UI-INS: ENHANCING GUI GROUNDING WITH MULTI-PERSPECTIVE INSTRUCTION-AS-REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI grounding, which maps natural-language instructions to actionable UI elements, is a core capability of GUI agents. Prior work largely treats instructions as a static proxy for user intent, overlooking the impact of instruction diversity on grounding performance. Through a careful investigation of existing grounding datasets, we find a 23.3% flaw rate in their instructions and show that inference-time exploitation of instruction diversity yields up to a 76% relative performance improvement. In this paper, we introduce the **"Instruction as Reasoning" paradigm**, treating instructions as dynamic analytical pathways that offer distinct perspective and enabling the model to select the most effective pathway during reasoning. To achieve this, we propose a two-stage training framework: supervised fine-tuning (SFT) on synthesized, diverse instructions to instill multi-perspective reasoning, followed by reinforcement learning (RL) to optimize pathway selection and composition. Our resulting models, UI-Ins-7B and UI-Ins-32B, achieve state-of-the-art results on five challenging benchmarks and exhibit emergent reasoning, selectively composing and synthesizing novel instruction pathways at inference. In particular, UI-Ins-32B attains the best grounding accuracy: **87.3%** on UI-I2E-Bench and **84.9%** on MMBench-GUI L2, besides, UI-Ins-7B yields superior agent performance, achieving a **66.1%** success rate on the AndroidWorld. All code, data, and models will be publicly released.

1 INTRODUCTION

Automated agents for graphical user interfaces (GUIs) are an important frontier in the pursuit of artificial general intelligence (AGI) (Wang et al., 2024b). Their effectiveness hinges on GUI grounding, i.e., the task of mapping a natural-language instruction to the corresponding actionable UI element in a screenshot or live interface.

The natural-language instruction is central to GUI grounding: it is a primary input alongside the GUI screenshot and conveys high-level user intent to be realized as low-level, executable actions. Accordingly, instruction clarity and precision are key determinants of grounding success. However, prior work has offered limited systematic study of instructions themselves. In this paper, we provide a multi-faceted analysis covering instruction diversity, quality, and algorithmic strategies, and establish a concrete basis for more effective grounding.

We focus on instruction diversity and reveal a fundamental mismatch: humans flexibly choose among multiple instructional perspectives, whereas current models are trained in a narrow, fixed style. For example, a single intent such as "close a window", human may describe its **appearance** ("click the red X"), **function** ("close the file manager"), spatial **location** ("the button in the top-right corner"), or high-level **intent** ("get rid of this screen"). Humans strategically switch among these perspectives, choosing the most effective description for the task at hand, as illustrated in Fig. 3. Our quantitative analysis in Sec 2.1 likewise show that leveraging instruction diversity is key to improving grounding accuracy. However, prevailing GUI grounding models are typically trained to map a single instruction style to an action, with limited capacity to reason across perspectives. This limitation forms a key bottleneck to adaptability and robust interpretation in GUI tasks.

Those insights motivate a paradigm shift: rather than treating instructions as static inputs, we should regard them as *dynamic reasoning pathways*. Different instruction types are not merely alternative phrasings; they encode distinct analytical angles for identifying a target. An intelligent GUI agent

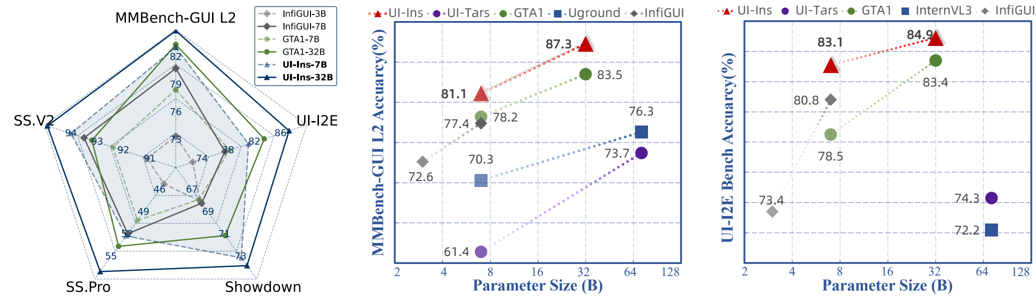


Figure 1: Performance comparisons of UI-Ins and other state-of-the-art methods.

should not only understand a command but also actively select the most effective reasoning process to infer the user's intent. We term this new paradigm **Instruction as Reasoning**.

Beyond this conceptual shift, we also find pervasive instruction-quality issues in grounding datasets. Specifically, we manually inspected 1,909 data entries sampled from prominent datasets, including OS-Atlas (Wu et al., 2024), Omniact (Kapoor et al., 2024), and Android Control (Li et al., 2024). As shown in Fig. 2b, we found that a notable 23.3% of these samples contained various quality deficiencies, introducing considerable noise that could adversely affect model training.

To realize this vision, we introduce a simple and effective framework. We propose a data pipeline systematically cleans noisy annotations and, crucially, augments existing data with a rich diversity of instruction types, creating a dataset curated specifically for multi-perspective instruction reasoning. With this high-quality data as our foundation, we then propose our Instruction as Reasoning framework. This novel two-stage training paradigm first uses Supervised Fine-Tuning (SFT) to explicitly teach the model these diverse reasoning pathways, and then employs Group Relative Policy Optimization (GRPO) (Guo et al., 2025; Shao et al., 2024) in a Reinforcement Learning (RL) stage, enabling the model to learn how to choose the optimal instruction as reasoning for any given situation. Leveraging our effective data processing pipeline and the Instruction as Reasoning algorithm, we introduce the UI-Ins-7B and UI-Ins-32B models. Empirical evaluations conducted across multiple distinct benchmarks validate the strength of our approach, as illustrated in Fig. 1.

