate regions, coarsely determining the target element scope. During decoding, coordinates are generated autoregressively by focusing on one specific region, which can be demonstrated by attention variance decrease. The deviation from the ground truth remains large for incorrect predictions, indicating that errors introduced during prefill persist throughout decoding.

These observations establish the prefill stage as the decisive step for grounding, where accurate target selection is critical for subsequent decoding. Based on this insight, we propose **Re-Prefill**, a training-free method that revisits inference by introducing an attention-guided second prefill stage to refine attention concentration and improve target selection. Specifically, after the initial prefill computation, we identify visual tokens that consistently receive high attention from the query position across layers, which represent the model's preliminary target hypothesis. These tokens, as well as the encoded instruction hidden states, are appended to the original input for re-prefill computation, enabling the model to re-evaluate its target selection before coordinate generation. The illustration in Figure 1 illustrates that Re-Prefill improves target selection by enabling the model to focus on the correct UI element before decoding.

We evaluate Re-Prefill on four VLMs of varying scales (Zhou et al., 2025; Xu et al., 2026; Bai et al., 2025) across five GUI grounding benchmarks, including ScreenSpot-Pro (Li et al., 2025b), ScreenSpot-V2 (Wu et al., 2025b), OSWorld-G (Xie et al., 2025), UI-Vision (Nayak et al., 2025), and MMBench-GUI (Wang et al., 2025b). Re-Prefill consistently improves grounding accuracy across all models and benchmarks without additional training, achieving gains of up to 4.3% on ScreenSpot-Pro and 2.7% on OSWorld-G.

Our contributions are as follows: (1) We identify a two-stage inference paradigm in GUI grounding, where the prefill stage determines target selection and the decoding stage generates coordinates conditioned on it. (2) We propose Re-Prefill, a training-free method that improves target selection by revisiting inference with an attention-guided second prefill stage, orthogonal to existing GUI grounding methods. (3) We demonstrate consistent improvements across models and benchmarks, establishing prefill as a key bottleneck and providing a new perspective for studying GUI grounding.

2 Revisiting GUI Grounding Inference

2.1 Preliminaries

GUI grounding aims to predict the coordinates (x, y) of a target UI element given a screenshot I and a natural language instruction q. A decoder-only VLM processes the input as a sequence of tokens

composed of three segments: system prompt tokens $\mathbf{S}$, visual tokens $\mathbf{V}$, and instruction tokens $\mathbf{T}$. The concatenated input is denoted as $\mathbf{x} = [\mathbf{S}; \mathbf{V}; \mathbf{T}]$.

Inference consists of two stages. In the *prefill stage*, the full sequence $\mathbf{x}$ is processed through L decoder layers in a single forward pass, where self-attention is applied over all tokens and the resulting key–value pairs are stored in a KV cache $\mathcal{C}$. We denote this process as $f_{\text{prefill}}(\mathbf{x}) \to (\mathbf{h}_q, \mathcal{C})$, where $\mathbf{h}_q$ is the hidden state at the final token position (the query position), which is used to initiate coordinate prediction. In the subsequent *decoding stage*, coordinate tokens are generated autoregressively, with each step conditioned on the KV cache and previously generated tokens: $y_t = f_{\text{decode}}(y_{<t}, \mathcal{C})$.

2.2 Prefill as the Bottleneck of GUI Grounding

To better understand where grounding errors originate, we analyze the attention dynamics of VLMs during inference. In particular, we investigate two key questions: (1) when target region selection is determined, and (2) whether an incorrect selection can be corrected during decoding.

We employ a unified probing mechanism based on query-position attention. Let $\mathbf{a}_t \in \mathbb{R}^{N_v}$ denote the attention from the query position to visual tokens at generation step t, where $t = 0$ corresponds to the prefill stage and $t \geq 1$ to decoding steps. To reduce layer-wise variation, $\mathbf{a}_t$ is averaged across all decoder layers. All results are reported on ScreenSpot-Pro with Qwen3-VL-8B-Instruct, with extended analysis across multiple models and benchmarks provided in Appendix D.2.

Early attention concentration determines target selection. We first examine how attention evolves spatially across generation steps. The spatial concentration of $\mathbf{a}_t$ is measured by its variance,

$$\sigma_t^2 \;=\; \sum_{i=1}^{N_v} a_t(i)\,\big\|\mathbf{p}_i - \boldsymbol{\mu}_t\big\|^2, \quad \boldsymbol{\mu}_t \;=\; \sum_{i=1}^{N_v} a_t(i)\,\mathbf{p}_i, \tag{1}$$

where $\mathbf{p}_i$ is the 2-D coordinate of v_i and $\boldsymbol{\mu}_t$ is the attention centroid. A large σ_t^2 indicates dispersed attention, while a small value indicates concentration on a localized region. As shown in Figure 1(b), σ_t^2 is high at $t = 0$, suggesting that attention during prefill is distributed over multiple candidate regions. At $t = 1$, the variance drops sharply, indicating that attention rapidly concentrates on a single region once decoding begins. Thereafter, the variance remains low with only minor fluctuations.

This behavior suggests that target region selection is largely completed during the prefill stage, and that decoding primarily refines coordinates within the selected region. The small fluctuations during decoding likely arise from attending to local contextual layout, which helps improve coordinate precision. Similar patterns are observed across multiple benchmarks, indicating that this behavior is consistent across models and datasets.

Prefill errors persist throughout decoding. We next examine whether incorrect target selection can be corrected during decoding. Samples are divided into Correct and Wrong groups according to whether the predicted coordinates fall inside the ground-truth bounding box. For each group, we measure the normalized distance between the attention centroid $\boldsymbol{\mu}_t$ and the ground-truth center $\mathbf{g}$:

$$d_t \;=\; \frac{\big\|\boldsymbol{\mu}_t - \mathbf{g}\big\|}{D}, \tag{2}$$

where D is the image diagonal. As shown in Figure 1(c), both groups exhibit a decrease in distance from $t = 0$ to $t = 1$, consistent with the attention concentration behavior observed earlier. After $t = 1$, the trajectories diverge. The Correct group stabilizes near the target, whereas the Wrong group remains significantly farther away, with the gap persisting throughout decoding. This persistent discrepancy indicates that errors introduced during prefill are rarely corrected during decoding. This limitation arises from the autoregressive inference process, where decoding conditions on cached representations without recomputing attention over visual tokens. Consequently, misalignment formed during prefill propagates through all subsequent decoding steps.

These observations identify the prefill stage as a structural bottleneck in GUI grounding, as it determines target selection while its errors remain irreversible. Motivated by this finding, we propose to enable explicit re-evaluation of target selection before decoding, leading to the Re-Prefill method described in Section 3.

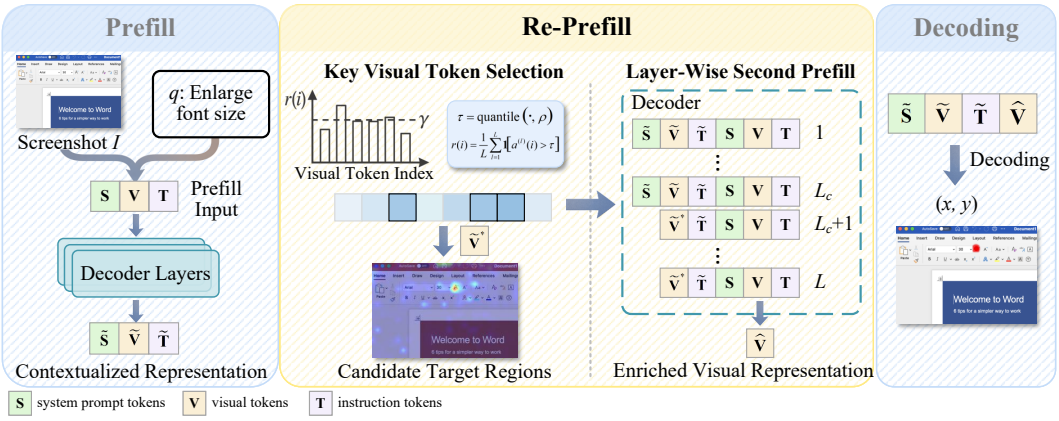


Figure 2: **Overview of Re-Prefill.** *(1) Prefill.* The input $[\mathbf{S}; \mathbf{V}; \mathbf{T}]$ is processed through L decoder layers to obtain contextualized representations $[\tilde{\mathbf{S}}; \tilde{\mathbf{V}}; \tilde{\mathbf{T}}]$. *(2) Key visual token selection.* Visual tokens that consistently receive high attention across layers are selected as $\tilde{\mathbf{V}}^*$, representing candidate target regions. *(3) Layer-wise second prefill.* A copy of the original input is re-encoded with layer-wise prefix modification, transitioning from full first-prefill representations to key visual tokens, producing enriched visual representations $\hat{\mathbf{V}}$. *(4) Decoding.* Coordinate tokens are generated autoregressively from the combined context $\mathcal{C}^* = [\tilde{\mathbf{S}}; \tilde{\mathbf{V}}; \tilde{\mathbf{T}}; \hat{\mathbf{V}}]$.

