GUI-AIMA: Aligning Intrinsic Multimodal Attention with a Context Anchor for GUI Grounding

Shijie Zhou[1⋆] Viet Dac Lai[2] Hao Tan[2] Jihyung Kil[2] Wanrong Zhu[2]
Changyou Chen[1] Ruiyi Zhang[2⋆⋆✉]

[1] University at Buffalo, USA
[2] Adobe Research, USA
{shijiezh, changyou}@buffalo.edu, ryzhang.cs@gmail.com

Abstract. Graphical user interface (GUI) grounding is a key capability for computer-use agents, mapping natural-language instructions to actionable regions on the screen. Existing Multimodal Large Language Model (MLLM) approaches typically formulate GUI grounding as a text-based coordinate generation task. However, directly generating precise coordinates from visual inputs is challenging and often data-intensive. A more intuitive strategy is to first *identify instruction-relevant visual patches and then determine the exact click location within them*. Motivated by recent observations that general MLLMs exhibit native grounding ability embedded in their attention maps, we propose GUI-AIMA, an attention-based and coordinate-free supervised fine-tuning framework for efficient GUI grounding. GUI-AIMA aligns the intrinsic multimodal attention of MLLMs with patch-wise grounding signals. These signals are calculated adaptively for diverse user instructions by multi-head aggregation on simplified query-visual attention matrices. Besides, its coordinate-free manner can easily integrate a plug-and-play zoom-in stage. GUI-AIMA-3B was trained with only 509k samples (∼101k screenshots), demonstrating exceptional data efficiency and verifying that light training can trigger the native grounding capability of MLLMs. It achieves state-of-the-art performance among 3B models, attaining an average accuracy of **61.5%** on ScreenSpot-Pro, **92.1%** on ScreenSpot-v2, **68.1%** on OSWorld-G, **79.1%** on MMBench-GUI-L2 and **60.0%** on UI-Vision. Project page: https://github.com/sjz5202/GUI-AIMA.

Keywords: GUI Agents · Visual Grounding · Multimodal Attention

1 Introduction

Graphical User Interface (GUI) agents [6, 16, 22] have emerged as pivotal tools in automating interactions with digital devices, spanning from mobile appli-

⋆ Work done at UB through University Collaborations.
⋆⋆ Project lead; work done while at Adobe Research; ✉Corresponding author.

cations [32, 33, 39, 51] to desktop software [10, 30, 31, 46, 54, 56]. GUI grounding is a core capability of GUI agents, requiring the model to map natural-language instructions to specific interface elements on the screen, such as buttons, text fields, and icons [50]. Early GUI grounding methods often rely on structured interface metadata, such as accessibility trees for mobile applications [55]. Although such representations provide explicit descriptions of interface elements, they are limited in accessibility, often overly verbose, and may miss critical visual cues such as layout and icons. These limitations have motivated a shift from metadata-based LLM grounding to screenshot-based multimodal large language models (MLLMs), which can directly reason over visual interfaces.

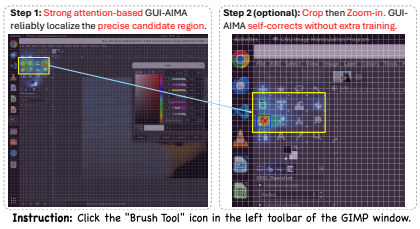

Fig. 1: An example of GUI-AIMA with two-step GUI grounding for high resolution screenshots.

However, most existing MLLM-based GUI grounding methods [14, 23, 44, 48] formulate the task in a coordinate-based manner, i.e., by directly predicting click coordinates as text tokens. This is an indirect formulation, since the model must compress fine-grained visual localization into textual coordinate generation. At the same time, recent evidence suggests that once GUI grounding is performed with MLLMs, useful query-visual grounding signals already emerge in their multimodal attention maps [47]. This makes a more natural alternative possible: instead of generating coordinate tokens, one can perform coordinate-free GUI grounding by directly exploiting the MLLM's native attention as patch-level grounding cues, first identifying the relevant visual patches and then determining the specific click center within the selected region.

Recent efforts have begun to explore this direction. TAG [47] leverages intrinsic query-visual attention in MLLMs, but aggregates attention maps from all query tokens in a vanilla manner, which is simple but often inaccurate. GUI-Actor [43] also avoids direct coordinate generation, but introduces additional grounding modules instead of utilizing native attention and requires an extra adaptation stage. These limitations leave open an important question: can we directly specialize the *intrinsic* multi-head attention of MLLMs for GUI grounding, without extra modules and without relying on impractical token-wise aggregation?

In this work, we propose GUI-AIMA, a coordinate-free GUI grounding framework that directly supervises the MLLM's multi-head self-attention (MHSA) [36] with patch-wise grounding signals. GUI-AIMA enables simple and fine-grained aggregation of patch-level attention vectors across all query tokens and all attention heads through two designs. First, we append a learnable `<ANCHOR>` token after visual and query tokens, and use its attention over visual patches as a surrogate aggregator of query-visual interactions, thereby avoiding explicit token-wise aggregation. Second, we introduce a head-weighting mechanism based on *visual-*

sink query tokens: a small subset of query-side tokens selected from hidden-state similarity that reflects a strong query-visual connection. These tokens are used as proxies to identify attention heads that are more likely to encode grounding-relevant cross-modal correspondences, enabling more precise multi-head aggregation while minimally perturbing less grounding-relevant heads.

Built on this coordinate-free attention formulation, GUI-AIMA naturally supports practical improvements in both training and inference. We design an overlap- and distance-aware patch labeling that better matches click behavior. Moreover, because GUI-AIMA produces grounding from patch-level attention, it naturally enables a two-step zoom-in inference procedure for high-resolution screenshots: the model first localizes a coarse region, and then refines the prediction on a cropped view without additional training, as shown in Fig. 1. This helps correct offset errors that commonly arise on dense, high-resolution interfaces.

GUI-AIMA is both effective and data-efficient. Under the ablation training setting, it improves ScreenSpot-Pro accuracy by 5.88% over vanilla attention grounding (43.39% over 37.51%). With only one-stage fine-tuning on 101k screenshots without extra modules, GUI-AIMA-3B outperforms similarly sized models and remains competitive with substantially larger MLLM-based GUI grounding methods.

Overall, contributions of GUI-AIMA are as follows: **(i)** We propose GUI-AIMA, an attention-based, coordinate-free framework that directly leverages the intrinsic attention of MLLMs for GUI grounding, without introducing extra grounding modules; **(ii)** We develop a simple attention aggregation strategy that replaces impractical vanilla token-wise aggregation and a head-weighting mechanism that emphasizes grounding-relevant attention heads; **(iii)** We design a patch-wise labeling that better matches click behavior, together with a two-step zoom-in inference procedure for improved high-resolution localization; **(iv)** With only one-stage fine-tuning on 101k screenshots, GUI-AIMA-3B achieves strong data efficiency, outperforming similarly sized models and remaining competitive with substantially larger GUI grounding systems.

2 Related Work

Coordinate-based GUI grounding: The core challenge for GUI agents [40] is grounding: aligning user instructions with the correct actionable elements in the screenshot, due to the semantic inconsistency of layouts and UI elements across diverse GUI environments [24]. In early attempts, along with the screenshots, extra structured inputs, such as HTML for U-Ground [14], or extractions from visual parsing modules, such as element extractions from OmniParser [38, 52] and generated captions as contexts in Aria-UI [50], are fed into MLLMs as the supplementary inputs for a better interface understanding and more precise localization. Later works, such as AGUVIS [48], SeeClick [6] and OS-Atlas [44], explore the GUI grounding leveraging MLLMs [2, 5] with screenshot-only inputs, enabling the end-to-end localization with improved scalability across diverse GUI environments. The grounding approach in these methods is to gener-

ate coordinate-based click centers or bounding boxes as text, which is indirect alignment of visual grounding requiring additional efforts, such as scaled GUI corpora [3, 31] and OCR pretraining [16] for connecting coordinates with UI elements. Recent endeavors manage to address this gap from the data-intensive manner of the coordinate-based GUI grounding methods via incorporating GUI-specific grounding reward signals into RL-training [24–26, 35] and performing autonomous GUI exploration [12, 43].