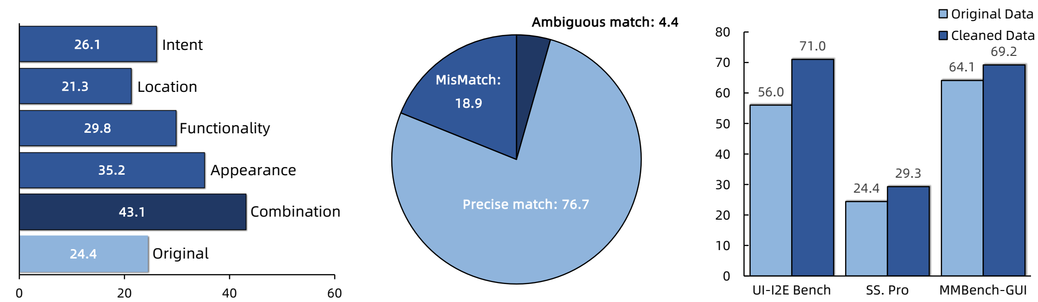
In summary, our contributions are as follows:

- **Systematic Investigation into Grounding Instruction.** We conduct a systematic analysis of instructions in GUI grounding, revealing two crucial insights: (1) a striking **23.3%** of samples' instructions in major datasets are flawed, and (2) there is massive potential in leveraging instruction diversity, which can unlock up to a **76%** relative performance gain even without training.
- **Instruction as Reasoning Paradigm.** Building on the insights above, we pioneer the "Instruction as Reasoning" paradigm, which reframes instructions from static inputs to dynamic reasoning pathways. We realize this through a SFT+GRPO training framework that first teaches the model use diverse instruction perspectives as reasoning and then incentivize it to select the optimal analytical perspective for any given task.
- **SOTA Performance Across Diverse Benchmarks.** Our UI-Ins-7B and UI-Ins-32B establish new SOTA performance across five major grounding benchmarks. Notably, UI-Ins-32B achieves **87.3%** on UI-I2E-Bench and **84.9%** on MMBench-GUI L2, significantly surpassing the strongest baseline. Moreover, our superior grounding capability leads to strong online agent performance on AndroidWorld when combined with GPT-5 as the planner, yielding a **66.1%** success rate.

2 How Much Do Instructions Really Matter?

The natural language instruction is a primary input to grounding tasks, serving as the sole carrier of user intent in GUI grounding. But to what extent do the key aspects of an instruction's formulation, namely its analytical perspective and its correctness, truly impact a model's performance? Prior works have largely treated the instruction as a simple input string, leaving its impact underexplored. We highlight that the instruction is a central, understudied variable in grounding. To probe this view, we conduct a preliminary analysis guided by two foundational research questions:

- **RQ1**: How does the diversity of instructional perspectives affect grounding accuracy?



(a) (b) (c)

Figure 2: Preliminary analysis of GUI Grounding Instructions. **(a)** Instruction diversity influences performance significantly. **(b)** Instruction quality problems in existing open-source datasets. **(c)** Low instruction quality undermines training efficacy.

- **RQ2**: What is the state of instruction quality in existing grounding datasets, and what is its impact?

2.1 Does Instruction Diversity Unlock Higher Performance?

Humans instinctively choose the most effective way to describe an object based on the context like Fig. 3. Does providing a model with similarly diverse, perspective-rich instructions unlock better performance? To investigate this, we conducted a controlled experiment on the ScreenSpot Pro benchmark. We systematically rewrote its original instructions to reflect four distinct perspectives: Appearance, Functionality, Location, and Intent. We then evaluated the zero-shot performance of Qwen2.5-VL-7B on each instruction set.

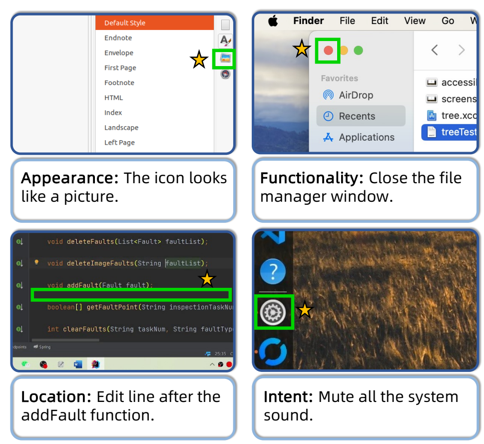


Figure 3: Examples of best-performing instructions in different scenarios.

The results, shown in Fig. 2a, reveal two critical insights. First, instruction diversity matters significantly. Instructions from perspectives of appearance, function, and intent all substantially outperform the original instructions. This demonstrates that *even without retraining, simply providing diverse instruction perspectives can unlock significant latent capabilities within the model*. Second, *the ability to select the most appropriate instruction perspective leads to a higher performance ceiling*. The "Combined" bar, representing the performance if a model could always pick the best-performing perspective for each sample, achieves a relative improvement of 76%, far surpassing any single instruction perspective.

Overall, these results reveal considerable untapped potential in leveraging instruction diversity, both by introducing multiple instruction perspectives and by selecting the optimal perspective per instance. This motivates our algorithm that learns to leverage diverse instruction perspectives as reasoning and dynamically chooses the best analytical angle.

2.2 Can We Trust Existing Datasets for Instruction Quality?

While utilizing instruction diversity is promising, its effectiveness rests on a foundation that the original instructions are correct. But is this foundation valid? To probe the instruction quality of the grounding datasets, we conducted a large-scale manual analysis. Specifically, we examined $1,909$ samples from three prominent datasets, OS-Atlas (Wu et al., 2024), AMEX (Chai et al., 2025), and Widget Captioning (Li et al., 2020).

Our analysis reveals pervasive instruction quality issues. As shown in Fig. 2b, 23.3% of instructions exhibit substantive flaws, including ambiguity or referring to nothing shown in Fig. 4. To further quantify the impact of such flaws, we trained the same model on the original dataset and on a cleaned version. Experimental results are depicted in Fig. 2c: models trained on cleaned data achieve

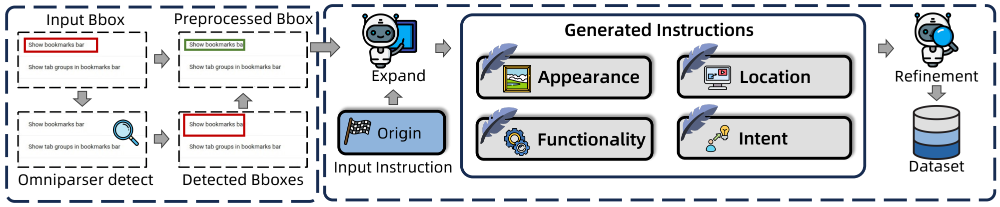


Figure 5: Overview of our data augmentation and verification pipeline.

substantial and consistent performance gains across multiple benchmarks. In other words, flawed instruction data can significantly degrade downstream performance when used for training.