3 Re-Prefill

Building on the above analysis, we propose **Re-Prefill**, a training-free method that enhances GUI grounding through an attention-guided second prefill stage. Re-Prefill first identifies key visual tokens that encode candidate target regions from the initial prefill, and then performs a second prefill over the original input, where these tokens are incorporated as guidance to refine target region selection before decoding. An overview of the full pipeline is shown in Figure 2.

Step 1: Initial prefill and attention extraction. A standard prefill stage is first applied to the input $\mathbf{x} = [\mathbf{S}; \mathbf{V}; \mathbf{T}]$, producing contextualized representations $\tilde{\mathbf{S}}$, $\tilde{\mathbf{V}}$, and $\tilde{\mathbf{T}}$, together with the KV cache $\mathcal{C}_1$. To support subsequent visual token filtering, we additionally extract per-layer attention maps from the query position to visual tokens. Specifically, for each decoder layer l, the attention $\mathbf{a}^{(l)} \in \mathbb{R}^{N_v}$ is computed and averaged across heads to capture cross-layer visual relevance signals.

Step 2: Key visual token selection. The initial prefill produces contextualized visual representations $\tilde{\mathbf{V}}$ and query-to-visual attention maps. Based on qualitative visualizations (Section 4.2 and Appendix D.1), we observe that the prefill-stage attention is often distributed across multiple plausible UI regions, while the region corresponding to the final prediction typically receives relatively high attention. This suggests that the first prefill implicitly narrows the search space to a set of candidate target regions before selecting one for subsequent decoding. In this process, visual tokens with relatively high attention act as coarse indicators of these candidate regions. Motivated by this observation, we seek to re-evaluate the candidate regions before decoding, thereby increasing the likelihood of selecting the correct target. This requires identifying a subset of contextualized visual tokens that reliably represent the candidate regions. To this end, we introduce a cross-layer consistency filter over L decoder layers. We first define a global high-attention threshold τ as the ρ-quantile over all attention values:

$$\tau = \text{quantile}\big(\{a^{(l)}(i)\}_{l,i}, \rho\big). \tag{3}$$

For each visual token, we compute its cross-layer activation ratio:

$$r(i) = \frac{1}{L} \sum_{l=1}^{L} \mathbb{1}\big[a^{(l)}(i) > \tau\big]. \tag{4}$$

The *key visual tokens* are then defined as:

$$\tilde{\mathbf{V}}^* = \{\, \tilde{v}_i \mid r(i) \geq \gamma \,\}, \tag{5}$$

which correspond to tokens that consistently exhibit high attention, encoding candidate target regions.

Step 3: Layer-wise second prefill. To enable refined target selection, we perform a second prefill over a copy of the original uncontextualized input $[\mathbf{S}; \mathbf{V}; \mathbf{T}]$, guided by the representations obtained from the first prefill.

The key design is to progressively transition from full-context conditioning to focused conditioning on key visual tokens. This is achieved through a layer-wise prefix injection strategy controlled by a continuity hyperparameter L_c, which determines the layer at which the prefix is updated. Specifically, for layers 1 to L_c, we prepend the complete first-prefill representations, resulting in $[\tilde{\mathbf{S}}; \tilde{\mathbf{V}}; \tilde{\mathbf{T}}; \mathbf{S}; \mathbf{V}; \mathbf{T}]$, so as to preserve global semantic alignment. For layers $L_c + 1$ to L, we replace the full visual representations with the selected key visual tokens and remove $\tilde{\mathbf{S}}$, yielding $[\tilde{\mathbf{V}}^*; \tilde{\mathbf{T}}; \mathbf{S}; \mathbf{V}; \mathbf{T}]$, so that the model focuses on candidate target regions while retaining instruction context. Since the instruction segment is typically short and contains essential information, no additional selection is applied. After all L layers, the visual tokens in the uncontextualized input are updated into enriched representations, denoted as $\hat{\mathbf{V}}$, which are subsequently used for decoding.

The hyperparameter L_c controls the trade-off between preserving semantic continuity and enabling effective re-focusing. Its impact is analyzed in Section 4.3.

Step 4: Decoding. The final stage produces coordinate tokens through autoregressive decoding. The decoding context $\mathcal{C}^*$ is constructed by concatenating the first-prefill representations with the enriched visual tokens $\hat{\mathbf{V}}$ from the second prefill, forming $\mathcal{C}^* = [\tilde{\mathbf{S}}; \tilde{\mathbf{V}}; \tilde{\mathbf{T}}; \hat{\mathbf{V}}]$. Coordinate tokens are then generated autoregressively over $\mathcal{C}^*$. Pseudocode of the full procedure is provided in Appendix B.

4 Experiments

4.1 Experimental Setup

Benchmarks. Re-Prefill is evaluated on five benchmarks spanning desktop, mobile, and web interfaces, including ScreenSpot-Pro (Li et al., 2025b), ScreenSpot-V2 (Wu et al., 2025b), OSWorld-G (Xie et al., 2025), UI-Vision (Nayak et al., 2025), and MMBench-GUI-L2 (Wang et al., 2025b).

Base models. Four decoder-only VLMs are considered, including Qwen3-VL-8B-Instruct, Qwen3-VL-32B-Instruct (Bai et al., 2025), MAI-UI-8B (Zhou et al., 2025), and GUI-Owl-1.5-8B-Instruct (Xu et al., 2026). The two Qwen3-VL models are general-purpose VLMs without GUI-specific post-training, whereas MAI-UI-8B and GUI-Owl-1.5-8B-Instruct are designed specifically for GUI grounding. This selection covers the primary design configurations in current GUI agents and enables evaluation of whether Re-Prefill generalizes across different base model families.

Implementation details. Re-Prefill involves three hyperparameters, including the quantile threshold ρ, the ratio threshold γ, and the continuity layer L_c. The values $\rho = 0.8$ and $\gamma = 0.1$ are used throughout, with $L_c = 3$ for the 36-layer 8B models and $L_c = 6$ for the 64-layer 32B model. On ScreenSpot-Pro, OSWorld-G, UI-Vision, and MMBench-GUI-L2, a lightweight zoom-in strategy is applied to both the baseline and Re-Prefill. A square region centered on the initial prediction is cropped, upsampled by $2\times$, and then reprocessed by the same model to obtain a refined prediction. The zoom-in strategy is not applied on ScreenSpot-V2, as its low-resolution images already exhibit saturated performance without post-processing. The results for the base models are reproduced in this work under identical inference settings, whereas results for external baselines are taken from the original papers and are indicated accordingly in each table. We run 8B models on 8 RTX 4090 GPUs and the 32B model on 4 RTX PRO 6000 GPUs.

4.2 Main Results

Tables 1–5 report results on the five benchmarks. The two MVP variants (Zhang et al., 2026) share base models with our evaluation (MVP-8B uses Qwen3-VL-8B-Instruct and MVP-32B uses Qwen3-VL-32B-Instruct), enabling a direct comparison of inference-time strategies under identical underlying VLMs.

High-resolution and dense UI grounding. ScreenSpot-Pro evaluates grounding on high-resolution interfaces with dense layouts. As shown in Table 1, Re-Prefill consistently improves all four models,

Table 1: Results on ScreenSpot-Pro. Methods marked with † are reproduced using the zoom-in strategy adopted in this work.