Coordinate-free GUI grounding: Recent works have begun to replace traditional coordinate-based grounding text with patch-level attention map predictions, where patches belonging to the correct region receive higher weight. TAG [47] first leveraged the multi-head self-attention between GUI instructions and visual tokens in MLLMs for tuning-free GUI grounding, showcasing the intrinsic potential of MLLMs' attention patterns for GUI tasks. However, the generalization of TAG for novel interfaces is restricted by the backbone and by the heuristics used for selecting text tokens and attention heads. SE-GUI [53] retains a coordinate-based prediction manner but employs self-attention to iteratively filter noisy training samples during training. Aside from these attention-based improvements, GUI-Actor [43] introduces an embedding-based grounding head that derives attention between input visual patch embeddings and the final hidden state of a special `<ACTOR>` token. Compared with GUI-AIMA, GUI-Actor adds an additional module to connect cross-layer visual and `<ACTOR>` embeddings, requiring an extra warm-up training phase and resulting in much lower training efficiency.

3 Methodology

GUI-AIMA formulates GUI grounding as the attention supervision: it uses the MLLM's intrinsic multi-head attention maps across layers as grounding indicators, instead of coordinate tokens, and directly supervises these attention maps without introducing additional grounding modules. As illustrated in Sec. 3.1, within each attention head of the MLLM, the **patch-level attention vector** from a single text token (of the user query) to all visual tokens indicates the relevance of each visual patch to the target grounding region, naturally serving as grounding cues. GUI-AIMA fully exploits the attention vectors from the entire text sequence and all attention heads in a simple but precise way: adding a special representative token for query-level aggregation (Sec. 3.1) and adaptive head-wise reweighting based on multimodal importance (Sec. 3.2).

3.1 Intrinsic Attention for GUI Grounding with an Anchor Token

Preliminaries on Multi-Head Self-Attention (MHSA). For each layer $1 \leq l \leq L$ of MLLMs, each head $1 \leq h \leq H$ computes the attention map $\mathbf{A}^{l,h}$ from preceding layer's embeddings $\mathbf{H}^{l-1} \in \mathbb{R}^d$ via $\mathbf{A}^{l,h} = \text{softmax}\big(\mathbf{Q}^{l,h}\mathbf{K}^{l,h^\top}/\sqrt{d_h} + \mathbf{M}\big)$, where $\mathbf{Q}^{l,h}$ and $\mathbf{K}^{l,h}$ are linear projections of $\mathbf{H}^{l-1}$ with dimension $d_h =$

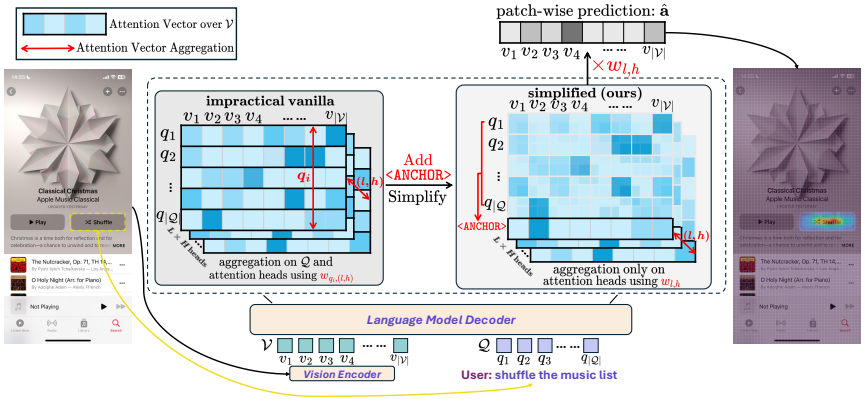

Fig. 2: With user query $\mathcal{Q}$, screenshot patches $\mathcal{V}$ and multi-head attentions $\{\mathbf{A}^{l,h}\}_{l\in[L],h\in[H]}$ from the MLLM, the vanilla attention grounding needs additional aggregation between all query tokens' grounding vectors. In our proposed simplified version, a special `<ANCHOR>` token can learn to implicitly aggregate all query tokens. Then we aggregate grounding vectors of `<ANCHOR>` token across layers and heads with carefully designed weights to produce the patch-wise predictions.

d/H, and $\mathbf{M}$ is the causal mask with $\mathbf{M}_{ij} = 0$ for $i \geq j$ and $\mathbf{M}_{ij} = -\infty$ for $i < j$.

Intrinsic Grounding Signals in Attention Maps $\mathbf{A}^{l,h}$. Let $\mathcal{V} = [v_1, \ldots, v_{|\mathcal{V}|}]$ and $\mathcal{Q} = [q_1, \ldots, q_{|\mathcal{Q}|}]$ denote the visual patch-token sequence and the user-query text token sequence, respectively. Given the attention map $\mathbf{A}^{l,h}$, the **patch-level attention vector** from query token q_i to all visual patches $\mathcal{V}$ is denoted by $\mathbf{A}^{l,h}_{q_i,\mathcal{V}}$. Its entry $\mathbf{A}^{l,h}_{q_i,v_j}$ measures how strongly patch v_j is associated with token q_i, and patches with larger attention values are more likely to contain regions relevant to q_i. Therefore, aggregating $\{\mathbf{A}^{l,h}_{q_i,\mathcal{V}}\}_{l,h,i}$ across layers, heads, and query tokens with weight $w_{q_i,(l,h)}$ produces intrinsic grounding cues over all visual patches:

$$\hat{\mathbf{a}} = \frac{1}{L \cdot H \cdot |\mathcal{Q}|} \sum_{l,h,i} w_{q_i,(l,h)}\, \mathbf{A}^{l,h}_{q_i,\mathcal{V}} \quad \in \mathbb{R}^{|\mathcal{V}|}, \tag{1}$$

where $w_{q_i,(l,h)} \geq 0$ and $\sum_{l,h,i} w_{q_i,(l,h)} = 1$. We refer to Eq. (1) as **Vanilla Attention Grounding** ("vanilla" in Fig. 2). However, effectively supervising $\hat{\mathbf{a}}$ under this formulation requires a carefully calibrated balance not only across attention heads, but also across $\mathbf{A}^{l,h}_{q_i,\mathcal{V}}$ from different query tokens, making $w_{q_i,(l,h)}$ impractical. Moreover, directly imposing supervision on the attention of query tokens may interfere with the pretrained language-vision behaviors [34], potentially impairing pre-existing knowledge of MLLM.

To address these issues, we introduce a special visual anchor token `<ANCHOR>` and append it *after* the GUI inputs, forming the sequence $[\mathcal{V}, \mathcal{Q}, \texttt{<ANCHOR>}]$. This placement is crucial under causal self-attention: the last token can attend

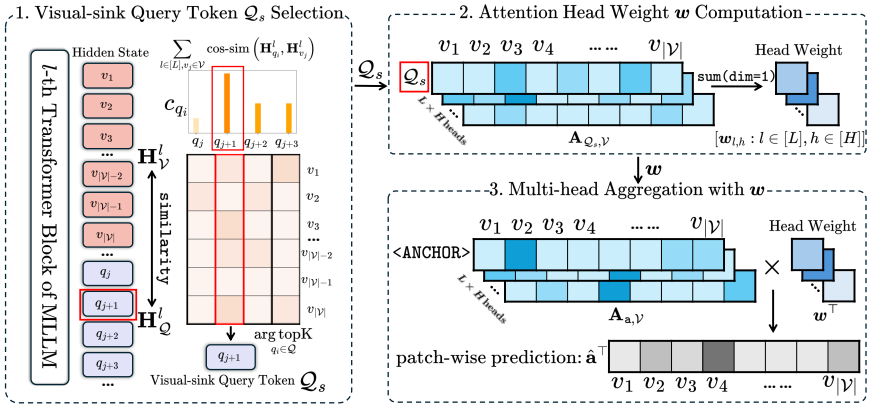

Fig. 3: Pipeline of GUI-AIMA for patch-wise prediction from the <ANCHOR> token with attention-head weighting: **(i)** GUI-AIMA first selects visual-sink query tokens $\mathcal{Q}_s$ (Eq. (5)) via computing hidden state similarities between query tokens and visual patches (Eq. (4)); **(ii)** it computes weights $\boldsymbol{w}$ of each attention head (Eq. (3)) based on selected $\mathcal{Q}_s$; **(iii)** GUI-AIMA aggregates the grounding vectors of <ANCHOR> token across layers and heads (Eq. (2)).

to *all* preceding visual and query tokens, enabling <ANCHOR> to summarize the entire instruction context in its attention distribution. As a result, token-wise weighting is avoided by letting <ANCHOR> serve as a representative query token for grounding, while disentangling GUI grounding supervision from the attentions of pre-existing query tokens.