These findings indicate that existing datasets suffer from instruction quality problems that actively harm model performance. Consequently, data cleaning is not optional niceties but necessary prerequisites for meaningful training, especially when our goal is to teach models to leverage diverse instruction perspectives as reasoning.

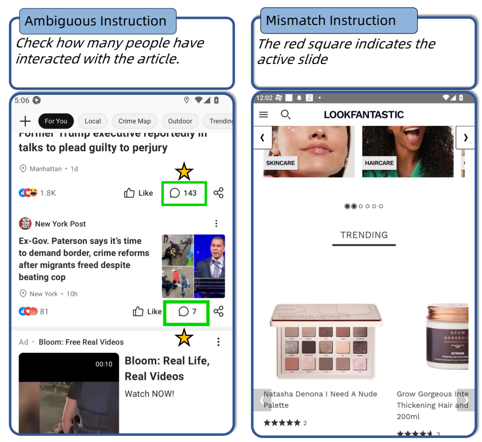


Figure 4: Instruction quality problems. **Left:** Ambiguous match. **Right:** Mismatch.

3 METHOD

Our methodology is architected to address the two fundamental challenges identified in Sec. 2: the pervasive data quality issues and the untapped potential of instruction diversity. We first introduce a high-fidelity data pipeline designed to establish the necessary preconditions for effective model training. With this robust data foundation, we then present our core algorithmic contribution, **Instruction as Reasoning**, a two-stage training framework that empowers models to use diverse instructions as reasoning pathways and to select the optimal analytical perspective during reasoning.

3.1 TASK DEFINITION

GUI Grounding aims to localize the UI element corresponding to an natural language instruction on a graphical interface (Wang et al., 2024b). Formally, given a GUI screenshot $\mathbf{S}$ and a natural language instruction $\mathbf{I}$, the model f should predict a coordinate point $\mathbf{p} = (x_p, y_p)$ that indicates the target element's location.

3.2 DATA PIPELINE FOR MULTI-PERSPECTIVE REASONING

Our preliminary analysis (Sec. 2) revealed that data quality is a prerequisite for meaningful training (Sec 2.2) and that instruction diversity unlocks significant performance gains (Sec. 2.1). To this end, we developed a data processing pipeline focused on two primary objectives: establishing a clean data foundation and then systematically augmenting it with diverse, multi-perspective instructions.

Pre-processing. To rectify the pervasive annotation noise found in existing datasets, we first perform a lightweight pre-processing step. We use OmniParser V2 (Lu et al., 2024) to detect all UI elements on a screenshot and apply a simple IoU-based method to refine or filter the original ground truth bounding box. This ensures each instruction is associated with a reliable spatial anchor, and the flaw instructions are filtered at the same time. The pre-processing forms the clean foundation necessary for the subsequent augmentation.

Multi-Perspective Instruction Augmentation. The core of our pipeline focuses on enriching instruction diversity. We leverage GPT-4.1 (OpenAI, 2025) to generate new instructions from the four fundamental analytical perspectives identified in our analysis: **appearance**, **functionality**, **location**, and **intent**. For each data instance, the model receives the screenshot with the highlighted

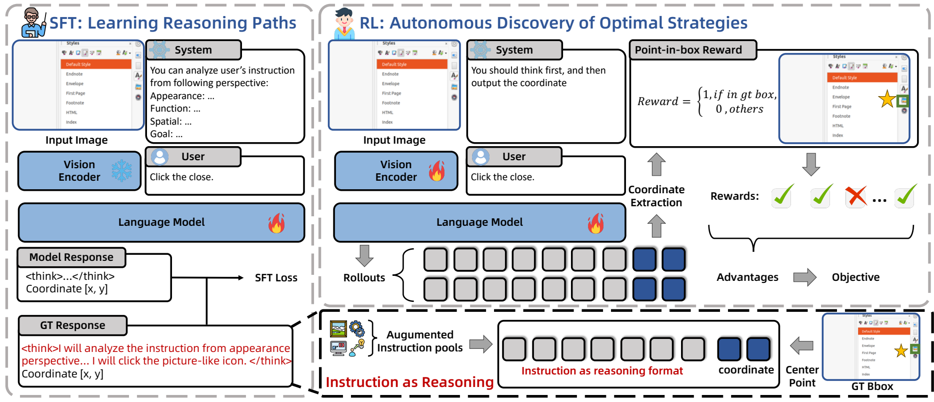


Figure 6: Overview of **Instruction as Reasoning**. We leverage diverse instructions as reasoning process to teach model multi-perspective reasoning paths in SFT stage, then let model explore unconstrained perspectives to find the optimal ways in different scenarios.

target element and is prompted to create a set of high-quality, diverse phrasings. To mitigate LLM hallucinations and ensure a strict one-to-one mapping, each generated instruction undergoes a verification step where GPT-4.1 confirms it unambiguously refers only to the target element. This process yields a high-fidelity, multi-perspective corpus specifically curated to teach complex reasoning.

3.3 Instruction as Reasoning

With such a multi-perspective dataset at hand, we introduce the framework to use it. As discussed in Sec. 2.1, leveraging diverse instruction perspectives and dynamically choosing the best analytical angle are key to unlock superior grounding performance. As shown in Fig. 6, our **Instruction as Reasoning** framework is a two-stage training approach that instills this capability: (i) a SFT stage that teaches the model to use multi-perspective instructions as explicit reasoning pathways, and (ii) a RL stage that trains the model to use the optimal perspective on a per-sample basis.

3.3.1 SFT Stage: Learning to Generate Diverse Reasoning

The goal of the SFT stage is to explicitly instill the model with the ability to perform **Instruction as Reasoning**: utilizing diverse instruction perspectives as analytical reasoning before predicting the grounding coordinates. Concretely, the model first generates an intermediate reasoning text, i.e., a rewritten instruction from one instruction perspective, which serves as an actionable reasoning pathway (Fig. 6). Then outputs the final coordinates.