Model	Development		Creative		CAD		Scientific		Office		OS		Overall
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	
UI-TARS-72B (Qin et al., 2025)	63.0	17.3	57.1	15.4	18.8	12.5	64.6	20.9	63.3	26.4	42.1	15.7	38.1
GTA1-32B (Yang et al., 2026)	83.1	37.9	72.2	25.9	70.1	31.3	84.7	39.1	89.3	64.2	76.6	51.7	63.6
UI-Ins-32B (Chen et al., 2026b)	51.8	29.7	83.1	26.9	69.7	18.9	83.3	34.5	88.7	50.9	70.1	34.8	57.0
GuirlVG (Kang et al., 2026)	64.9	7.6	42.9	11.2	28.9	9.4	63.9	16.4	63.8	26.4	43.9	13.5	36.1
GUI-Cursor-7B (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	56.5
MAI-UI-8B (Zhou et al., 2025)	78.6	58.6	78.8	46.9	80.7	43.8	86.1	49.1	88.1	81.1	76.6	51.7	70.9
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)	90.2	68.9	84.8	56.6	86.8	62.5	89.5	55.4	91.5	71.6	86.9	53.9	77.8
Training-free methods													
RegionFocus (Luo et al., 2025b)	75.3	25.5	76.3	30.8	71.6	28.1	87.5	39.1	87.0	60.4	74.8	36.0	61.6
UI-Zoomer (Tang et al., 2026a)	85.7	42.1	75.1	44.8	76.1	40.6	84.0	42.7	86.5	69.8	83.2	48.3	67.8
ZoomClick (Jiang et al., 2025)	88.3	55.9	82.8	38.5	80.7	42.2	90.2	40.9	93.8	77.4	84.1	51.7	72.1
MVP-8B (Zhang et al., 2026)	-	-	-	-	-	-	-	-	-	-	-	-	65.3
MVP-32B (Zhang et al., 2026)	-	-	-	-	-	-	-	-	-	-	-	-	74.0
Qwen3-VL-8B-Instruct (Bai et al., 2025)†	87.7	37.9	78.8	36.4	72.6	28.1	86.8	38.2	89.3	54.7	79.4	47.2	65.8
+ Re-Prefill	90.9	47.6	81.8	39.2	73.1	40.6	89.6	41.8	92.7	66.0	86.0	50.6	70.1$_{+4.3}$
MAI-UI-8B (Zhou et al., 2025)†	83.8	49.0	77.3	40.6	79.2	34.4	85.4	48.2	91.5	69.8	80.4	56.2	69.6
+ Re-Prefill	87.7	56.6	80.3	44.8	78.7	45.3	81.9	50.9	91.5	77.4	83.2	55.1	72.0$_{+2.4}$
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)†	81.2	69.7	83.8	58.0	84.8	57.8	93.1	65.5	93.2	69.8	85.1	61.8	78.0
+ Re-Prefill	88.3	73.8	85.9	58.0	85.8	64.1	91.0	65.5	93.2	75.5	86.9	67.4	80.1$_{+2.1}$
Qwen3-VL-32B-Instruct (Bai et al., 2025)†	89.6	49.7	85.4	46.2	83.8	45.3	89.6	47.3	94.4	77.4	91.6	53.9	74.3
+ Re-Prefill	93.5	54.5	87.4	51.7	83.8	43.8	94.4	49.1	94.9	77.4	88.8	64.0	76.8$_{+2.5}$

Table 2: Results on OSWorld-G. Methods marked with † are reproduced using the zoom-in strategy adopted in this work.

Model	Text Match.	Element Rec.	Layout Under.	Fine-grained Manip.	Refusal	Overall
UI-TARS-72B (Qin et al., 2025)	69.4	60.6	62.9	45.6	-	57.1
UI-TARS-1.5-7B (ByteDance Seed Team, 2025)	36.8	62.7	62.2	50.8	-	52.8
GTA1-32B (Yang et al., 2026)	63.2	78.4	73.3	65.2	-	65.2
MAI-UI-8B (Zhou et al., 2025)	72.8	67.6	71.1	56.4	-	64.2
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)	67.8	68.5	68.5	65.5	42.6	65.8
Qwen3-VL-8B-Instruct (Bai et al., 2025)†	72.8	66.7	67.2	53.7	-	63.0
+ Re-Prefill	75.5	69.7	71.2	54.4	-	65.7$_{+2.7}$
MAI-UI-8B (Zhou et al., 2025)†	75.5	70.3	70.0	53.7	-	65.5
+ Re-Prefill	76.6	71.8	74.7	55.7	-	67.7$_{+2.2}$
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)†	75.5	71.8	73.5	56.4	-	67.2
+ Re-Prefill	77.4	73.0	74.3	59.7	-	68.8$_{+1.6}$
Qwen3-VL-32B-Instruct (Bai et al., 2025)†	76.6	73.9	77.9	54.4	-	69.0
+ Re-Prefill	78.9	74.8	77.5	57.0	-	70.1$_{+1.1}$

with gains of $+4.3$ (Qwen3-VL-8B), $+2.4$ (MAI-UI-8B), $+2.1$ (GUI-Owl-8B), and $+2.5$ (Qwen3-VL-32B). For Qwen3-VL models, performance increases from 65.8% to 70.1% (8B) and from 74.3% to 76.8% (32B), outperforming MVP (65.3% and 74.0%) under the same backbone. For GUI-specialized models, Re-Prefill also yields consistent gains, improving MAI-UI-8B from 69.6% to 72.0% and GUI-Owl-1.5-8B from 78.0% to 80.1%, despite prior GUI-specific training. These results show that Re-Prefill complements training-based approaches by refining target selection at inference time. The larger gains on general-purpose models suggest that improving the prefill stage is particularly beneficial when initial target selection is less reliable.

Complex desktop interaction grounding. OSWorld-G evaluates grounding in desktop environments requiring structural UI understanding and interaction-aware reasoning. It provides original and refined annotation versions. This section reports results on the original annotations, while the refined results are provided in Appendix C.2. As shown in Table 2, Re-Prefill improves all models, with gains of $+2.7$ (Qwen3-VL-8B), $+2.2$ (MAI-UI-8B), $+1.6$ (GUI-Owl-8B), and $+1.1$ (Qwen3-VL-32B). Compared with ScreenSpot-Pro, improvements are smaller but more stable across model scales. Qwen3-VL improves from 63.0% to 65.7% (8B) and from 69.0% to 70.1% (32B), while MAI-UI-8B and GUI-Owl-1.5-8B increase from 65.5% to 67.7% and from 67.2% to 68.8%, respectively. These results indicate that Re-Prefill remains effective under complex interaction settings, improving both visual discrimination and structural reasoning. The smaller gains suggest that when grounding relies more on interaction structure than visual ambiguity, the benefit of prefill refinement becomes less pronounced but remains consistent.

Cross-platform UI grounding. ScreenSpot-V2 evaluates grounding across mobile, desktop, and web interfaces. As shown in Table 3, Re-Prefill improves all models, with gains of $+1.1$ (Qwen3-

Table 3: Results on ScreenSpot-V2. Methods marked with † are reproduced in this work.

Model	Mobile		Desktop		Web		Overall
	Text	Icon	Text	Icon	Text	Icon	
UI-TARS-72B (Qin et al., 2025)	94.8	86.3	91.2	87.9	91.5	87.7	90.3
GTA1-32B (Yang et al., 2026)	99.7	90.5	99.0	94.3	95.7	90.1	95.2
GUI-Cursor-7B (Zhao et al., 2026)	99.2	90.6	94.4	91.3	96.1	89.0	93.9
UI-Ins-32B (Chen et al., 2026b)	98.6	90.0	99.0	87.9	97.0	93.1	94.9
GuirlVG (Kang et al., 2026)	98.3	89.6	94.3	80.7	95.7	86.2	91.9
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)	97.4	90.5	96.4	90.7	94.2	89.7	93.7
Qwen3-VL-8B-Instruct (Bai et al., 2025)†	99.3	88.2	97.9	87.9	96.6	87.2	93.6
+ Re-Prefill	99.7	89.6	97.9	90.0	97.0	90.1	94.7$_{+1.1}$
MAI-UI-8B (Zhou et al., 2025)†	99.0	89.6	97.9	91.4	97.4	91.6	94.2
+ Re-Prefill	99.3	90.0	97.9	92.1	97.9	91.6	95.3$_{+1.1}$
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)†	98.6	90.0	97.4	89.3	96.2	88.2	93.9
+ Re-Prefill	98.3	91.0	97.9	92.1	96.2	89.7	94.6$_{+0.7}$
Qwen3-VL-32B-Instruct (Bai et al., 2025)†	99.0	91.5	98.5	90.0	96.6	91.1	95.0
+ Re-Prefill	99.3	91.9	98.5	91.4	97.4	90.1	95.3$_{+0.3}$

VL-8B), +1.1 (MAI-UI-8B), +0.7 (GUI-Owl-8B), and +0.3 (Qwen3-VL-32B). Improvements are smaller than on ScreenSpot-Pro and OSWorld-G due to the already high baseline performance (above 93%), leaving limited room for improvement. Despite this saturation, gains remain consistent across model types and platforms, including both mobile and desktop interfaces. These results suggest that Re-Prefill generalizes well across diverse UI domains, acting as a robust inference-time enhancement even when performance is near saturation.

Instruction-diverse grounding. We further evaluate Re-Prefill on instruction-heavy grounding scenarios using UI-Vision and MMBench-GUI-L2, which require robust query understanding under varying levels of semantic abstraction. On UI-Vision (Table 4), Re-Prefill improves all models, with gains of +1.0 (Qwen3-VL-8B), +1.3 (MAI-UI-8B), +0.4 (GUI-Owl-8B), and +0.9 (Qwen3-VL-32B). On MMBench-GUI-L2 (Table 5), the gains increase to +2.2, +1.3, +0.6, and +1.1, respectively. The consistent improvements across both benchmarks indicate that Re-Prefill remains effective under diverse instructions. Compared with scenarios dominated by visual ambiguity or structural complexity, these results suggest that refining the prefill stage also benefits semantic alignment between instructions and UI elements, enabling more robust grounding under varying query formulations.

Table 4: Results on UI-Vision. Methods marked with † are reproduced using the zoom-in strategy adopted in this work.