Concretely, for each layer-head pair (l, h), we use the **anchored patch-level attention vector** $\mathbf{A}^{l,h}_{\mathtt{a},\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}|}$, i.e., the attention from <ANCHOR> to all visual tokens in $\mathcal{V}$, to learns an implicit aggregation of $\{\mathbf{A}^{l,h}_{q_i,\mathcal{V}}\}_{q_i \in \mathcal{Q}}$ over all query tokens. With these anchored attentions, we simplify Eq. (1) into a **multi-head aggregation** that only weights attention heads:

$$\hat{\mathbf{a}} = \frac{1}{L \cdot H} \sum_{l,h} w_{l,h}\, \mathbf{A}^{l,h}_{\mathtt{a},\mathcal{V}} \quad \in \mathbb{R}^{|\mathcal{V}|}, \tag{2}$$

where $w_{l,h} \geq 0$ and $\sum_{l,h} w_{l,h} = 1$, and we denote the head-weight vector as $\boldsymbol{w} = [\, w_{l,h} : l \in [L],\ h \in [H]\,] \in \mathbb{R}^{1 \times L \cdot H}$. This **Simplified Attention Grounding** is shown as "simplified" in Fig. 2.

3.2 Attention Head Weighting using Visual-sink Query Tokens

In Sec. 3.1, we introduce a special <ANCHOR> token to avoid token-wise aggregation over all query words. We now discuss how to determine the head weights $w_{l,h}$ for the multi-head aggregation in Eq. (2). Previous works [7, 20] have shown the functional diversity between attention heads across transformer blocks in LLMs.

For GUI grounding, we aim to emphasize heads that capture strong query-visual interactions, while down-weighting heads that mainly serve other functions. This selective weighting allows attention supervision to focus on grounding-relevant heads and to minimally perturb neutral heads, helping preserve pretrained capabilities.

An Attention-based View of Head Weighting. We measure the grounding relevance of each attention head by the amount of query-to-visual attention mass it carries: heads with stronger query-visual interactions are more likely to encode grounding-relevant correspondences. For a selected query-side subset $\mathcal{Q}' \subseteq \mathcal{Q} \cup \{\texttt{<ANCHOR>}\}$, we define the head score of attention head (l, h) as the cumulative attention mass from $\mathcal{Q}'$ to all visual tokens, with further normalization:

$$w_{l,h}(\mathcal{Q}') = \sum_{q \in \mathcal{Q}'} \sum_{v_j \in \mathcal{V}} \mathbf{A}^{l,h}_{q,v_j}, \quad w_{l,h} = \frac{\exp\left(w_{l,h}\right)}{\sum_{l'=1}^{L} \sum_{h'=1}^{H} \exp\left(w_{l',h'}\right)} \tag{3}$$

Under this view, different head-weighting strategies can be unified by varying only the query-side token subset used to compute the score. A straightforward choice is to use all query tokens, i.e., $\mathcal{Q}' = \mathcal{Q}$. However, this choice is imprecise, since query tokens contribute unevenly to grounding, and aggregating over all of them can introduce noisy attention mass unrelated to the actual target region. Another simple choice is to use only the anchor token, i.e., $\mathcal{Q}' = \{\texttt{<ANCHOR>}\}$, since <ANCHOR> serves as the representative token for grounding in Eq. (2). Although simpler, this strategy can be unreliable early in training, because <ANCHOR> is randomly initialized and may not yet serve as a stable context summarizer.

Visual-sink Query Token Selection. To estimate grounding-relevant head weights via Eq. (3), we first identify a query-side token subset that captures query-visual interaction in a head-agnostic manner: for a user query $\mathcal{Q}$, we estimate the visual affinity of each token q_i at layer l, denoted as $c^l_{q_i}$, using cosine similarity between its hidden state and all visual patch states (Step 1 in Fig. 3). And We denote the selected top-K tokens via $c^l_{q_i}$ by $\mathcal{Q}_s$:

$$c^l_{q_i} = \sum_{v_j \in \mathcal{V}} \texttt{Sim}(\mathbf{H}^l_{q_i}, \mathbf{H}^l_{v_j}). \tag{4}$$

A larger $c^l_{q_i}$ indicates that the representation of q_i is more strongly aligned with the global visual context at layer l.

We use hidden-state similarity instead of directly relying on attention values because grounding-relevant query-visual interactions are often concentrated in only a subset of attention heads, and may therefore not be uniformly observable from raw attention patterns alone [11, 28, 37]. In contrast, the similarity score computed in Eq. (4) provides a head-agnostic and more global representation-space signal, making it more stable for identifying tokens that consistently correlate with cross-modal interaction (see the supplementary material for analysis).

As shown in Fig. 4, the top-ranked token often shows a sink-like behavior: it repeatedly concentrates interaction with the visual context without necessarily

being the most semantically informative word. We therefore refer to these tokens as **visual-sink query tokens**, emphasizing their role as *functional proxies* for cross-modal interaction. In practice, we find that using a shared token set across all layers provides better stability and performance, instead of layer-wise $\mathcal{Q}_s^l$ for each layer as $\mathcal{Q}_s^l := \arg\operatorname{topK}_{q_i \in \mathcal{Q}}(c_{q_i}^l)$. Specifically, we adopt a **global uniform** selection that first aggregates $c_{q_i}^l$ across layers and then selects a shared visual-sink query token set $\mathcal{Q}_s$:

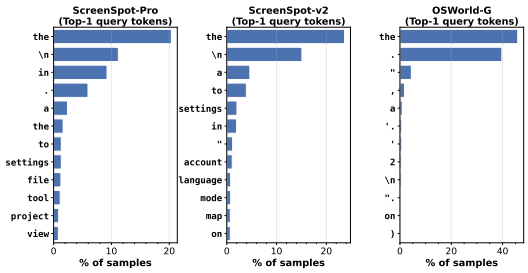

Fig. 4: Distribution of top-1 selected visual-sink query tokens under c_{q_i}.

$$\mathcal{Q}_s := \arg\operatorname*{topK}_{q_i \in \mathcal{Q}}(c_{q_i}), \quad c_{q_i} = \sum_{l=1}^{L} c_{q_i}^l. \tag{5}$$

With $\mathcal{Q}_s$, we compute the normalized head weights $w_{l,h}$ using Eq. (3). Intuitively, if a head assigns substantial query-to-visual attention mass through these selected tokens, its attention pattern is more consistent with the cross-modal alignment pattern found in representation space. We therefore assign such heads larger weights in Eq. (2) to produce the final patch-level prediction $\hat{\mathbf{a}}$.

3.3 Training and Inference Design

Coordinate-free Supervision with Patch-wise Labels. Since our attention grounding is defined over visual patches rather than pixel coordinates, we convert each ground-truth bounding box $gt^{\text{bbox}} = [x_1, y_1, x_2, y_2]$ into a patch-level label vector $\boldsymbol{p} \in \mathbb{R}^{|\mathcal{V}|}$, where $|\mathcal{V}|$ is the number of visual patches. A patch v_i is assigned a positive label only if it overlaps with gt^{bbox}.

To emphasize central patches while softly down-weighting partially overlapped boundary patches, we assign each positive patch a weight based on both its overlap with the ground-truth box and its distance to the box center, following GUI-G^2 [35]:

$$p_{v_i} = \mathbf{IoU}(v_i, gt^{\text{bbox}}) \cdot \mathcal{N}(\boldsymbol{\mu}_{v_i}; \boldsymbol{\mu}_{gt}, \boldsymbol{\Sigma}_{gt}), \tag{6}$$

where $\boldsymbol{\mu}_{v_i}$ and $\boldsymbol{\mu}_{gt}$ denote the centers of patch v_i and gt^{bbox}, respectively, and $\boldsymbol{\Sigma}_{gt} = \operatorname{diag}\big((\sigma_x^{gt})^2, (\sigma_y^{gt})^2\big)$ with $\sigma_x^{gt} = \alpha(x_2 - x_1)$ and $\sigma_y^{gt} = \alpha(y_2 - y_1)$.

The final label vector is normalized as $\boldsymbol{p} = \text{normalize}(\{p_{v_i}\}_{i=1}^{|\mathcal{V}|})$, which is both overlap-aware and center-biased. We then supervise the predicted patch distribution $\hat{\mathbf{a}}$ from Eq. (2) using the KL divergence:

$$\mathcal{L}_{\text{Attn}} = \mathbb{D}_{\text{KL}}(\boldsymbol{p} \,\|\, \text{normalize}(\hat{\mathbf{a}})).$$

Two-step Inference with Zoom-in 🔍. Under GPU memory constraints, high-resolution GUI screenshots are usually down-sampled, yielding fewer visual patch tokens for the MLLM. The resulting information loss and reduced

fine-grained spatial granularity inevitably harm grounding accuracy. GUI-AIMA provides flexible spatial granularity since its patch-wise grounding. We can easily perform a two-step inference by adding zoom-in **without extra training**: (1) feed the compressed high-resolution screenshot to predict the approximate location. We use the center of this predicted location to determine a specific area to focus on. (2) we re-run the inference on the newly cropped region, providing a much more accurate result. This two-step inference targets to mitigate failure cases on high-resolution screens where the model identifies the right region, but the center prediction is slightly offset from the ground-truth box. Detailed observations and analysis of the zoom-in strategy are provided in Sec. 4.4.