The grounding model, with parameters θ, is training objective is to maximize the log-likelihood of the target sequence $\mathbf{Y_{gt}}$ across the entire dataset $\mathcal{D}$, formally expressed as:

$$\max_{\theta} \sum_{(\mathbf{S},\mathbf{I},\mathbf{Y}_{gt})\in\mathcal{D}} \log P(\mathbf{Y}_{gt}|\mathbf{S},\mathbf{I};\theta), \quad \text{where } \mathbf{Y}_{gt} = \mathbf{R}_{gt} \oplus \mathbf{p}_{gt} \tag{1}$$

In this formulation, $\oplus$ denotes sequence concatenation. The ground-truth reasoning text, $\mathbf{R}_{gt}$, is randomly sampled from one of the four augmented instruction perspectives, while $\mathbf{p}_{gt}$ represents the ground-truth coordinates. An example of SFT prompt and answer is in Appendix E.1. This unified objective elegantly compels the model to co-optimize two distinct but related skills:

- **Reasoning Generation:** Learning to produce a reasoning ($\mathbf{R}_{gt}$) in an instruction perspective.
- **Grounded Prediction:** Learning to predict the correct coordinates ($\mathbf{p}_{gt}$) conditioned on both inputs and its self-generated reasoning.

By fine-tuning on this objective, the model learns to reasoning from diverse instruction perspectives, creating a foundational skill for RL stage training.

3.3.2 RL STAGE: LEARNING TO SELECT THE OPTIMAL PERSPECTIVE

The SFT stage equips the model with the ability to generate reasoning from multiple instruction perspectives. However, it does not teach the model *which* reasoning pathway is optimal for a given context. To transcend this limitation and incentivize the model to dynamically select the most effective analytical perspective, we introduce a RL stage.

The goal of this stage is to fine-tune the SFT-trained model to discover and select reasoning strategies that maximize grounding accuracy. To achieve this, we employ Group Relative Policy Optimization (GRPO) (Guo et al., 2025). In this phase, we modify the prompt to simply ask the model to"think" before answering, without providing the explicit list of predefined perspectives (appearance, function, etc.). This open-ended instruction encourages the model to explore a wider space of reasoning patterns, including synthesizing multiple perspectives or even formulating entirely novel ones. The model then learns to select the optimal analytical perspective from the feedback of RL rewards.

We calculate rewards by a point-in-box function, then, the rewards $\{r_i\}_{i=1}^{G}$ are normalized into advantages via Z-score normalization:

$$\hat{A}_{i,t} = \frac{r_i - \frac{1}{G}\sum_{i=1}^{G} r_i}{\sqrt{\frac{1}{G}\sum_{i=1}^{G}\left(r_i - \frac{1}{G}\sum_{i=1}^{G} r_i\right)^2}} \tag{2}$$

where G is the rollout number. Finally, the model is optimized by minimizing the objective:

$$L = -\frac{1}{G}\sum_{i=1}^{G} \frac{\pi(o_i \mid I, S)}{\pi_{\text{old}}(o_i \mid I, S)} \cdot \hat{A}_{i,t} \tag{3}$$

where $\pi_{\text{old}}(\cdot \mid \cdot)$ denotes the old policy and $\hat{A}_{i,t}$ is the advantage associated with prediction o_i. By iteratively applying this process, the model learns to prioritize reasoning pathways that consistently lead to correct grounding, effectively learning an optimal, context-dependent strategy for instruction perspective selection. Interestingly, we find that the model also learns to synthesize multiple perspectives and even formulate entirely novel instruction perspectives (detailed in Sec 4.4).

4 EXPERIMENT AND RESULTS

Table 1: Overall performance on **MMBench-GUI L2** and **UI-I2E-Bench** benchmarks. The aggregated accuracy (%) for different instruction types is reported. We use '-' to denote unavailability, and '*' to denote the results evaluated by us.

Model	**Size**	**MMBench-GUI L2**			**UI-I2E-Bench**		
		Basic	Advanced	Avg.	Explicit	Implicit	Avg.
Qwen2.5-VL (Bai et al., 2025)	7B	38.0	29.8	33.9	58.4	51.0	53.8
OS-Atlas (Wu et al., 2024)	7B	52.8	30.1	41.4	63.2	55.8	58.6
Aguvis (Xu et al., 2025)	7B	51.0	40.5	45.7	61.1	48.4	53.2
Uground-V1 (Gou et al., 2025)	7B	78.4	53.0	65.7	81.3	63.6	70.3
UI-TARS-1.5 (Seed, 2025)	7B	78.4	50.4	64.3	81.3	68.2	73.2
UI-TARS (Qin et al., 2025)	7B	-	-	-	71.4	55.3	61.4
UI-I2E-VLM (Liu et al., 2025a)	7B	-	-	-	72.0	67.9	69.5
InfiGUI-G1 (Liu et al., 2025c)	7B	88.5	73.2	80.8	85.0	72.7	77.4
GTA1 (Yang et al., 2025)	7B	84.4*	72.6*	78.5*	87.0*	72.8*	78.2*
GTA1 (Yang et al., 2025)	32B	89.0*	77.9*	83.4*	91.4*	78.7*	83.5*
Qwen2.5-VL (Bai et al., 2025)	72B	54.4	29.3	41.8	49.6	52.5	51.4
Uground-V1 (Gou et al., 2025)	72B	-	-	-	84.5	71.3	76.3
UI-TARS-DPO (Qin et al., 2025)	72B	83.2	65.6	74.3	-	-	-
UI-TARS (Qin et al., 2025)	72B	-	-	-	80.9	69.4	73.7
InternVL3 (Zhu et al., 2025)	72B	80.4	64.1	72.2	-	-	-
UI-Ins-7B	7B	89.0	77.3	83.1	88.9	76.3	81.1
UI-Ins-32B	32B	**90.5**	**79.4**	**84.9**	**92.9**	**83.9**	**87.3**

Table 2: Performance comparison on **ScreenSpot-Pro**, **ScreenSpot-V2**, and **ShowDown**.