Model	Basic	Func.	Spat.	Overall
UI-TARS-72B (Qin et al., 2025)	31.4	30.5	14.7	25.5
MAI-UI-8B (Zhou et al., 2025)	51.6	50.5	26.6	42.4
Training-free methods				
UI-Zoomer (Tang et al., 2026a)	42.5	40.1	19.7	33.7
ZoomClick (Jiang et al., 2025)	45.1	45.1	28.3	39.2
MVP-8B (Zhang et al., 2026)	38.3	38.8	18.7	31.9
MVP-32B (Zhang et al., 2026)	49.4	52.0	30.8	44.1
Qwen3-VL-8B-Instruct (Bai et al., 2025)†	38.2	39.9	19.3	32.1
+ Re-Prefill	39.4	41.7	19.3	33.1$_{+1.0}$
MAI-UI-8B (Zhou et al., 2025)†	50.7	50.4	27.0	42.2
+ Re-Prefill	52.3	52.8	27.0	43.5$_{+1.3}$
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)†	50.1	44.7	29.8	41.2
+ Re-Prefill	50.1	46.1	29.7	41.6$_{+0.4}$
Qwen3-VL-32B-Instruct (Bai et al., 2025)†	49.9	51.1	31.1	43.7
+ Re-Prefill	51.4	51.0	32.4	44.6$_{+0.9}$

Qualitative analysis. Figure 3 visualizes query-position attention across stages for a representative example, comparing the baseline with Re-Prefill. In the baseline, attention at prefill is dispersed over multiple candidate regions and may contract onto an incorrect region during decoding. In contrast, Re-Prefill produces more focused attention on the correct target during prefill, leading to more accurate localization in subsequent decoding. This observation aligns with the two-stage inference structure in Section 2.2, where target selection is determined during prefill and propagated through decoding. Additional qualitative results are provided in Appendix D.1.

Overall, Re-Prefill consistently improves grounding accuracy across diverse settings. A detailed analysis of computational efficiency is provided in Appendix F.

Table 5: Results on MMBench-GUI-L2. Methods marked with † are reproduced using the zoom-in strategy adopted in this work.

Model	Windows		MacOS		Linux		iOS		Android		Web		Overall
	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	
UI-TARS-72B (Qin et al., 2025)	78.6	51.8	80.3	62.7	68.6	51.5	90.8	81.2	93.0	80.0	88.1	68.5	74.3
UI-TARS-1.5-7B (ByteDance Seed Team, 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
UI-Ins-32B (Chen et al., 2026b)	84.9	68.4	88.4	73.4	68.6	56.1	96.5	91.2	97.2	92.4	94.8	85.1	84.9
MAI-UI-8B (Zhou et al., 2025)	92.3	74.3	90.7	86.4	81.2	67.3	97.1	90.0	97.5	92.7	95.8	86.0	88.8
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)	89.7	65.4	88.1	72.8	72.8	56.6	95.9	83.9	95.2	82.9	93.2	77.6	82.5
Qwen3-VL-8B-Instruct (Bai et al., 2025)†	90.4	70.6	86.7	73.4	77.5	62.8	95.2	85.8	95.5	86.8	94.8	78.2	84.2
+ Re-Prefill	90.0	73.5	89.0	78.3	82.2	67.9	96.2	86.7	96.1	88.5	95.5	82.1	86.4$_{+2.2}$
MAI-UI-8B (Zhou et al., 2025)†	93.0	71.0	91.9	78.6	83.2	67.9	96.5	87.6	98.0	88.2	96.4	83.1	87.2
+ Re-Prefill	93.0	74.6	93.0	80.6	83.8	71.4	95.9	90.0	97.8	91.3	95.5	84.4	88.5$_{+1.3}$
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)†	92.6	65.8	88.4	73.1	82.2	58.2	97.1	86.4	97.5	83.9	95.8	80.2	84.5
+ Re-Prefill	90.8	68.8	87.8	74.3	82.7	61.7	96.5	86.7	97.8	85.1	95.5	81.2	85.1$_{+0.6}$
Qwen3-VL-32B-Instruct (Bai et al., 2025)†	93.4	71.3	90.4	80.3	86.9	70.4	97.1	90.3	97.8	91.5	96.8	84.1	88.4
+ Re-Prefill	92.6	72.4	91.9	84.7	85.9	76.0	97.5	91.8	96.9	91.5	97.1	86.4	89.5$_{+1.1}$

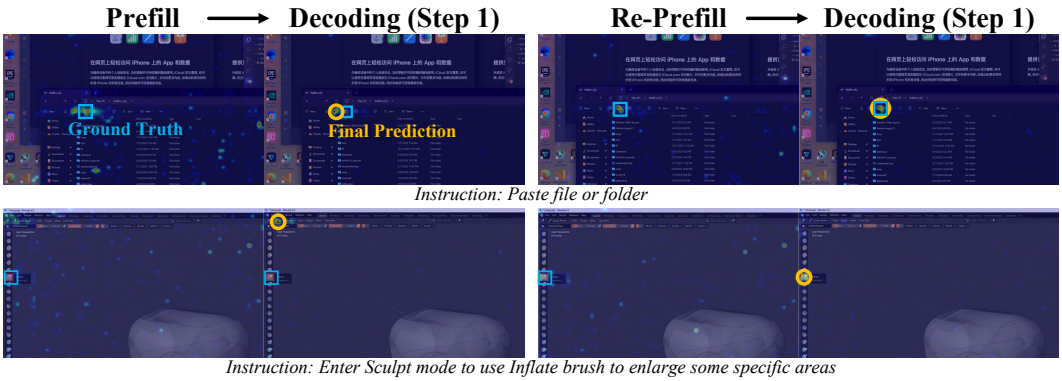


Figure 3: **Query-position attention heatmaps across stages on ScreenSpot-Pro.** The first two panels illustrate the baseline transition, while the last two panels show the corresponding transition under Re-Prefill. The blue rectangle marks the ground-truth target, and the orange circle indicates the predicted coordinate. Re-Prefill focuses attention on the correct region during prefill, suppresses distractors, and leads to more accurate localization in subsequent decoding.

4.3 Ablation Studies

Ablation studies are conducted on ScreenSpot-Pro with Qwen3-VL-8B-Instruct as the base model.

Analysis of the Re-Prefill mechanism. Three variants are designed to verify the contribution of the Re-Prefill mechanism (Table 6). *Embedding Addition* reuses the first-prefill visual tokens $\tilde{\mathbf{V}}$ in the decoding context instead of the enriched $\hat{\mathbf{V}}$, thereby skipping the second prefill stage. *Blind Re-Prefill* omits key-token selection and appends the full contextualized visual tokens $\tilde{\mathbf{V}}$ in the second prefill. *Random Token Selection* replaces attention-based token selection with random sampling. With zoom-in, all three variants perform worse than Re-Prefill. The largest gap is observed for Embedding Addition (66.0% vs. 70.1%), confirming that the second prefill provides the primary improvement. The gaps to Blind Re-Prefill (69.4%) and Random Token Selection (69.1%) further show the benefit of focusing the second pass on attention-derived key visual tokens. Without zoom-in, Re-Prefill still improves the baseline from 54.8% to 58.1%, confirming that its effectiveness does not depend on zoom-in post-processing. The larger gap to Blind Re-Prefill (51.7%) further suggests that key-token selection is particularly

Table 6: Analysis of the Re-Prefill mechanism. Embedding Addition removes the second prefill and uses the first-prefill visual tokens in decoding. Blind Re-Prefill omits selection and uses all visual tokens. Random Token Selection replaces attention-based selection with random sampling.

Method	w/o Zoom-in	w/ Zoom-in
Baseline	54.8	65.8
Embedding Addition	54.6	66.0
Blind Re-Prefill	51.7	69.4
Random Token Selection	56.9	69.1
Re-Prefill (Ours)	**58.1**	**70.1**

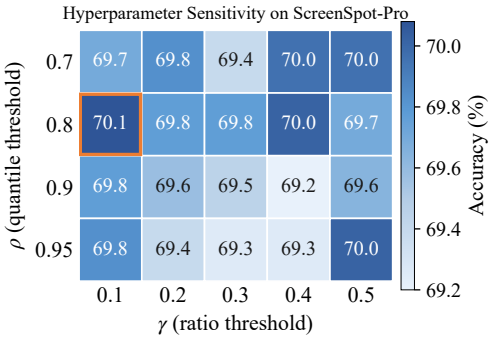


Figure 4: Effect of ρ and γ. The orange box marks the default configuration (ρ=0.8, γ=0.1). All twenty cells exceed the baseline (65.8%) with a spread of less than one point, indicating that Re-Prefill is largely insensitive to these thresholds.