Efficient Attention Map Extraction. FlashAttention [9] avoids materializing full attention maps, while standard eager attention [36] can return them but is slower and more memory-intensive. In GUI-AIMA, we use FlashAttention for the regular forward pass and eager attention only for the grounding-relevant rows, i.e., the query tokens and `<ANCHOR>` over visual tokens. This reduces the extraction cost roughly from $O((|\mathcal{V}| + |\mathcal{Q}| + 1)^2)$ to $O((|\mathcal{Q}| + 1) \cdot |\mathcal{V}|)$. Since $|\mathcal{Q}| \ll |\mathcal{V}|$ in GUI inputs, the additional overhead is negligible, shown in Sec. 4.3.

4 Experiments

4.1 Implementation Details.

Training Setup. We use Qwen2.5-VL-3B-Instruct [2] as the MLLM backbone for GUI-AIMA and all ablation variants. Following GUI-Actor [43], we retain the next-token prediction loss for GUI-AIMA's grounding format. GUI-AIMA-3B is trained on 8 NVIDIA A100-80G GPUs with a global batch size of 64 and a learning rate of 5×10^{-6}. We set $\alpha = 0.8$ for the adaptive deviation control in Gaussian labeling. For visual-sink query token selection, we use the top-1 token under the global uniform criterion in Eq. (5). The patch grid resolution is 28×28 in Qwen2.5-VL base model. GUI-AIMA-3B is trained for one epoch with all parameters unfrozen, without extra modules or a warm-up stage.

Training Data and Inference Setup. To examine the scalability of GUI-AIMA with increased training data, we train two model variants: GUI-AIMA-3B-lite is trained on the full training sets of GUIAct [4], AndroidControl [21], and Wave-UI [18], together with 60k randomly sampled samples from UGround [14] and 60k from the GTA1 training set [49], resulting in 259k instructions and approximately 85k images in total. GUI-AIMA-3B is then further trained with an additional 250k instructions (509k in total) from GroundCUA [13] to better scale to desktop scenarios. For the 45k ablation training set for ablations in Sec. 4.3, we use the first 10k samples from GUIAct, AndroidControl, and Wave-UI, together with the first 15k samples from UGround. For the two-step inference in Sec. 3.3, we set the crop size to 616 pixels and use a 2× zoom-in ratio in the second pass. Implementation details of the baselines are provided in the supplementary material.

Benchmarks and Metrics. We evaluate GUI-AIMA on five GUI grounding benchmarks covering a wide range of grounding scenarios: ScreenSpot-Pro [19],

Table 1: Performance across various categories in **ScreenSpot-Pro**, reporting Text, Icon, and Average scores. "-" indicates unreported results in original papers. Methods with * are Qwen2.5-VL-based. 🔍 denotes two-step inference with zoom-in.

	Model	CAD		Dev		Creative		Scientific		Office		OS		Average		
		Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	**Avg.**
General	Claude Computer [17]	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
General	Qwen2.5-VL-3B [2]	9.1	7.3	22.1	1.4	26.8	2.1	38.2	7.3	33.9	15.1	10.3	1.1	23.6	3.8	16.1
General	Qwen2.5-VL-7B [2]	16.8	1.6	46.8	4.1	35.9	7.7	49.3	7.3	52.5	20.8	37.4	6.7	38.9	7.1	26.8
General	Qwen3-VL-8B [1]	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
General	Qwen3-VL-30B-A3B [1]	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
SFT	OS-Atlas-7B [44]	12.2	4.7	33.1	1.4	28.8	2.8	37.5	7.3	33.9	5.7	27.1	4.5	28.1	4.0	18.9
SFT	UGround-V1-7B [14]	15.8	1.2	51.9	2.8	47.5	9.7	57.6	14.5	60.5	13.2	38.3	7.9	45.2	8.1	31.1
SFT	UI-TARS-72B [31]	18.8	12.5	62.9	17.2	57.1	15.4	64.6	20.9	63.3	26.4	42.1	15.7	50.9	17.6	38.1
SFT	JEDI-3B* [45]	27.4	9.4	61.0	13.8	53.5	8.4	54.2	18.2	64.4	32.1	38.3	9.0	49.8	13.7	36.1
SFT	JEDI-7B* [45]	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
SFT	GUI-Actor-3B* [43]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	42.2
SFT	GUI-Actor-3B* + Verifier	-	-	-	-	-	-	-	-	-	-	-	-	-	-	45.9
SFT	GUI-Actor-7B* [43]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	44.6
SFT	GUI-Actor-7B* + Verifier	-	-	-	-	-	-	-	-	-	-	-	-	-	-	47.7
RL	GUI-G^2-3B* [35]	20.3	9.4	42.2	2.1	43.4	2.8	43.8	10.0	46.3	15.1	29.0	4.5	37.6	6.0	25.5
RL	UI-R1-E-3B* [25]	37.1	12.5	46.1	6.9	41.9	4.2	56.9	21.8	65.0	26.4	32.7	10.1	-	-	33.5
RL	InfiGUI-R1-3B* [24]	33.0	14.1	51.3	12.4	44.9	7.0	58.3	20.0	65.5	28.3	43.9	12.4	49.1	14.1	35.7
RL	SE-GUI-3B* [53]	38.1	12.5	55.8	7.6	47.0	4.9	61.8	16.4	59.9	24.5	40.2	12.4	50.4	11.8	35.9
RL	GUI-G1-3B* [57]	39.6	9.4	50.7	10.3	36.6	11.9	61.8	30.0	67.2	32.1	23.5	10.6	49.5	16.8	37.1
RL	InfiGUI-G1-3B* [24]	50.8	25.0	64.9	20.0	51.5	16.8	68.8	32.7	70.6	32.1	49.5	15.7	-	-	45.2
RL	UI-TARS-1.5-7B [31]*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	49.6
RL	GroundNext-3B (RL)* [13]	55.3	32.8	65.6	36.6	50.0	24.5	66.0	37.3	74.6	50.9	45.8	29.2	59.9	33.6	49.8
RL	GTA1-7B* [49]	**66.9**	20.7	62.6	18.2	53.3	17.2	76.4	31.8	**82.5**	50.9	48.6	25.9	65.5	25.2	50.1
RL	InfiGUI-G1-7B* [24]	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
RL	GroundNext-7B (RL)* [13]	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-AIMA-3B-lite*	44.7	21.9	68.2	28.3	59.6	21.7	69.4	37.3	70.6	49.1	65.4	31.5	62.0	30.0	49.8
	GUI-AIMA-3B-lite* + 🔍	54.3	26.6	**77.9**	37.2	67.2	31.5	76.4	39.1	80.8	64.2	61.7	32.6	69.5	36.8	57.0
	GUI-AIMA-3B*	50.3	34.4	68.8	46.9	52.0	30.1	75.0	43.6	74.0	50.9	56.1	39.3	62.1	40.2	**53.8**
	GUI-AIMA-3B* + 🔍	58.4	**45.3**	73.4	**48.3**	63.6	**41.3**	77.8	**47.3**	**82.5**	**66.0**	67.3	**49.4**	70.0	**47.9**	**61.5**

OSWorld-G [45] and UI-Vision [27] (in-domain) for the desktop scenario, Screen--Spot-v2 [44] and MMBench-GUI-L2 [42] for general evaluations across mobile, desktop, and web domains. For grounding accuracy, we follow the standard center-point metric [23]: a prediction is considered correct if the click point falls within the ground-truth bounding box. **Baselines.**Three categories of methods are compared: (1) General MLLMs; (2) GUI-specific supervised fine-tuned models; (3) GUI-specific Reinforcement fine-tuned models. For our variants, GUI-AIMA-3B-lite is trained on the subset of GUI-Actor [43]'s training data, while GUI-AIMA-3B additionally incorporates the 250k subset of GroundNext [13]'s training data (710k). UI-Vision results are in-domain for both GUI-AIMA-3B and GroundNext.

4.2 Main Results: Grounding Performance

We compare GUI-AIMA with state-of-the-art methods on ScreenSpot-Pro (Tab. 1), ScreenSpot-v2 (Tab. 2a), UI-Vision, MMBench-GUI-L2 (Tab. 2b), and OSWorld-G (Tab. 3).