Model	Size	ScreenSpot-Pro			ScreenSpot-V2			ShowDown
		Text	Icon	Avg.	Text	Icon	Avg.	Avg.
UI-R1 (Lu et al., 2025)	3B	23.3	6.8	17.8	95.6	81.6	89.5	-
ZonUI (Hsieh et al., 2025)	3B	38.3	13.0	28.7	-	-	-	-
Qwen2.5-VL (Bai et al., 2025)	7B	2.1	0.3	1.6	94.2	81.8	88.8	-
OS-Atlas (Wu et al., 2024)	7B	-	-	-	92.5	73.3	85.1	41.1
GUI-R1 (Luo et al., 2025)	7B	41.5	11.7	31.0	-	-	-	-
UI-TARS (Qin et al., 2025)	7B	46.0	16.0	35.7	95.4	86.6	91.6	66.1
UI-TARS-1.5 (Seed, 2025)	7B	-	-	42.0	92.9	83.3	89.0	67.2
UI-AGILE (Lian et al., 2025)	7B	58.7	18.0	44.0	-	-	-	-
GUI-G^2 (Tang et al., 2025)	7B	64.9	18.4	47.5	96.1	89.7	93.3	-
UGround-v1 (Gou et al., 2025)	7B	-	-	-	88.1	86.8	87.7	57.8
InfiGUI-G1 (Liu et al., 2025c)	7B	69.1	24.5	51.9	97.4	88.4	93.5	68.2*
GTA1 Yang et al. (2025)	7B	58.7	**34.9**	50.1	95.7	88.1	92.4	67.9*
Phi-ground (Zhang et al., 2025)	7B	-	-	43.2	93.2	71.0	83.8	62.5
GUI-Actor (Wu et al., 2025)	7B	-	-	44.6	96.0	87.0	92.1	-
SE-GUI (Yuan et al., 2025)	7B	61.8	22.8	43.2	-	-	90.3	-
GTA1 Yang et al. (2025)	32B	65.6	28.1	53.6	97.1	88.3	93.2	71.1*
UI-Ins-7B	7B	70.0	23.5	52.2	**98.2**	88.6	94.0	73.1
UI-Ins-32B	32B	**73.7**	30.0	**57.0**	**98.2**	**90.6**	**94.9**	**73.8**

4.1 Experimental Settings

Data and Implementation Details We source data from several public datasets, including OS-Atlas, Omniact, Android Control, AMEX, and AgentNet, covering diverse operating systems such as Windows, MacOS, Linux, and Android. All data is subsequently processed through our pipeline to ensure quality. We employ Qwen2.5-VL-7B and Qwen2.5-VL-32B as our backbone architectures. More data details are in Sec. E and more implementation details are in D.1.

Baselines and Metrics We compare our method against extensive recent SOTA baselines to provide a comprehensive grounding performance evaluation. These include models that are primarily trained using supervised fine-tuning, such as Jedi (Xie et al., 2025) and Aguvis (Xu et al., 2025), as well as methods that incorporate RL paradigm, such as GUI-Actor (Wu et al., 2025) and InfiGUI-G1 (Liu et al., 2025c). Besides, we also compare UI-Ins with some agentic frameworks such as AgentS2 (Zhou et al., 2024) and InfiGUIAgent (Liu et al., 2025b) on the online benchmark.
Following prior works (Yang et al., 2025; Liu et al., 2025c; Tang et al., 2025), we evaluate GUI Grounding performance using the point-in-box accuracy.

Evaluation Benchmarks We evaluate our method on five widely-used grounding benchmarks and a challenging online agent environment.

- **Grounding Benchmarks:** MMBench-GUI L2 (Xuehui Wang et al., 2025) tests performance on hierarchical instructions, while UI-I2E-Bench (Liu et al., 2025a) focuses on explicit instructions and deeper semantic reasoning for implicit instructions. Showdown (Team, 2025) evaluates instruction-following and low-level control capabilities. ScreenSpot-Pro Li et al. (2025) examines semantic understanding in high-resolution professional software.
- **Online Agent Benchmark:** To evaluate our model's practical utility in a dynamic setting, we report performance on **AndroidWorld** (Rawles et al., 2024a). This benchmark is particularly challenging as it requires the agent to complete multi-step tasks in a live, interactive environment.

4.2 Results

Main Results As shown in Tab. 1, UI-Ins-32B achieves state-of-the-art (SOTA) results on both the MMBench-GUI L2 and UI-I2E-Bench benchmarks, while UI-Ins-7B demonstrates a significant performance advantage over similarly-sized models. While our models show improvements on basic and explicit instructions, they exhibit even more substantial gains on the challenging "advanced" (MMBench-GUI L2) and "implicit" (UI-I2E-Bench) subsets. This further validates the effectiveness of our Instruction as Reasoning approach. Furthermore, to provide a broader validation of our models' capabilities, we conduct extensive evaluations on the ScreenSpot-V2, ScreenSpot-Pro, and

Showdown benchmarks. As detailed in Tab. 2, UI-Ins-32B again achieves SOTA performance, and UI-Ins-7B consistently delivers superior results compared to its peers in the same parameter class. UI-Ins-32B performs well in different os platforms(Fig. 7) and we also provide a error analysis, which indicates the lack of knowlegde(Fig. 8) and hallucination can causes(Fig. 9) the failure. More result details are shown in Sec. F.1.

Online Agent Results To assess real-world utility, we evaluated our model as the grounding component for a mobile agent on the challenging AndroidWorld benchmark (Rawles et al., 2024b). As shown in Tab. 3, Paired with a GPT-5 planner, our UI-Ins model achieves a **66.1%** success rate. This result significantly outperforms specialized baselines, demonstrating that superior grounding capability directly translates to enhanced agent performance.

Table 3: Performance on AndroidWorld.

Model	Success Rate
InfiGUIAgent (Liu et al., 2025b)	9.0
Ponder&Press (Wang et al., 2024a)	34.5
Uground (Gou et al., 2025)	44.0
Aria-UI (Yang et al., 2024)	44.8
UI-Tars (Qin et al., 2025)	46.6
AgentS2 (Zhou et al., 2024)	54.3
UI-Ins-7B	**66.1**

4.3 Ablation Study

Data Pipeline Ablation Study We first manually inspect 1542 data produced by data pipeline, where the error rate is less than 8%. This is significantly lower than the inital error rate, 23.3%. To further validate the effectiveness of our data pipeline, we conduct an ablation study via SFT training. As shown in Tab. 11, our data pipeline provides a consistent performance improvement across multiple benchmarks. We show the details in Sec. C.2

Table 4: Ablation study on training stages and the reasoning component. We report accuracy on MMBench-GUI L2 (MM), UI-I2E-Bench (I2E), Showdown (Show), ScreenSpot-Pro (Pro), and ScreenSpot-V2 (V2).