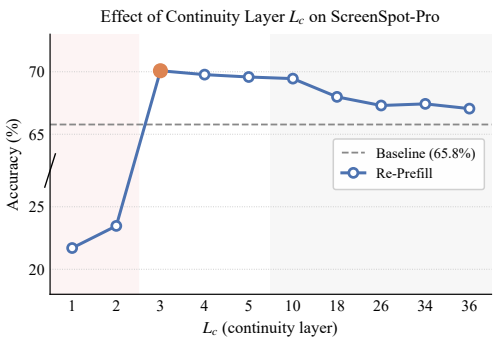


Figure 5: Effect of L_c. The optimum at L_c=3 balances two modes. For small L_c, insufficient semantic alignment arises between uncontextualized input tokens and the first-prefill prefix (red zone). For large L_c, noise from unrelated tokens propagates into deeper layers (grey zone).

important for suppressing irrelevant visual information in dense high-resolution inputs, while zoom-in partially reduces such interference through cropping.

Analysis of hyperparameters ρ and γ. The selection of $\tilde{\mathbf{V}}^*$ depends on the quantile threshold ρ (Eq. 3) and the ratio threshold γ (Eq. 5). Figure 4 presents a 4×5 grid over $\rho \in \{0.70, 0.80, 0.90, 0.95\}$ and $\gamma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. All twenty configurations outperform the 65.8% baseline, and the variation across the grid remains within 1.0 point (69.2% to 70.1%), indicating that Re-Prefill is largely insensitive to these thresholds. The best configuration ($\rho = 0.8$, $\gamma = 0.1$) lies at the more permissive end of both axes, expanding $\tilde{\mathbf{V}}^*$ to include not only the most highly attended region but also neighboring candidates that compete for attention. This design allows the second prefill stage to recover when the initial selection corresponds to an incorrect UI element. This configuration is adopted in all main results.

Analysis of continuity layers L_c. L_c determines how many early decoder layers retain the full prefix from the first prefill before it is reduced to the key visual tokens $\tilde{\mathbf{V}}^*$ and instruction tokens $\tilde{\mathbf{T}}$. A sweep of L_c from 1 to 36 is conducted on Qwen3-VL-8B-Instruct ($L = 36$). Figure 5 shows that the accuracy drops to approximately 22% for $L_c \in \{1, 2\}$, reaches a maximum of 70.1% at $L_c = 3$, and then decreases monotonically. This behavior reflects the dual role of L_c. Early layers ($\leq L_c$) use the full contextualized representations from the first prefill as a prefix to align the copy of uncontextualized input tokens with it. Later layers ($> L_c$) retain only $\tilde{\mathbf{V}}^*$ and $\tilde{\mathbf{T}}$ so that subsequent processing concentrates on candidate regions without interference from unrelated tokens. When L_c is small, the prefix is truncated before semantic alignment is established, causing later layers to operate on uncalibrated inputs with incomplete cues, which leads to severe performance degradation. When L_c is large, unrelated tokens propagate into deeper layers and dilute attention away from candidate regions, thereby reducing accuracy. The optimal value at $L_c = 3$ balances these two failure modes.

5 Conclusion

This work analyzes the inference dynamics of decoder-only VLMs for GUI grounding and identifies a two-stage paradigm in which target selection is completed during the prefill stage, while decoding only refines coordinates within the selected region. The prefill stage therefore constitutes the decisive factor for grounding accuracy, yet it remains unaddressed by existing multi-run methods. Motivated by this observation, Re-Prefill is proposed as a training-free approach that calibrates the prefill stage through attention-guided re-evaluation, in which key visual tokens and instruction tokens are reintroduced for a second prefill stage prior to coordinate decoding. Across four VLMs and five benchmarks, consistent improvements are achieved without additional training, reaching gains of up to 4.3 points on ScreenSpot-Pro and 2.7 points on OSWorld-G. These results establish prefill calibration as a principled direction for inference-time computation in vision-language models.

AI USE STATEMENT

Generative AI tools were used only to assist with polishing the writing. They were **not** used to develop the research methodology, conduct experiments, analyze experimental results, generate code, or produce figures and tables. All AI-assisted content was reviewed and revised by the authors to ensure technical correctness, originality, and consistency with the actual research. The authors take full responsibility for the final content of this work, including all text, claims, and results.

ETHICS STATEMENT

This work does not involve human subjects or user studies. All benchmarks and model checkpoints used in our experiments are publicly available. All procedures strictly conform to the guidelines established by the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We provide the implementation details required to reproduce Re-Prefill throughout the main paper and appendix. The experimental setup (Section 4.1) specifies the model checkpoints, inference settings, hyperparameters, and the hardware used for evaluation. The pseudocode of our method is provided in Appendix B, while robustness and hyperparameter sensitivity analyses are reported in Appendix E. All evaluated models are obtained from publicly available checkpoints, and all benchmarks used in this work are publicly available. The experiments contain stochastic components. The reported results follow the experimental protocol described in the paper. We will publicly release the implementation code to facilitate reproduction of the reported results.

REFERENCES

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

ByteDance Seed Team. Ui-tars-1.5: A multimodal ui understanding and reasoning model, 2025. URL https://seed-tars.com/1.5.

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

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 *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=dsQHm7YX9c.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024.

Yong Du, Yuchen Yan, Fei Tang, Zhengxi Lu, Chang Zong, Weiming Lu, Shengpei Jiang, and Yongliang Shen. Test-time reinforcement learning for gui grounding via region consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30593–30601, 2026.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kxnoqaisCT.

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

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14281–14290, 2024.

Zhiyuan Jiang, Shenghao Xie, Wenyi Li, Wenqiang Zu, Peihang Li, Jiahao Qiu, Siqi Pei, Lei Ma, Tiejun Huang, Mengdi Wang, et al. Zoom in, click out: Unlocking and evaluating the potential of zooming for gui grounding. *arXiv preprint arXiv:2512.05941*, 2025.

Weitai Kang, Bin Lei, Gaowen Liu, Caiwen Ding, and Yan Yan. GuirlVG: Incentivize GUI visual grounding via empirical exploration on reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zrH2AlupAo.

Hyunseok Lee, Jeonghoon Kim, Beomjun Kim, Jihoon Tack, Chansong Jo, Jaehong Lee, Cheonbok Park, Sookyo In, Jinwoo Shin, and Kang Min Yoo. Reguide: Data efficient gui grounding via spatial reasoning and search. *arXiv preprint arXiv:2505.15259*, 2025.

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. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ViMgQW9fyt.

Aiden Yiliu Li, Bizhi Yu, Daoan Lei, Tianhe Ren, and Shilong Liu. Chain-of-ground: Improving gui grounding via iterative reasoning and reference feedback. *arXiv preprint arXiv:2512.01979*, 2025a.

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*, pp. 8778–8786, 2025b.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for generalist gui agent. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025a.

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*, 2025b.

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 *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Anthony Nguyen. Improved gui grounding via iterative narrowing. *arXiv preprint arXiv:2411.13591*, 2024.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, et al. Gui agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22522–22538, 2025.

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

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

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

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

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

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, et al. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*, 2024.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, et al. Opencua: Open foundations for computer-use agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, et al. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. *arXiv preprint arXiv:2507.19478*, 2025b.

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 *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26257–26267, 2025a.

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 *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=n9PDaFNi8t.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

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

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, Jinrui Ding, Xiandi Ma, Yuchen Xie, Peng Pei, Xunliang Cai, and Xipeng Qiu. EvoCUA: Evolving computer use agents via learning from scalable synthetic experience. In *Second Workshop on Agents in the Wild: Safety, Security, and Beyond*, 2026.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, Ran Xu, Liyuan Pan, Caiming Xiong, and Junnan Li. GTA1: GUI test-time scaling agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3VIPmz7iAi.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22418–22433, 2025.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=IbzDaIDyt6.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Large language model-brained GUI agents: A survey. *Transactions on Machine Learning Research*, 2025.

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*, 2026.

Yu Zhao, Wei-Ning Chen, Huseyin A 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. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=dqmt0Yd4rc.

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

APPENDIX OVERVIEW

This appendix is organized as follows:

- Appendix A: Related Work
- Appendix B: Pseudocode of Re-Prefill
- Appendix C: Additional Evaluation Results
- Appendix D: Analysis of Prefill-Stage Dynamics
- Appendix E: Analysis of Robustness and Sensitivity
- Appendix F: Analysis of Computational Efficiency
- Appendix G: Limitations

A RELATED WORK

A.1 GUI GROUNDING

GUI grounding is a core capability for autonomous GUI agents (Nguyen et al., 2025; Zhang et al., 2025; Wang et al., 2024; Tang et al., 2025). Early approaches address this task through supervised fine-tuning on grounding-specific datasets (Lin et al., 2024; Gou et al., 2025; Hong et al., 2024). SeeClick (Cheng et al., 2024) formulates GUI grounding as a standalone pre-training objective and demonstrates that task-specific training improves agent performance. Subsequent work extends this direction by introducing more diverse instruction types and cross-platform training data (Wu et al., 2025b; Gou et al., 2025; Yang et al., 2025; Xue et al., 2026; Wang et al., 2025a). More recent studies apply reinforcement learning to GUI grounding (Luo et al., 2025a; Yuan et al., 2025; Gu et al., 2025; Chen et al., 2026a). These methods optimize task-level metrics and learn from interaction feedback. GUI-Cursor (Zhao et al., 2026) models grounding as an iterative cursor movement process, where predictions are refined through visual feedback. Other approaches improve grounding through multi-perspective instruction reasoning (Chen et al., 2026b), stabilized reward optimization (Kang et al., 2026), and joint planning with grounding in agent frameworks (Yang et al., 2026). Despite strong performance, both supervised and reinforcement learning approaches require substantial training data and computational resources.