Desktop scenario. GUI-AIMA achieves strong performance on challenging high-resolution desktop benchmarks. On ScreenSpot-Pro, GUI-AIMA-3B-lite reaches 49.8% accuracy, and additional desktop-focused training with 200k

Table 2: Performance comparison on **ScreenSpot-v2** and **UI-Vision/MMBench-GUI-L2**. "-" indicates unreported results in original papers. Methods with * are Qwen2.5-VL-based.

	Model	Mobile		Desktop		Web		Avg.
		Text	Icon	Text	Icon	Text	Icon	
General	Operator [30]	47.3	41.5	90.2	80.3	92.8	84.3	70.5
General	GPT-4o [29] + OmniParser-v2 [52]	95.5	74.6	92.3	60.9	88.0	59.6	80.7
General	Qwen2.5-VL-3B [2]	93.4	73.5	88.1	58.6	88.0	71.4	80.9
General	Qwen2.5-VL-7B [2]	97.6	87.2	90.2	74.2	93.2	81.3	88.8
General	Qwen3-VL-8B [1]	**99.7**	87.7	94.8	83.6	95.3	85.7	92.1
General	Qwen3-VL-30B-A3B [1]	99.0	87.7	95.4	82.9	95.3	83.7	91.7
SFT	OS-Atlas-7B [44]	95.2	75.8	90.7	63.6	90.6	77.3	84.1
SFT	UGround-V1-7B [14]	95.0	83.3	95.0	77.8	92.1	77.2	87.6
SFT	UI-TARS-7B [31]	96.9	89.1	95.4	85.0	93.6	85.2	91.6
SFT	JEDI-3B* [45]	96.6	81.5	**96.9**	78.6	88.5	83.7	88.6
SFT	JEDI-7B* [45]	**96.9**	87.2	95.9	87.9	94.4	84.2	91.7
SFT	GUI-Actor-3B [43]	97.6	83.4	**96.9**	83.6	94.0	85.7	91.0
SFT	GUI-Actor-3B + Verifier	98.3	85.3	**96.9**	87.9	95.3	**86.7**	**92.4**
RL	GUI-G^2-3B* [35]	96.5	82.5	95.4	75.0	88.4	72.4	86.3
RL	InfiGUI-R1-3B* [24]	97.1	81.2	94.3	77.1	91.7	77.6	87.5
RL	GroundNext-3B (RL)* [13]	94.8	**96.4**	93.9	87.1	90.6	79.3	88.5
RL	UI-R1-E-3B* [25]	98.2	83.9	93.2	83.7	94.8	75.0	89.5
RL	UI-TARS-1.5-7B* [31]	95.9	84.8	94.9	80.7	90.6	86.2	89.7
RL	GroundNext-7B (RL)* [13]	96.6	88.2	95.4	87.9	94.9	75.9	90.4
RL	InfiGUI-G1-3B* [24]	99.3	88.2	94.8	82.9	94.9	80.3	91.1
	GUI-AIMA-3B-lite*	99.2	85.9	96.1	88.9	**96.1**	80.2	91.5
	GUI-AIMA-3B*	99.6	84.8	95.6	**91.3**	94.3	84.8	92.1

(a) ScreenSpot-v2.

	Model	UI-Vision				MMBench-GUI-L2		
		Basic	Func.	Spatial	Avg.	Basic	Advanced	Avg.
General	Qwen2.5-VL-7B* [2]	1.2	0.8	0.5	0.9	38.0	29.8	33.9
General	Qwen3-VL-2B* [1]	16.4	19.1	4.6	13.1	84.3	54.9	69.5
General	Qwen3-VL-8B* [1]	27.8	29.6	9.4	21.9	90.2	72.7	81.4
General	Qwen3-VL-30B-A3B* [1]	31.2	31.9	14.6	25.6	91.0	76.5	83.7
SFT	OS-Atlas-Base-7B [44]	12.2	11.2	3.7	9.0	52.8	30.1	41.4
SFT	Aguvis-7B [48]	17.8	18.3	5.1	13.7	-	-	45.7
SFT	UGround-V1-7B [14]	15.4	17.1	6.3	12.9	78.4	53.0	65.7
SFT	GUI-Actor-3B* [43]	-	-	-	19.7	-	-	69.8
SFT	GUI-Actor-7B* [43]	-	-	-	21.9	-	-	70.9
SFT	JEDI-7B* [45]	-	-	-	24.8	-	-	70.4
SFT	GroundNext-3B (SFT)* [13]	70.9	59.8	45.1	58.2	86.5	64.5	75.5
RL	UI-TARS-1.5-7B* [31]	-	-	-	20.8	78.4	50.4	64.3
RL	InfiGUI-G1-3B* [24]	31.2	28.0	8.2	22.0	84.1	62.9	73.4
RL	InfiGUI-G1-7B* [24]	36.2	31.9	11.5	26.1	88.3	73.3	80.8
RL	GTA1-7B* [49]	-	-	-	25.7	-	-	79.4
RL	GUI-G^2-7B* [35]	-	-	-	25.6	-	-	79.5
RL	Holo2-8B [15]	43.6	43.5	19.7	35.1	91.7	77.4	84.5
RL	Holo2-30B-A3B [15]	51.0	50.1	23.2	40.9	**92.8**	**80.7**	**86.8**
RL	GroundNext-3B (RL)* [13]	**72.9**	**63.9**	**50.6**	**62.1**	87.3	67.1	77.1
	GUI-AIMA-3B-lite*	39.2	33.9	10.6	27.9	86.2	66.5	76.3
	GUI-AIMA-3B*	70.8	63.1	46.2	60.0	88.3	69.9	79.1

(b) UI-Vision and MMBench-GUI-L2.

GroundCUA instructions further scales GUI-AIMA-3B to 53.8%, outperforming strong SFT and RL baselines. Moreover, our patch-level attention grounding naturally enables a two-step zoom-in inference, which substantially improves performance to 57.0% for GUI-AIMA-3B-lite and 61.5% for GUI-AIMA-3B, with particularly large gains on the abstract icon subset. Despite being trained with less than one-third of the GroundCUA data used by GroundNext-3B, GUI-AIMA-3B achieves comparable in-domain performance on UI-Vision, while outperforming it on the other benchmarks. On OSWorld-G, given already a competitive base model (62.8%), GUI-AIMA-3B is further improved to 68.1% with zoom-in, performing competitively against the strongest large-scale baselines (GTA1-7B and Qwen3-VL-30B-A3B). These results highlight the practical advantage of directly supervising intrinsic attention for dense, high-resolution desktop interfaces.

General grounding across domains. On ScreenSpot-v2, GUI-AIMA-3B outperforms same-scale and larger baselines, and remains competitive with the two-step GUI-Actor-3B with verifier, which is trained on 9.6M samples, demonstrating the data efficiency of GUI-AIMA. Notably, GUI-AIMA achieves strong icon grounding across desktop domains, indicating robust generalization beyond text-heavy cases. On MMBench-GUI-L2, GUI-AIMA-3B consistently outperforms same-scale baselines and matches many larger models.

Overall, GUI-AIMA is both effective and scalable. The intrinsic attention grounding manner provides data efficiency for better scaling and flexibility for inference-time improvements without additional training. The consistent improvement from GUI-AIMA-3B-lite to GUI-AIMA-3B with additional single-domain data further demonstrates the scalability of intrinsic attention supervision. The improvement of GUI-AIMA over GUI-Actor shows the advantage

Table 3: Performance comparison of different models across various task categories based on Mobile, Desktop, Web and Average scores on **OSWorld-G-Refine**. Methods with * are Qwen2.5-VL-based. 🔍 denotes two-step inference with zoom-in.