SFT	RL	MM	I2E	Show	Pro	V2
Ablation: Training stages.						
✗	✗	63.4	56.0	43.6	24.4	86.5
✗	✓	72.4	69.2	66.6	37.0	88.6
✓	✗	76.3	70.1	67.5	37.1	90.6
✓	✓	**83.1**	**81.1**	**73.1**	**52.2**	**94.0**
Ablation: Reasoning in training stages.						
✗	✗	79.1	70.7	66.1	44.8	91.7
✗	✓	78.8	71.6	68.4	48.0	92.0
✓	✗	81.6	76.2	72.0	47.5	93.1
✓	✓	**83.1**	**81.1**	**73.1**	**52.2**	**94.0**

Training Stage Ablation Study Here we validate the necessity of SFT+RL training stages for the **Instruction as Reasoning** method. We compare the Qwen2.5-VL-7B model against two variants: one trained only with SFT and another trained only with RL . In all configurations, the model is prompted to generate an intermediate reasoning step. Top results of Tab. 4 indicate that both the SFT and RL stages are critical for achieving optimal performance. The absence of either stage leads to a accuracy degradation, highlighting the importance of first teaching the model diverse reasoning paths and then allowing it to autonomously optimize its strategy.

4.4 Deeper Insights into Instruction-as-Reasoning

Having established the strong performance of our models, we now delve deep into the *Instruction-as-Reasoning* framework to understand its success. We investigate three central questions below:

Is an intermediate reasoning step necessary? A fundamental question is whether letting the model to generate an intermediate reasoning trace is beneficial at all. To answer this, we conducted an ablation study by completely removing the reasoning generation component from both the SFT and RL stages, training the model to directly predict coordinates. Experimental results are depicted in Tab. 4. Compared to our method (the 4th row), removing reasoning (the first row) leads to a substantial performance drop across all benchmarks, with an accuracy

Table 5: Comparison between free-form reasoning and Instruction as Reasoning in RL stage.

Method	SS.Pro	SS.V2
Free-Form Reasoning (FFR) in RL		
RL (w/o FFR)	50.1	92.4
RL (w/ FFR)	46.9↓ (6.4)%	93.2↑ (0.8)%
Instruction-as-Reasoning (IR) in RL		
RL (w/o IR)	47.5	93.1
RL (w/ IR)	52.2↑ (9.9%)	94.0↑ (0.9%)

decrease over 10% on UI-I2E-Bench. This result confirms including intermediate reasoning step is crutial to the success of Instruction-as-Reasoning framework.

Instruction-as-Reasoning vs. Free-Form Thinking Given that reasoning is critical, what kind of reasoning is effective? Prior works (Lu et al., 2025; Yang et al., 2025) have shown that free-form-reasoning often fails to improve, and can even degrade, grounding performance. We test this hypothesis against our instruction-as-Reasoning approach in Tab. 5.
First, we examine Free-Form Reasoning (FFR). As the top section of Tab. 5 shows, applying FFR on a standard SFT model degrades performance, causing a 6.4% relative drop on SS.Pro.
In contrast, we evaluate our Instruction-as-Reasoning (IR) approach. As the bottom section of the table 4 shows, training the model with IR yields a significant accuracy increase by a relative 9.9%. We can thus conclude from the experiments that free-form-thinking fails to improve, where as our instruction-as-thinking is the key to unlocking effective reasoning for GUI grounding.

The Hidden Benefit: Stabilizing SFT+RL We compare our SFT+RL framework with a standard one in Tab. 6. When a model is trained with a standard coordinate-based SFT and then moved to RL, it suffers from policy collapse and leads to performance degradation. In contrast, our instruction-as-reasoning-based SFT acts as a powerful exploratory warm-up. By pre-training the model to generate diverse reasoning pathways, we empower it with a strong exploratory capability, achieving significant performance increase during RL. This demonstrate that our SFT strategy not only teaches reasoning format, but also enables effective and stable policy optimization in the RL phase.

Table 6: Impact of reasoning in SFT in early RL stages. We report scores after 100 RL steps and the relative change from the SFT-only baseline.

Method	SS.Pro	SS.V2
SFT (w/o IR)	37.0	90.6
SFT (w/o IR) + RL	34.9↓ (5.7%)	89.9↓ (0.8%)
SFT (w/ IR)	37.1	90.6
SFT (w/ IR) + RL	46.0↑ (24.0%)	92.8↑ (2.4%)

Emergent Capabilities: Reasoning Beyond Predefined Perspectives Does our framework merely teach the model to use the four predefined perspectives? A qualitative analysis of 500 model responses reveals that it learns far deeper. We observe three key emergent capabilities:
Strategic Selection: The model learns to strategically select different reasoning perspectives for different scenarios, as shown in Tab. 7. Besides this, model just trained after SFT stage also have the ability to select the a better perspective(Tab. 16).
Spontaneous Synthesis: The model often combines multiple perspectives into a single, cohesive reasoning (e.g., "Find the blue 'Save' button [appearance] at the bottom of the form [location]). This synthesis is not explicitly taught but emerges as an effective reasoning strategy during RL.
Emergent Perspective: Most impressively, the model is capable of generating entirely new analytical angles beyond the four trained perspectives, such as reasoning from the instruction perspective of current state or component type.

Table 7: Details of reasoning types categorized by GPT-4.1 from 500 UI-Ins-7B thinking processes. Note that a process may contain multiple types, resulting in a total of 1,950 reasoning instances.

Perspective	App.	Loc.	Comp.	Str.	Func.	Intent	Seq.	Others	Total
Count	608	569	284	197	194	54	19	25	**1950**
Percentage (%)	31.2	29.2	14.6	10.1	9.9	2.8	1.0	1.3	**100.0**

5 Conclusion

In this work, we conducted a systematic investigation into the natural language instruction, a critical yet underexplored component of GUI grounding. We first identified and quantified the severe quality and diversity issues prevalent in existing open-source datasets. To address these, we introduced a high-fidelity two-stage data pipeline to curate a reliable foundation for model training. Building upon this, we proposed **Instruction as Reasoning**, a novel SFT+RL framework designed to explicitly leverage instructional diversity by treating different perspectives as distinct reasoning pathways. Our resulting models, UI-Ins-7B and UI-Ins-32B, establish a new state of the art across five benchmarks, verifying the effectiveness our approach.