A.2 INFERENCE-TIME ENHANCEMENT FOR GUI GROUNDING

Inference-time scaling improves performance without additional training (Snell et al., 2025). In GUI grounding, one line of work focuses on zoom-in-based inference. These methods iteratively crop and zoom into candidate regions, increasing resolution and reducing the search space (Jiang et al., 2025; Li et al., 2025a; Luo et al., 2025b; Lei et al., 2026; Nguyen, 2024). For example, ZoomClick (Jiang et al., 2025) identifies the essential characteristics of zooming and enhances target-focused resolution through a repeated crop-and-zoom strategy. RegionFocus (Luo et al., 2025b) applies dynamic zoom-in stragtegy to reduce background interference, thus enhancing grounding accuracy. Another line improves grounding accuracy by aggregating predictions across multiple runs (Zhang et al., 2026; Tang et al., 2026a; Wu et al., 2025a; Du et al., 2026; Lee et al., 2025). For instance, MVP (Zhang et al., 2026) aggregates predictions from multiple cropped views to reduce variance. DiMo-GUI (Wu et al., 2025a) reasons over textual and iconic elements separately and selects the best candidate based on global analysis. UI-Zoomer (Tang et al., 2026a) samples multiple candidate predictions and uses consensus voting together with adaptive cropping to achieve more robust grounding. Despite their differences, these approaches share a common limitation. They do not modify the internal forward process of the model. Instead, they vary the input through iterative zoom-and-crop operations or aggregate outputs from multiple inferences. As a result, each forward pass still contains a single prefill stage, and target selection must be completed in one attempt.

B PSEUDOCODE OF RE-PREFILL

Algorithm 1 gives the complete procedure of Re-Prefill described in Section 3. Given an input sequence $\mathbf{x} = [\mathbf{S}; \mathbf{V}; \mathbf{T}]$, the algorithm performs an initial prefill, selects key visual tokens through

Algorithm 1 Re-Prefill Inference

Require: Input sequence $\mathbf{x} = [\mathbf{S}; \mathbf{V}; \mathbf{T}]$; quantile threshold ρ; ratio threshold γ; continuity layers L_c
Ensure: Predicted coordinates (x, y)

Step 1: Initial prefill and attention extraction
$\big(\mathbf{h}_q, \mathcal{C}_1, \{\mathbf{a}^{(l)}\}_{l=1}^{L}, [\tilde{\mathbf{S}}; \tilde{\mathbf{V}}; \tilde{\mathbf{T}}]\big) \leftarrow f_{\text{prefill}}(\mathbf{x})$

Step 2: Key visual token selection
$\tau \leftarrow \text{quantile}\big(\{\mathbf{a}^{(l)}\}_{l=1}^{L},\ \rho\big)$
for each visual token index $i = 1, \ldots, N_v$ **do**
  $r(i) \leftarrow \frac{1}{L} \sum_{l=1}^{L} \mathbb{1}[\, a^{(l)}(i) > \tau \,]$
end for
$\tilde{\mathbf{V}}^* \leftarrow \{\, \tilde{v}_i \mid r(i) \geq \gamma \,\}$

Step 3: Layer-wise second prefill
for decoder layer $l = 1, \ldots, L$ **do**
  if $l \leq L_c$ **then**
    prefix $\leftarrow [\, \tilde{\mathbf{S}};\ \tilde{\mathbf{V}};\ \tilde{\mathbf{T}} \,]$
  else
    prefix $\leftarrow [\, \tilde{\mathbf{V}}^*;\ \tilde{\mathbf{T}} \,]$
  end if
  apply decoder layer l to $[\, \text{prefix};\ \mathbf{S};\ \mathbf{V};\ \mathbf{T} \,]$
end for
Extract enriched visual representations $\hat{\mathbf{V}}$ at the second set of visual-token positions

Step 4: Decoding context composition and generation
$\mathcal{C}^* \leftarrow [\, \tilde{\mathbf{S}};\ \tilde{\mathbf{V}};\ \tilde{\mathbf{T}};\ \hat{\mathbf{V}} \,]$
$(x, y) \leftarrow f_{\text{decode}}(\mathcal{C}^*)$
return (x, y)

cross-layer consistency, executes a layer-wise second prefill with hyperparameter L_c, and composes the decoding context $\mathcal{C}^*$.

C Additional Evaluation Results

C.1 Complementarity with Existing Training-Free Methods

Re-Prefill operates on a different dimension from existing training-free methods. While methods such as MVP (Zhang et al., 2026), ZoomClick (Jiang et al., 2025), and UI-Zoomer (Tang et al., 2026a) refine the visual input or aggregate predictions through multiple views or adaptive zooming, Re-Prefill targets the internal inference process by revisiting target selection during prefill. To ensure a fair comparison, we reproduce MVP, ZoomClick, and UI-Zoomer under the same experimental setting, using Qwen3-VL-8B-Instruct as the base model and ScreenSpot-Pro as the evaluation benchmark.

Table 7: Complementarity of Re-Prefill with existing training-free methods on ScreenSpot-Pro using Qwen3-VL-8B-Instruct. All baselines are reproduced under the same experimental setting.

Method	Accuracy (%) ↑
MVP (Zhang et al., 2026)	65.7
MVP + Re-Prefill	**69.5**$_{+3.8}$
ZoomClick (Jiang et al., 2025)	66.1
ZoomClick + Re-Prefill	**69.8**$_{+3.7}$
UI-Zoomer (Tang et al., 2026a)	61.4
UI-Zoomer + Re-Prefill	**63.7**$_{+2.3}$

As shown in Table 7, Re-Prefill consistently improves all baselines. It improves MVP from 65.7% to 69.5% $(+3.8)$, ZoomClick from 66.1% to 69.8% $(+3.7)$, and UI-Zoomer from 61.4% to 63.7% $(+2.3)$. These results show that Re-Prefill consistently improves existing input-space refinement and multi-view inference methods, demonstrating its complementarity with existing inference-time enhancement strategies. Importantly, the simplicity of Re-Prefill is grounded in a systematic empirical analysis rather than heuristic choices. By examining attention dynamics across inference stages, model architectures, and benchmarks, we

Table 8: Results on OSWorld-G (refined annotation). Methods marked with † are reproduced using the zoom-in strategy adopted in this work.

Model	Text Match.	Element Rec.	Layout Under.	Fine-grained Manip.	Refusal	Overall
UI-TARS-1.5-7B (ByteDance Seed Team, 2025)	52.6	75.4	72.4	66.7	-	64.2
GTA1-32B (Yang et al., 2026)	63.2	83.6	84.4	70.5	-	72.2
MAI-UI-8B (Zhou et al., 2025)	79.3	78.8	84.2	59.7	-	72.9
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)	73.1	69.4	71.0	74.8	42.5	69.3
Training-free methods						
MVP-8B (Zhang et al., 2026)	82.4	76.7	79.5	61.7	-	72.7
MVP-32B (Zhang et al., 2026)	81.6	76.4	77.5	62.4	-	72.0
Qwen3-VL-8B-Instruct (Bai et al., 2025)†	78.9	78.2	80.2	57.7	-	71.9
+ Re-Prefill	81.2	79.1	82.2	61.7	-	73.8$_{+1.9}$
MAI-UI-8B (Zhou et al., 2025)†	79.7	80.0	81.8	58.4	-	73.2
+ Re-Prefill	81.2	79.7	81.8	59.1	-	73.5$_{+0.3}$
GUI-Owl-1.5-8B-Instruct (Xu et al., 2026)†	81.6	79.4	82.6	65.1	-	74.6
+ Re-Prefill	81.6	79.7	83.4	65.8	-	75.0$_{+0.4}$
Qwen3-VL-32B-Instruct (Bai et al., 2025)†	82.4	85.2	87.0	62.4	-	77.3
+ Re-Prefill	83.5	84.2	85.8	65.8	-	77.5$_{+0.2}$

identify the prefill stage as a key bottleneck in GUI grounding. This finding motivates a lightweight second-prefill design that directly targets the identified bottleneck without modifying model parameters or requiring additional training. The consistent gains when combined with existing methods further support prefill refinement as a distinct direction for improving GUI grounding.