	Model	Text Matching	Element Recognition	Layout Understanding	Fine-grained Manipulation	Avg.
General	Operator [30]	51.3	42.4	46.6	31.5	40.6
	Gemini-2.5-Pro [8]	59.8	45.5	49.0	33.6	45.2
	Qwen2.5-VL-3B [2]	41.4	28.8	34.8	13.4	27.3
	Qwen2.5-VL-7B [2]	45.6	32.7	41.9	18.1	31.4
	Qwen3-VL-8B [1]	78.2	71.5	72.3	55.7	67.0
	Qwen3-VL-30B-A3B [1]	77.8	<u>75.8</u>	<u>74.7</u>	54.4	**69.3**
SFT	OS-Atlas-7B [44]	44.1	29.4	35.2	16.8	27.7
	UGround-V1-7B [14]	51.3	40.3	43.5	24.8	36.4
	UI-TARS-7B [31]	60.2	51.8	54.9	35.6	47.5
	JEDI-3B* [45]	67.4	53.0	53.8	44.3	50.9
	JEDI-7B* [45]	65.9	55.5	57.7	46.9	54.1
	GUI-Actor-3B* [43]	64.4	60.6	64.8	33.6	54.6
	GUI-Actor-7B* [43]	65.9	62.7	66.4	38.2	56.6
RL	SE-GUI-7B* [53]	-	-	-	-	33.9
	InfiGUI-G1-3B* [24]	65.5	53.0	56.1	34.2	49.6
	InfiGUI-G1-7B* [24]	<u>72.0</u>	63.6	66.8	46.3	59.9
	GUI-G^2-7B* [35]	-	-	-	-	61.9
	UI-TARS-1.5-7B* [31]	52.6	75.4	72.4	<u>66.7</u>	64.2
	GTA-1-7B* [49]	63.2	**82.1**	74.2	**70.5**	67.7
	GUI-AIMA-3B-lite*	64.8	65.5	68.8	36.8	58.3
	GUI-AIMA-3B-lite* + 🔍	71.3	70.6	73.1	47.4	63.8
	GUI-AIMA-3B*	67.4	71.2	72.7	39.5	62.8
	GUI-AIMA-3B* + 🔍	**72.4**	<u>75.8</u>	**78.7**	48.7	<u>68.1</u>

of intrinsic attention supervision without adding an extra module and training stage, with much less training effort in a natural manner.

4.3 Ablations

Tab. 4 shows the ablation results on ScreenSpot-v2 and ScreenSpot-Pro. All the ablation variants are trained on the 45k ablation training data with Qwen2.5-VL-3B-Instruct as the backbone. GUI-Actor (45k) is trained in two stages same as the original setting.

Compare Coordinate-free modeling manners. In Tab. 4, we include 3 different coordinate-free GUI grounding manners, one embedding-based: GUI-Actor, which relies on hidden embeddings for computing similarity with extra modules, and two attention-based: vanilla attention grounding as Eq. (1) and simplified attention grounding as Eq. (2). With the same importance on all the query tokens and attention heads, the vanilla attention grounding can converge faster than GUI-Actor on more complex visual grounding tasks in ScreenSpot-Pro. When simplifying the aggregation on query tokens, even with naive "weighting uniformly", the simplified attention grounding can have **1.50%** and **4.24%** improvement over GUI-Actor for ScreenSpot-v2 and ScreenSpot-Pro. These results demonstrate the effectiveness and faster convergence of attention-based visual grounding methods than embedding-based methods, and show the advantage of compressing query contexts into `<ANCHOR>` token instead of manually controlling the grounding importance from each query token.

Table 4: Ablation results on ScreenSpot-v2 and ScreenSpot-Pro of coordinate-free GUI grounding methods fine-tuned with a 45k ablation dataset. Variants in blue represent the selected settings for GUI-AIMA.

Model	ScreenSpot-v2			ScreenSpot-Pro		
	Text	Icon	**Avg.**	Text	Icon	**Avg.**
Existing Coordinate-free GUI Grounding						
GUI-Actor-3B (45k)	96.24	75.99	87.42	50.67	12.25	35.99
Vanilla Attention Grounding (45k)	95.68	75.45	86.87	53.02	12.42	37.51
Simplified Attention Grounding: GUI-AIMA-3B (45k)						
Attention Head Weighting without $\mathcal{Q}_s$ in Eq. (3)						
+ weighting uniformly	97.08	78.34	88.92	56.50	13.91	40.23
+ weighting with all $\mathcal{Q}' = \mathcal{Q}$	96.24	78.34	88.44	52.61	13.41	37.63
+ weighting with $\mathcal{Q}' = \{$<ANCHOR>$\}$	96.66	76.35	87.81	56.91	14.57	40.73
Attention Head Weighting with $\mathcal{Q}' = \mathcal{Q}_s$ in Eq. (3)						
+ layer-wise top-1 $\mathcal{Q}_s^l$	96.52	**81.23**	**89.86**	57.32	15.73	41.43
+ layer-wise top-3 $\mathcal{Q}_s^l$	96.66	79.06	88.99	57.83	15.07	41.49
+ global top-1 $\mathcal{Q}_s$	**97.49**	78.88	89.39	59.26	14.40	42.13
+ global top-3 $\mathcal{Q}_s$	95.82	70.94	84.98	57.01	14.40	40.73
+ weighted patch-wise labeling	97.21	79.60	89.54	**61.00**	14.90	**43.39**

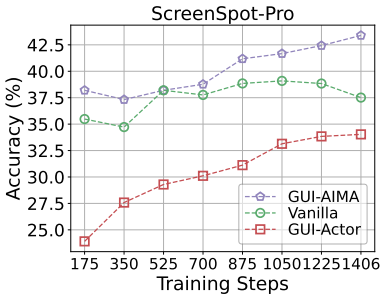

Fig. 5: Performance curves over training steps on ScreenSpot-Pro for GUI-AIMA, GUI-Actor, and vanilla attention grounding trained on the 45k ablation dataset.

Table 5: Inference overhead on OSWorld-G. AR (<ANCHOR>) is constrained to generate the same <ANCHOR> sequence as GUI-AIMA, isolating the grounding attention recomputation. AR (coordinate) denotes coordinate-based grounding with a longer sequence.

	Ours (FlashAttention + partial Eager)		Ours (full Eager)		AR (<ANCHOR>)		AR (coordinate)	
Setting	Time (ms)	Max Mem. (GB)	Time (ms)	Max Mem. (GB)	Time (ms)	Max Mem. (GB)	Time (ms)	Max Mem. (GB)
Single-step	778.5	10.69	4548.1	24.75	**723.6**	**8.40**	1135.3	**8.39**
Two-step	1016.5	10.69	5032.7	24.75	–	–	–	–

GUI grounding without Visual-sink Query Token $\mathcal{Q}_s$. For attention-based GUI grounding variants, weighting the query-visual correlation of each head via the cumulative sum of all query's predictions as $\mathcal{Q}' = \mathcal{Q}$ (TAG's manner) or only the prediction from <ANCHOR> as $\mathcal{Q}' = \{$<ANCHOR>$\}$ in Eq. (3) leads to more biased attention predictions than uniform weighting, with worse ScreenSpot-v2 performance of both variants and no improvement on ScreenSpot-Pro as shown in "Attention Head Weighting without $\mathcal{Q}_s$ in Eq. (3)" part of Tab. 4. This bias comes from the misleading weighting from visual-unrelated tokens for the variants using all $\mathcal{Q}$ and from using the premature <ANCHOR> token for grounding in the later variant.

The effect of Visual-sink Query Token $\mathcal{Q}_s$. For ablation of using $\mathcal{Q}_s$ for $w_{l,h}$, we vary the K value between 1 and 3 in the topK selection of $\mathcal{Q}_s$ tokens and explore whether differing $\mathcal{Q}_s^l$ for each MLLM's layer, i.e., $\mathcal{Q}_s^l := \arg\text{topK}_{q_i \in \mathcal{Q}}(c_{q_i}^l)$, or using the same $\mathcal{Q}_s$ for all layers as in Eq. (5). In Tab. 4, we can observe that the layer-wise manner performs slightly better on the icon-based tasks, but uniformly-selected same $\mathcal{Q}_s$ for all layers leads to more balanced results on both text and icon sets. Among both manners, top-1 selection remains the overall best performance. These ablations verify the "global top-1 $\mathcal{Q}_s$" setting of GUI-AIMA, with **1.90%** improvement over "weighting uniformly" on ScreenSpot-Pro.

Patch-wise labeling considering overlapping and distance. Here, we also justify the overlapping- and distance-aware [35] patch-wise labeling for fine-

Table 6: Sensitivity to Gaussian scaling factor α in Eq. (6).

α	0.6	0.7	0.8	0.9	1.0
ScreenSpot-v2	87.34	89.15	**89.54**	89.47	88.60
ScreenSpot-Pro	39.91	41.62	**43.39**	42.88	42.25

Table 7: Cross-model validation on InternVL3.5 trained with 45K data.

Model	SSv2	SSPro	OSW-G	Overall
InternVL3.5-2B	79.6	12.2	27.7	39.8
GUI-AIMA-InternVL3.5-2B	86.7	16.0	33.9	45.5
InternVL3.5-4B	85.1	18.1	33.9	45.7
GUI-AIMA-InternVL3.5-4B	**88.8**	**19.9**	**36.4**	**48.4**

Table 8: Analysis experiment results on ScreenSpot-pro. **Relax@k** measures how many previously incorrect offsets are recovered when the ground-truth bounding box is expanded by k visual patches along each dimension. **Corrected** refers to the number of offsets corrected by the second step, while **Lost** refers to the number of predictions that degrade after the second step.