6 REPRODUCIBILITY STATEMENT

We are committed to full reproducibility. Upon publication, we will release all code, data, and models.

- **Code and Models:** The source code for our training framework for UI-Ins-7B and UI-Ins-32B, will be made public. The models will also be released.
- **Data:** Our cleaned and augmented dataset will be released. Data processing details and prompts are described in Section 3.2 and Appendix C.
- **Hyperparameters:** All implementation details are specified in Section D.1, enabling a faithful replication of our experiments.

Further environment details will be included in the public repository to ensure a smooth replication process.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2138–2156. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.110. URL http://dx.doi.org/10.18653/v1/2025.findings-acl.110.

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.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

ZongHan Hsieh, Tzer-Jen Wei, and ShengJing Yang. Zonui-3b: A lightweight vision-language model for cross-resolution gui grounding. https://arxiv.org/abs/2506.23491, 2025. arXiv:2506.23491 [cs.CV], version 2, last revised 1 Jul 2025.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models, 2025. URL https://arxiv.org/abs/2503.06749.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web, 2024.

Kaixin Li, Meng Ziyang, 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 *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=XaKNDIAHas.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents, 2024. URL https://arxiv.org/abs/2406.03679.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements, 2020. URL https://arxiv.org/abs/2010.04295.

Shuquan Lian, Yuhang Wu, Jia Ma, Yifan Ding, Zihan Song, Bingqi Chen, Xiawu Zheng, and Hui Li. Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding, 2025. URL https://arxiv.org/abs/2507.22025.

Xinyi Liu, Xiaoyi Zhang, Ziyun Zhang, and Yan Lu. Ui-e2i-synth: Advancing gui grounding with large-scale instruction synthesis, 2025a. URL https://arxiv.org/abs/2504.11257.

Yuhang Liu, Pengxiang Li, Zishu Wei, Congkai Xie, Xueyu Hu, Xinchen Xu, Shengyu Zhang, Xiaotian Han, Hongxia Yang, and Fei Wu. Infiguiagent: A multimodal generalist gui agent with native reasoning and reflection. *arXiv preprint arXiv:2501.04575*, 2025b.

Yuhang Liu, Zeyu Liu, Shuanghe Zhu, Pengxiang Li, Congkai Xie, Jiasheng Wang, Xueyu Hu, Xiaotian Han, Jianbo Yuan, Xinyao Wang, et al. Infigui-g1: Advancing gui grounding with adaptive exploration policy optimization. *arXiv preprint arXiv:2508.05731*, 2025c.

Yuqi Liu, Bohao Peng, Zhisheng Zhong, Zihao Yue, Fanbin Lu, Bei Yu, and Jiaya Jia. Seg-zero: Reasoning-chain guided segmentation via cognitive reinforcement. *arXiv preprint arXiv:2503.06520*, 2025d.

Yuqi Liu, Tianyuan Qu, Zhisheng Zhong, Bohao Peng, Shu Liu, Bei Yu, and Jiaya Jia. Vision-reasoner: Unified visual perception and reasoning via reinforcement learning. *arXiv preprint arXiv:2505.12081*, 2025e.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent, 2024. URL https://arxiv.org/abs/2408.00203.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Guanjing Xiong, and Hongsheng Li. Ui-r1: Enhancing action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*, 2025.

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

OpenAI. Gpt-4.1 announcement. https://openai.com/index/gpt-4-1/, 2025. Accessed: 2025-08-03.

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.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2024a. URL https://arxiv.org/abs/2405.14573.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2024b. URL https://arxiv.org/abs/2405.14573.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. *URL https://arxiv. org/abs/2402.03300*, 2(3):5, 2024.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. Gui-g^2: Gaussian reward modeling for gui grounding, 2025. URL https://arxiv.org/abs/2507.15846.

General Agents Team. The showdown computer control evaluation suite, 2025. URL https://github.com/generalagents/showdown.

Ye Wang, Ziheng Wang, Boshen Xu, Yang Du, Kejun Lin, Zihan Xiao, Zihao Yue, Jianzhong Ju, Liang Zhang, Dingyi Yang, Xiangnan Fang, Zewen He, Zhenbo Luo, Wenxuan Wang, Junqi Lin, Jian Luan, and Qin Jin. Time-r1: Post-training large vision language model for temporal video grounding, 2025. URL https://arxiv.org/abs/2503.13377.

Yiqin Wang, Haoji Zhang, Jingqi Tian, and Yansong Tang. Ponder & press: Advancing visual gui agent towards general computer control. *arXiv preprint arXiv:2412.01268*, 2024a.

Yiqin Wang, Haoji Zhang, Jingqi Tian, and Yansong Tang. Ponder & press: Advancing visual gui agent towards general computer control, 2024b. URL https://arxiv.org/abs/2412.01268.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *arXiv preprint arXiv:2506.03143*, 2025.

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: A foundation action model for generalist gui agents, 2024. URL https://arxiv.org/abs/2410.23218.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis, 2025. URL https://arxiv.org/abs/2505.13227.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction, 2025. URL https://arxiv.org/abs/2412.04454.

JingJing Xie Xuehui Wang, Zhenyu Wu et al. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. *arXiv preprint arXiv:2507.19478*, 2025.

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, 2025. URL https://arxiv.org/abs/2507.05791.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions, 2024. URL https://arxiv.org/abs/2412.16256.

Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025.

Miaosen Zhang, Ziqiang Xu, Jialiang Zhu, Qi Dai, Kai Qiu, Yifan Yang, Chong Luo, Tianyi Chen, Justin Wagle, Tim Franklin, et al. Phi-ground tech report: Advancing perception in gui grounding. *arXiv preprint arXiv:2507.23779*, 2025.

Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents. 2024. URL https://arxiv.org/abs/2406.18532.

Yuqi Zhou, Sunhao Dai, Shuai Wang, Kaiwen Zhou, Qinglin Jia, and Jun Xu. Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents, 2025. URL https://arxiv.org/abs/2505.15810.