C.2 Results on OSWorld-G-Refined

This appendix presents results on OSWorld-G with the refined annotation. As shown in Table 8, Re-Prefill improves all base models under the refined annotation, with gains of $+1.9$ on Qwen3-VL-8B-Instruct, $+0.3$ on MAI-UI-8B, $+0.4$ on GUI-Owl-1.5-8B-Instruct, and $+0.2$ on Qwen3-VL-32B-Instruct. Compared with MVP using the same base models, Re-Prefill reaches 73.8 versus 72.7 for MVP-8B at the 8B scale and 77.5 versus 72.0 for MVP-32B at the 32B scale. These results demonstrate that the effectiveness of Re-Prefill remains consistent under refined annotations.

D Analysis of Prefill Dynamics

D.1 Qualitative Analysis

Figure 6 presents qualitative visualizations of query-position attention over visual tokens. For each example, we compare the baseline transition from prefill to the first decoding step with the corresponding transition under Re-Prefill, where a second prefill is applied before decoding. All results are produced using Qwen3-VL-8B-Instruct on ScreenSpot-Pro. Ground-truth targets are marked with blue rectangles, and predicted coordinates are shown as orange hollow circles for visibility.

Prefill vs. Re-Prefill. Under the baseline, attention at the prefill stage is distributed across multiple candidate regions, consistent with the high variance observed in Figure 1(b). While these regions often include the ground-truth target, distractors with similar appearance or semantics are also activated, making disambiguation difficult in a single forward pass. With Re-Prefill, attention becomes more concentrated on the correct region, while responses on distractor regions are noticeably suppressed. As a result, the effective candidate set is reduced, leading to more reliable target selection before decoding. This supports the design of Re-Prefill, where key visual tokens extracted from the first prefill guide a second prefill to refine attention within candidate regions and improve target selection.

The first decoding step. At the first decoding step, attention contracts sharply to a localized region in both settings, consistent with the drop in spatial variance from $t = 0$ to $t = 1$ in Figure 1(b). The predicted coordinate is placed within this region, indicating that target selection has already been determined before decoding. With Re-Prefill, this localized region is more likely to align with the ground-truth target, whereas the baseline may converge to a distractor region. This demonstrates that errors introduced during prefill persist into decoding, while improvements from Re-Prefill are directly propagated to coordinate prediction, consistent with the prefill bottleneck identified in Section 2.2.



Figure 6: **Query-position attention heatmaps across stages on ScreenSpot-Pro.** Each row shows one example. Columns 1–2 present the baseline transition from prefill to the first decoding step, while Columns 3–4 show the corresponding transition with Re-Prefill. The blue rectangle marks the ground-truth target, and the orange circle indicates the predicted coordinate. Compared to the baseline, Re-Prefill focuses attention on the correct region during prefill, suppresses distractors, and improves localization for subsequent decoding.

D.2 Analysis of Cross-Model and Cross-Benchmark Attention Dynamics

This appendix extends the analysis in Section 2.2 to multiple models and benchmarks. Experiments are conducted on two VLMs, Qwen3-VL-8B-Instruct and GUI-Owl-1.5-8B-Instruct, across three datasets, ScreenSpot-Pro, OSWorld-G, and MMBench-GUI. Figure 7 summarizes the results. The first two rows correspond to Qwen3-VL-8B-Instruct, and the last two rows correspond to GUI-Owl-1.5-8B-Instruct. The columns correspond to ScreenSpot-Pro, OSWorld-G, and MMBench-GUI. For each model, the first row shows spatial variance across generation steps, and the second row shows attention-centroid deviation from the ground truth for correct and incorrect predictions.

Consistent patterns are observed across all models and datasets. Spatial variance is high at the prefill stage ($t = 0$), drops sharply after the first decoding step ($t = 1$), and remains low thereafter, indicating rapid concentration of attention to a single region. In parallel, incorrect predictions are already misaligned with the ground truth at $t = 0$ and remain so during decoding, whereas correct predictions stay well aligned. These results show that the two-stage inference behavior identified in Section 2.2 holds across model architectures and benchmark domains, indicating that the prefill-stage bottleneck is a general property of VLM-based GUI grounding rather than an artifact of a specific model or dataset.

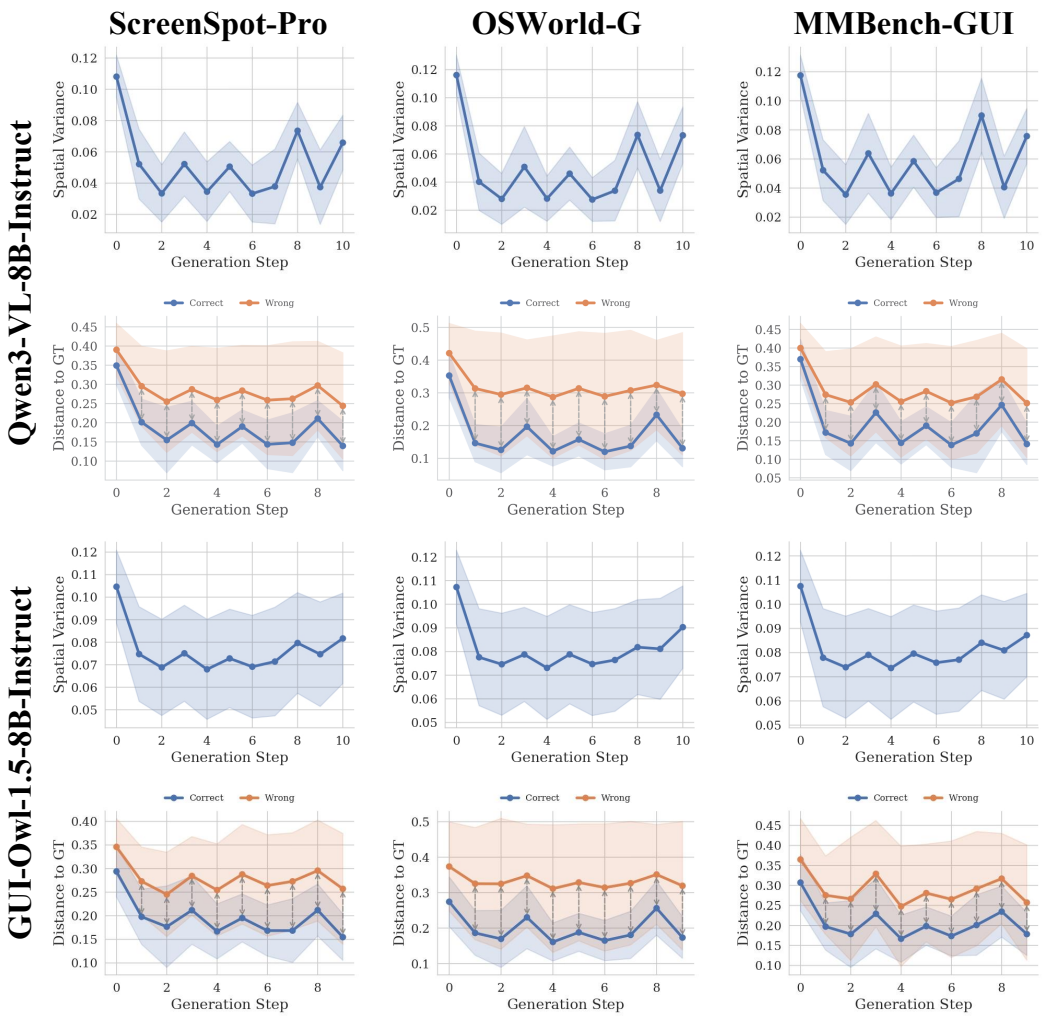


Figure 7: **Spatial variance and prefill-stage error analysis across models and benchmarks.** Rows 1–2 show results for Qwen3-VL-8B-Instruct, and Rows 3–4 for GUI-Owl-1.5-8B-Instruct. For each model, the first row shows spatial variance across generation steps, and the second row shows attention-centroid deviation for correct and incorrect predictions. Across all settings, attention is dispersed at prefill and rapidly contracts after the first decoding step, while errors introduced at prefill persist throughout decoding.

D.3 Analysis of Error Correction in Re-Prefill

Re-Prefill relies on the first-prefill attention to identify candidate visual tokens, but this does not necessarily amplify an incorrect initial prediction. During the first prefill, attention is typically distributed over multiple plausible UI regions rather than concentrated exclusively on a single target, and the ground-truth region often retains non-trivial attention even when the final prediction is incorrect. The second prefill therefore re-evaluates these candidate regions under an explicit visual prior, allowing the grounding state to be refined before coordinate decoding.

We further examine this behavior on samples that are incorrectly grounded by the baseline. For these samples, we measure the normalized Distance-to-GT defined in Eq. (2) across decoding steps. As shown in Table 9, Re-Prefill consistently reduces the distance to the ground-truth

Table 9: Distance-to-GT analysis on samples incorrectly grounded by the baseline. Lower is better.