Model	Relax@1	Relax@3	Relax@5	**Total**	**Corrected**	**Lost**	**Improved**
1-step							
Origin 1-step	158	92	56	306	-	-	-
2-step							
Crop-only	123	83	43	249	156	107	49
Crop + 1.5×Zoom-in	47	74	38	159	208	48	160
Crop + 2×Zoom-in	39	63	45	147	215	33	**182**
Crop + 3×Zoom-in	39	69	44	152	219	42	177
Crop + 4×Zoom-in	31	72	38	141	224	48	176

tuning GUI-AIMA in Eq. (6). Based on "global top-1 $\mathcal{Q}_s$", down-weighting partial positive and distanced image patches instead of labeling all patches equally leads to **1.26%** further enhancement on ScreenSpot-Pro. We further report the sensitivity to the Gaussian scaling factor α in Eq. (6) in Tab. 6, supporting our selection of $\alpha = 0.8$.

Cross-model validation. To validate the generality of GUI-AIMA beyond Qwen2.5-VL, we apply it to InternVL3.5 [41]'s 2B and 4B models trained with the same 45K ablation data, where the main adaptation is to map visual patch indices back to spatial coordinates under InternVL3.5's dynamic image tiling. As shown in Tab. 7, GUI-AIMA consistently improves over the corresponding backbones, and the adapted 2B model is competitive with the 4B backbone.

Training and inference efficiency. In Fig. 5 , we show the convergence tendency of three methods: GUI-AIMA, vanilla attention grounding and GUI-Actor (with extra warm-up stage already) initialized from Qwen-2.5-VL-3B. GUI-AIMA converges fastest and achieves the best final results, with steady improvements. Vanilla attention grounding fluctuates as supervision on all tokens from the pretrained vocabulary impairs the general capacity. And GUI-Actor needs a longer training period to customize its extra modules for GUI grounding. For inference overhead, we report detailed comparisons in Tab. 5 on OSWorld-G using a NVIDIA A6000 GPU. For a fair comparison, AR (`<ANCHOR>`) is constrained to generate the same `<ANCHOR>` sequence as GUI-AIMA, isolating the grounding attention recomputation. AR (coordinate) denotes coordinate-based grounding with a longer coordinate sequence. Our implementation is time-efficient but more memory-consuming than AR (coordinate), while much more efficient than full eager attention.

4.4 Analysis of Two-step Inference with Zoom-in

In Tab. 8, we present the analysis results of the two-step inference with zoom-in introduced in Sec. 3.3. The analysis focuses on incorrect offset predictions,

where the predicted center points fall outside the ground-truth bounding box by a small degree. Tab. 8 reports the statistics of offset errors for one-step and two-step inference with different offset distances, and the number of Corrected predictions as well as the Lost originally-correct predictions after applying the second crop-and-zoom-in step. We observe that performing inference on the focused region determined by the first-step prediction already corrects more than a 18.6% samples (57) with "Crop-only" operation, arising from fewer irrelevant patches showing as distracting noise for our patch-wise grounding. And after cropping, zooming in by a factor from 1.5× to 4× significantly reduces offset errors, especially those with slight deviation (Relax@1). It also helps resolving large-offset errors, achieving up to 31% (29 fixed) and 32% (18 fixed) error reduction for Relax@3 and Relax@5, respectively. From the analysis results, we find that the performance of two-stem inference is not very sensitive to the zoom-in factor, while a 2-time zoom-in provides the best performance.

5 Conclusion

We present GUI-AIMA, an attention-based, coordinate-free framework for GUI visual grounding that directly supervises intrinsic multi-head self-attention with patch-wise grounding signals. GUI-AIMA replaces impractical vanilla token-wise attention aggregation with an anchored-attention formulation and improves multi-head aggregation via query-aware head weighting guided by visual-sink query tokens. Together with a click-aligned patch-wise labeling and a plug-and-play two-step zoom-in inference, GUI-AIMA provides a lightweight and scalable approach to grounding without introducing extra grounding modules. Across ScreenSpot-Pro, ScreenSpot-v2, OSWorld-G, UI-Vision, and MMBench-GUI-L2, GUI-AIMA achieves strong performance at the 3B scale, demonstrating both data efficiency on a small training set and consistent gains when scaled with additional desktop-domain data. Ablations further validate the contribution of each component, including anchored aggregation, head weighting, and labeling design. Future work includes extending attention-supervised grounding to more general visual grounding tasks and complex interaction scenarios.

Acknowledgements

This work is partially supported under the AI Research Institutes program by National Science Foundation and the Institute of Education Sciences, U.S. Department of Education, through Award # 2229873 – National AI Institute for Exceptional Education, NSF RI-2223292, an Amazon research award, and an Adobe gift fund. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the Institute of Education Sciences, or the U.S. Department of Education. We thank Tong Sun for the discussion.

References

1. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
2. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., et al.: Qwen2.5-vl technical report. arXiv preprint arXiv:2502.13923 (2025)
3. Chai, Y., Huang, S., Niu, Y., Xiao, H., Liu, L., Zhang, D., Ren, S., Li, H.: Amex: Android multi-annotation expo dataset for mobile gui agents. arXiv preprint arXiv:2407.17490 (2024)
4. Chen, W., Cui, J., Hu, J., Qin, Y., Fang, J., Zhao, Y., Wang, C., Liu, J., Chen, G., Huo, Y., et al.: Guicourse: From general vision language models to versatile gui agents. arXiv preprint arXiv:2406.11317 (2024)
5. Chen, Z., Wang, W., Cao, Y., Liu, Y., Gao, Z., Cui, E., Zhu, J., Ye, S., Tian, H., Liu, Z., et al.: Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. arXiv preprint arXiv:2412.05271 (2024)
6. Cheng, K., Sun, Q., Chu, Y., Xu, F., Li, Y., Zhang, J., Wu, Z.: Seeclick: Harnessing gui grounding for advanced visual gui agents. arXiv preprint arXiv:2401.10935 (2024)
7. Clark, K., Khandelwal, U., Levy, O., Manning, C.D.: What does bert look at? an analysis of bert's attention. arXiv preprint arXiv:1906.04341 (2019)
8. Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al.: Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261 (2025)
9. Dao, T., Fu, D., Ermon, S., Rudra, A., Ré, C.: Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in neural information processing systems **35**, 16344–16359 (2022)
10. Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., Su, Y.: Mind2web: Towards a generalist agent for the web. Advances in Neural Information Processing Systems **36**, 28091–28114 (2023)
11. Elhelo, A., Geva, M.: Inferring functionality of attention heads from their parameters. arXiv preprint arXiv:2412.11965 (2024)
12. Fan, Y., Zhao, H., Zhang, R., Shen, Y., Wang, X.E., Wu, G.: Gui-bee: Align gui action grounding to novel environments via autonomous exploration. arXiv preprint arXiv:2501.13896 (2025)
13. Feizi, A., Nayak, S., Jian, X., Lin, K.Q., Li, K., Awal, R., Lù, X.H., Obando-Ceron, J., Rodriguez, J.A., Chapados, N., et al.: Grounding computer use agents on human demonstrations. arXiv preprint arXiv:2511.07332 (2025)
14. Gou, B., Wang, R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., Su, Y.: Navigating the digital world as humans do: Universal visual grounding for gui agents. arXiv preprint arXiv:2410.05243 (2024)
15. H-Company: Holo2: Open foundation models for navigation and computer use agents. https://hcompany.ai/holo2 (2025)
16. Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., 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)