Method	Step 1	Step 3	Step 5	Step 7	Avg.
Baseline	0.24	0.24	0.24	0.26	0.25
Re-Prefill	**0.20**	**0.18**	**0.19**	**0.23**	**0.20**

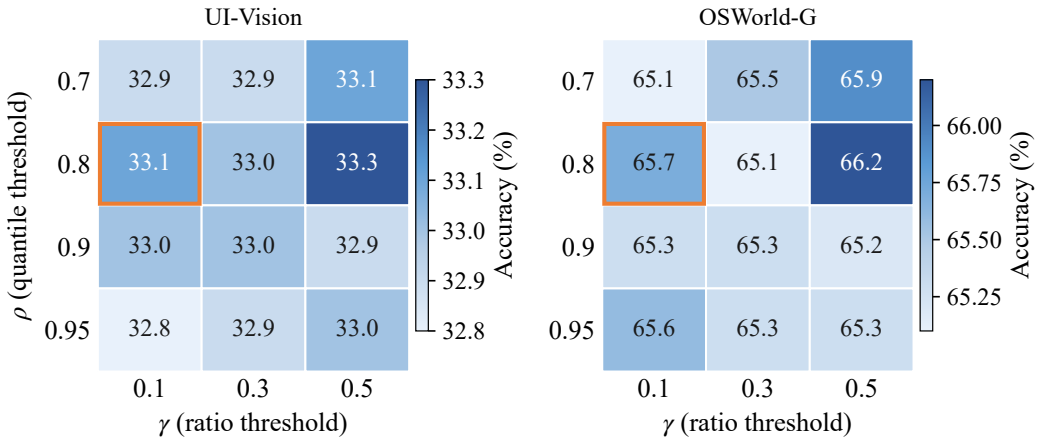


Figure 8: Additional sensitivity analysis of ρ and γ on UI-Vision and OSWorld-G with Qwen3-VL-8B-Instruct. The orange boxes indicate the default configuration ($\rho = 0.8$, $\gamma = 0.1$). Performance remains stable across all tested settings on both benchmarks.

target across decoding steps, with the improvement already evident at the first decoding step. This result provides evidence that the second prefill can revise an initially misaligned grounding state before decoding, rather than simply reinforcing the first-prefill attention pattern.

E Analysis of Robustness and Sensitivity

E.1 Robustness to Attention Noise

Re-Prefill does not rely on perfectly calibrated attention maps, as its key-token selection reduces noise through head averaging and cross-layer consistency. Moreover, the quantile-based threshold depends on relative attention rankings rather than absolute magnitudes, reducing sensitivity to calibration differences across layers and models.

To further evaluate this robustness, we perturb the attention maps with relative Gaussian noise:

$$A' = A + \alpha \sigma_A \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 1), \tag{6}$$

where σ_A denotes the standard deviation of the original attention map and α controls the perturbation strength. Experiments are conducted on ScreenSpot-Pro with Qwen3-VL-8B-Instruct.

Table 10: Robustness of Re-Prefill to attention perturbations on ScreenSpot-Pro with Qwen3-VL-8B-Instruct.

Noise Strength α	Accuracy (%) ↑
0	70.1
0.1	69.6 (−0.5)
0.5	69.5 (−0.6)
1.0	69.0 (−1.1)

As shown in Table 10, Re-Prefill remains stable under substantial attention perturbations. The accuracy decreases by only 0.5 points at $\alpha = 0.1$, 0.6 points at $\alpha = 0.5$, and 1.1 points even at $\alpha = 1.0$. These results indicate that Re-Prefill does not require precisely calibrated attention maps, consistent with the robustness induced by its head-averaged and cross-layer-consistent selection mechanism.

E.2 Sensitivity to Hyperparameters ρ and γ

We extend the analysis of ρ and γ in Section 4.3 to UI-Vision and OSWorld-G, which contain more challenging instructions. As shown in Figure 8, performance remains stable across all tested configurations, ranging from 32.8 to 33.3 on UI-Vision and from 65.1 to 66.2 on OSWorld-G. All configurations also outperform the corresponding baselines of 32.1 and 63.0, respectively, further demonstrating that Re-Prefill is not sensitive to ρ and γ under more complex instruction scenarios.

Table 11: Effect of the continuity layer L_c on ScreenSpot-Pro with Qwen3-VL-32B-Instruct.

L_c	Baseline	4	6	10	20	30
Accuracy (%)	74.3	**77.0**	76.8	76.9	76.0	76.2

Table 12: TTFT and end-to-end latency on ScreenSpot-Pro with Qwen3-VL-8B-Instruct.

Method	TTFT (ms) ↓	End-to-End Latency (ms) ↓
Baseline	865.3 ± 0.6	1469.6 ± 149.8
Re-Prefill	1790.6 ± 8.9	2465.2 ± 40.1
Overhead	$+925.3$ ms ($\sim 2.1\times$)	$+995.6$ ms ($\sim 1.7\times$)

E.3 Sensitivity to Continuity Layer Across Model Scales

We evaluate the continuity layer L_c on Qwen3-VL-32B-Instruct ($L = 64$) to examine its transferability across model scales. As shown in Table 11, Re-Prefill consistently outperforms the 74.3% baseline across $L_c \in \{4, 6, 10, 20, 30\}$, with accuracy ranging from 76.0% to 77.0%. The best performance is achieved at $L_c = 4$, while nearby shallow-to-middle settings remain comparably effective. Together with the 36-layer model results in Figure 5, these results suggest that L_c does not require precise tuning to a single layer, although its optimal value may shift with model depth.

F Analysis of Computational Efficiency

We further analyze the latency overhead introduced by Re-Prefill. To isolate the computational cost of the additional prefill stage, we first measure the time-to-first-token (TTFT) and end-to-end latency *without* the zoom-in strategy used in the main evaluation. As shown in Table 12, Re-Prefill increases TTFT from 865.3 ms to 1790.6 ms and end-to-end latency from 1469.6 ms to 2465.2 ms, corresponding to an overhead of 925.3 ms ($\sim 2.1\times$) and 995.6 ms ($\sim 1.7\times$), respectively. The additional cost mainly comes from the second prefill stage, which reprocesses the input before coordinate decoding.

We also compare the computational efficiency of Re-Prefill with existing training-free methods, including MVP (Zhang et al., 2026) and ZoomClick (Jiang et al., 2025), under identical base models. Experiments are conducted on Qwen3-VL-8B-Instruct and Qwen3-VL-32B-Instruct across ScreenSpot-Pro, OSWorld-G, and MMBench-GUI. Unlike Table 12, this comparison follows the complete evaluation pipeline used in our main experiments, including the shared zoom-in strategy on ScreenSpot-Pro, OSWorld-G, and MMBench-GUI. This setting enables a fair comparison with existing methods that also rely on zoom-in-based inference. We report the average inference time per sample together with grounding accuracy. To ensure fairness, MVP and ZoomClick are reproduced under the same experimental environment and settings. As shown in Table 13, Re-Prefill achieves higher accuracy while maintaining competitive runtime. On 8B models, Re-Prefill is slightly faster than or comparable to existing methods. On 32B models, Re-Prefill operates under a comparable computational budget to other training-free methods while achieving higher grounding performance. Overall, Re-Prefill provides a favorable efficiency–performance trade-off. It improves grounding accuracy with competitive efficiency, particularly on smaller models.

G Limitations

While Re-Prefill demonstrates consistent improvements across models and benchmarks, several limitations remain. As shown in the experimental results, the improvements from Re-Prefill are more pronounced in visually complex settings with high resolution and dense UI elements, where accurate target selection is the main challenge. In contrast, gains are smaller in instruction-heavy settings, where performance is more constrained by the ability to interpret complex or abstract instructions. This suggests that while Re-Prefill improves target selection during the prefill stage,

Table 13: Comparison of computational efficiency among training-free methods. Inference time is reported as the average per-sample latency in seconds. Methods marked with † are reproduced under the same experimental settings in this work.

Methods	ScreenSpot-Pro		OSWorld-G		MMBench-GUI	
	Time (s)↓	Acc. (%)↑	Time (s)↓	Acc. (%)↑	Time (s)↓	Acc. (%)↑
Qwen3-VL-8B-Instruct						
MVP†	9.0	65.7	4.2	59.3	5.4	84.4
ZoomClick†	11.2	66.1	**4.1**	56.7	7.0	84.4
Re-Prefill	**8.0**	**70.1**	**4.1**	**65.7**	**4.8**	**86.4**
Qwen3-VL-32B-Instruct						
MVP†	**11.8**	73.9	6.4	65.3	8.4	87.8
ZoomClick†	12.4	71.5	**6.2**	58.2	**7.6**	86.4
Re-Prefill	12.2	**76.8**	7.5	**70.1**	7.8	**89.5**

it does not directly enhance instruction understanding. Extending the approach to better support complex instruction comprehension remains an important direction for future work.