17. Hu, S., Ouyang, M., Gao, D., Shou, M.Z.: The dawn of gui agent: A preliminary case study with claude 3.5 computer use (2024), https://arxiv.org/abs/2411.10323
18. Jeffries, D., Team, K.: Wave ui dataset (2024), https://huggingface.co/datasets/agentsea/wave-ui
19. Li, K., Meng, Z., Lin, H., Luo, Z., Tian, Y., Ma, J., Huang, Z., Chua, T.S.: Screenspot-pro: Gui grounding for professional high-resolution computer use. arXiv preprint arXiv:2504.07981 (2025)
20. Li, K., Patel, O., Viégas, F., Pfister, H., Wattenberg, M.: Inference-time intervention: Eliciting truthful answers from a language model. Advances in Neural Information Processing Systems **36**, 41451–41530 (2023)
21. Li, W., Bishop, W.E., Li, A., Rawles, C., Campbell-Ajala, F., Tyamagundlu, D., Riva, O.: On the effects of data scale on ui control agents. Advances in Neural Information Processing Systems **37**, 92130–92154 (2024)
22. Liang, Y., Zhou, S., Gu, Y., Tan, H., Wu, G., Dernoncourt, F., Kil, J., Rossi, R.A., Zhang, R.: Anticipatory planning for multimodal ai agents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5925–5935 (2026)
23. Lin, K.Q., Li, L., Gao, D., Yang, Z., Wu, S., Bai, Z., Lei, W., Wang, L., Shou, M.Z.: Showui: One vision-language-action model for gui visual agent. arXiv preprint arXiv:2411.17465 (2024)
24. Liu, Y., Liu, Z., Zhu, S., Li, P., Xie, C., Wang, J., Hu, X., Han, X., Yuan, J., Wang, X., et al.: Infigui-g1: Advancing gui grounding with adaptive exploration policy optimization. arXiv preprint arXiv:2508.05731 (2025)
25. Lu, Z., Chai, Y., Guo, Y., Yin, X., Liu, L., Wang, H., Xiao, H., Ren, S., Xiong, G., Li, H.: Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. arXiv preprint arXiv:2503.21620 (2025)
26. Luo, R., Wang, L., He, W., Xia, X.: Gui-r1: A generalist r1-style vision-language action model for gui agents. arXiv preprint arXiv:2504.10458 (2025)
27. Nayak, S., Jian, X., Lin, K.Q., Rodriguez, J.A., Kalsi, M., Awal, R., Chapados, N., Özsu, M.T., Agrawal, A., Vazquez, D., et al.: Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. arXiv preprint arXiv:2503.15661 (2025)
28. Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al.: In-context learning and induction heads. arXiv preprint arXiv:2209.11895 (2022)
29. OpenAI: Introducing gpt-4o. Available at: https://openai.com/index/hello-gpt-4o (2024)
30. OpenAI: Computer-using agent: Introducing a universal interface for ai to interact with the digital world (2025), https://openai.com/index/computer-using-agent
31. Qin, Y., Ye, Y., Fang, J., Wang, H., Liang, S., Tian, S., Zhang, J., Li, J., Li, Y., Huang, S., et al.: Ui-tars: Pioneering automated gui interaction with native agents. arXiv preprint arXiv:2501.12326 (2025)
32. Rawles, C., Clinckemaillie, S., Chang, Y., Waltz, J., Lau, G., Fair, M., Li, A., Bishop, W., Li, W., Campbell-Ajala, F., et al.: Androidworld: A dynamic benchmarking environment for autonomous agents. arXiv preprint arXiv:2405.14573 (2024)
33. Rawles, C., Li, A., Rodriguez, D., Riva, O., Lillicrap, T.: Androidinthewild: A large-scale dataset for android device control. Advances in Neural Information Processing Systems **36**, 59708–59728 (2023)

34. Samaran, J., Garcia, N., Otani, M., Chu, C., Nakashima, Y.: Attending self-attention: a case study of visually grounded supervision in vision-and-language transformers. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop. pp. 81–86 (2021)
35. Tang, F., Gu, Z., Lu, Z., Liu, X., Shen, S., Meng, C., Wang, W., Zhang, W., Shen, Y., Lu, W., et al.: GUI-G^2: Gaussian reward modeling for gui grounding. arXiv preprint arXiv:2507.15846 (2025)
36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
37. Voita, E., Talbot, D., Moiseev, F., Sennrich, R., Titov, I.: Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. arxiv 2019. arXiv preprint arXiv:1905.09418 (2019)
38. Wan, J., Song, S., Yu, W., Liu, Y., Cheng, W., Huang, F., Bai, X., Yao, C., Yang, Z.: Omniparser: A unified framework for text spotting key information extraction and table recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15641–15653 (June 2024)
39. Wang, J., Xu, H., Ye, J., Yan, M., Shen, W., Zhang, J., Huang, F., Sang, J.: Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. arXiv preprint arXiv:2401.16158 (2024)
40. Wang, S., Liu, W., Chen, J., Zhou, Y., Gan, W., Zeng, X., Che, Y., Yu, S., Hao, X., Shao, K., et al.: Gui agents with foundation models: A comprehensive survey. arXiv preprint arXiv:2411.04890 (2024)
41. Wang, W., Gao, Z., Gu, L., Pu, H., Cui, L., Wei, X., Liu, Z., Jing, L., Ye, S., Shao, J., et al.: Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. arXiv preprint arXiv:2508.18265 (2025)
42. Wang, X., Wu, Z., Xie, J., Ding, Z., Yang, B., Li, Z., Liu, Z., Li, Q., Dong, X., Chen, Z., et al.: Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. arXiv preprint arXiv:2507.19478 (2025)
43. Wu, Q., Cheng, K., Yang, R., Zhang, C., Yang, J., Jiang, H., Mu, J., Peng, B., Qiao, B., Tan, R., et al.: Gui-actor: Coordinate-free visual grounding for gui agents. arXiv preprint arXiv:2506.03143 (2025)
44. Wu, Z., Wu, Z., Xu, F., Wang, Y., Sun, Q., Jia, C., Cheng, K., Ding, Z., Chen, L., Liang, P.P., et al.: Os-atlas: A foundation action model for generalist gui agents. arXiv preprint arXiv:2410.23218 (2024)
45. Xie, T., Deng, J., Li, X., Yang, J., Wu, H., Chen, J., Hu, W., Wang, X., Xu, Y., Wang, Z., et al.: Scaling computer-use grounding via user interface decomposition and synthesis. arXiv preprint arXiv:2505.13227 (2025)
46. Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T.J., Cheng, Z., Shin, D., Lei, F., et al.: Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. Advances in Neural Information Processing Systems **37**, 52040–52094 (2024)
47. Xu, H.M., Chen, Q., Wang, L., Liu, L.: Attention-driven gui grounding: Leveraging pretrained multimodal large language models without fine-tuning (2024), https://arxiv.org/abs/2412.10840
48. Xu, Y., Wang, Z., Wang, J., Lu, D., Xie, T., Saha, A., Sahoo, D., Yu, T., Xiong, C.: Aguvis: Unified pure vision agents for autonomous gui interaction. arXiv preprint arXiv:2412.04454 (2024)

49. Yang, Y., Li, D., Dai, Y., Yang, Y., Luo, Z., Zhao, Z., Hu, Z., Huang, J., Saha, A., Chen, Z., Xu, R., Pan, L., Xiong, C., Li, J.: Gta1: Gui test-time scaling agent (2025), https://arxiv.org/abs/2507.05791
50. Yang, Y., Wang, Y., Li, D., Luo, Z., Chen, B., Huang, C., Li, J.: Aria-ui: Visual grounding for gui instructions. arXiv preprint arXiv:2412.16256 (2024)
51. Ye, J., Zhang, X., Xu, H., Liu, H., Wang, J., Zhu, Z., Zheng, Z., Gao, F., Cao, J., Lu, Z., et al.: Mobile-agent-v3: Foundamental agents for gui automation. arXiv preprint arXiv:2508.15144 (2025)
52. Yu, W., Yang, Z., Wan, J., Song, S., Tang, J., Cheng, W., Liu, Y., Bai, X.: Omniparser v2: Structured-points-of-thought for unified visual text parsing and its generality to multimodal large language models (2025), https://arxiv.org/abs/2502.16161
53. Yuan, X., Zhang, J., Li, K., Cai, Z., Yao, L., Chen, J., Wang, E., Hou, Q., Chen, J., Jiang, P.T., et al.: Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. arXiv preprint arXiv:2505.12370 (2025)
54. Zhang, C., Li, L., He, S., Zhang, X., Qiao, B., Qin, S., Ma, M., Kang, Y., Lin, Q., Rajmohan, S., et al.: Ufo: A ui-focused agent for windows os interaction. arXiv preprint arXiv:2402.07939 (2024)
55. Zhang, C., Yang, Z., Liu, J., Li, Y., Han, Y., Chen, X., Huang, Z., Fu, B., Yu, G.: Appagent: Multimodal agents as smartphone users. In: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems. pp. 1–20 (2025)
56. Zheng, B., Gou, B., Kil, J., Sun, H., Su, Y.: Gpt-4v (ision) is a generalist web agent, if grounded. arXiv preprint arXiv:2401.01614 (2024)
57. Zhou, Y., Dai, S., Wang, S., Zhou, K., Jia, Q., Xu, J.: Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents. arXiv preprint arXiv:2505.15810 (2025)