Jinguo Zhu, Weiyun Wang, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models, 2025. URL https://arxiv.org/abs/2504.10479.

A THE USE OF LARGE LANGUAGE MODELS (LLMS)

LLMs are only used in polish writing.

B RELATED WORK

B.1 REINFORCEMENT LEARNING FOR MLLMS

Reinforcement Learning (RL) has demonstrated a distinct advantage in enhancing the generalization capabilities of Multimodal Large Language Models (MLLMs), leading to its rapid adoption across various vision-language downstream tasks. Prior works such as Seg-Zero (Liu et al., 2025d) and Vision Reasoner (Liu et al., 2025e) have showcased its unique strengths in general-purpose grounding. Concurrently, Vision-R1 (Huang et al., 2025) revealed that RL algorithms can significantly boost the reasoning abilities of MLLMs. Beyond static images, this paradigm has also been extended to the video domain, where Time-R1 (Wang et al., 2025) successfully applied GRPO to video temporal localization tasks, achieving exceptional performance.

B.2 GROUNDING FOR GUI AGENTS

GUI grounding has recently undergone rapid development, and current GUI agents can be primarily categorized by their training methodologies. Early works predominantly employed Supervised Fine-Tuning (SFT). For instance, Jedi (Xie et al., 2025) synthesized a 4-million-example dataset using multi-perspective decoupling to improve SFT for grounding. AGUVIS (Xu et al., 2025) introduced a unified, vision-based framework that operates directly on screen images, enabling cross-platform interaction through a two-stage training pipeline. Similarly, OS-Atlas (Wu et al., 2024) addressed the lack of high-quality open-source data by introducing a cross-platform grounding corpus with over 13 million elements.More recently, a majority of works have transitioned to RL-based training methods, which typically yield higher performance and better generalization. For example, GUI-G1 (Zhou et al., 2025) refines the online RL training method by proposing a fast-thinking template and a difficulty-aware policy update. To tackle semantic alignment issues, InfiGUI-G1 (Liu et al., 2025c) introduced Adaptive Exploration Policy Optimization (AEPO), a framework that enhances exploration through a multi-answer strategy. Diverging from coordinate-based methods, GUI-Actor (Wu et al., 2025) proposed a novel attention-based action head that learns to align a special '<ACTOR>' token with visual features, enabling a coordinate-free approach to grounding.

Table 8: Manual inspection results of our refined dataset, breaking down error types by count and percentage out of 1,542 total instances.

Metric	Ambiguous match	Mismatch	Precise match	All
Count	18	81	1443	1542
Percentage (%)	1.17	5.25	93.58	100.00

Table 9: Summary of SFT Data Sources

Metric	AgentNet	Os-Atlas	AMEX	OmniAct	Total
Count	138,635	73,335	70,023	1,159	**283,152**
Percentage (%)	50.8	26.9	25.7	0.4	**100.0**

Table 10: Summary of RL Data Sources

Metric	AgentNet	Os-Altas	AMEX	OmniAct	Android Control	Total
Count	12,570	11,048	6,553	1,520	1,160	**32,851**
Percentage (%)	38.3	33.6	19.9	4.6	3.5	**100.0**

C DATA DETAILS

C.1 INSTRUCTION DIVERSITY AUGMENTATION

To enhance instructional diversity, we expanded the instruction set based on frequently occurring scenarios, categorizing them into four types: appearance-based, function-based, spatial-based, and intent-based. When leveraging GPT-4.1 to augment instructions from open-source datasets, we mitigated potential hallucinations arising from poor-quality original instructions. To achieve this, we visually grounded the process by overlaying the ground-truth point or bounding box as a distinct circular or rectangular marker on the input image.

Instruction Diversity Augumentation Prompt

Task:
Generate and Translate Unambiguous Grounding Instructions
Input:
GUI Screenshot: An image of a user interface.
Original Instruction: An initial English instruction.
Highlighted Element: A visual marker e.g., a red <annotation_type> on the screenshot pointing to the target UI element.
— CORE OBJECTIVE —
Your primary task is to first translate the Original Instruction into high-quality Chinese, and then generate four new, distinct types of grounding instructions. For all generated instructions, you must adhere to this critical rule: the instruction must correspond to one and only one element on the entire screen—the one highlighted. Clarity and uniqueness are the top priorities.
— IMPORTANT SAFEGUARD —
The <annotation_type> is a ground-truth annotation provided only for your reference. Your instructions must never refer to the annotation itself.
It is noticeable that the original instruction may can not align with the ground-truth annotation, you should follow the ground-truth annotation first.
Instructions Generation Requirements:
Generate one new, clear, and unambiguous instruction for each of the following four categories.
Appearance-Based:
A direct and literal description of the element's visual characteristics (e.g., its text, icon, color, shape). Combine features as needed to ensure the description is completely unique.
Function-Based:
A clear description of the element's purpose or the immediate outcome of interacting with it (e.g., "the button used to confirm and save your profile changes").
Spatial-Based:
An instruction that identifies the element based on its position relative to other prominent, easily identifiable UI elements (landmarks). The described spatial relationship must lead to a unique location.
Goal-Based:
A concise phrase that describes the user's ultimate goal or intent. The user must infer which single UI element on the screen fulfills this goal.
Output Format:
The final output must be a single, well-formed JSON object. The JSON structure should begin with the original instruction and its translation, followed by the newly generated instructions.
Now, please process the following inputs and generate the instructions in the specified JSON format.
Original Instruction:
<instruction_here>

Prompt for Instruction Refinement breakable

Task:
Quality Evaluation of a GUI Grounding Datum
Role:
You are a meticulous Data Quality Analyst specializing in user interface datasets. Your task is to critically evaluate a given data sample for its quality and correctness in a structured, two-step process.
Input:
GUI Screenshot: An image of a user interface.
Grounding Instruction: An English command intended to guide a user to a specific element.
Ground-Truth Bounding Box: A red box drawn on the screenshot, highlighting the target UI element.
——-IMPORTANT——-
Ground-Truth Point: A blue hollow circle drawn on the center of the Ground-Truth Bounding Box, which is the key to help you locate the target UI element, because screenshots usually have other red bboxes which may cause distribution.
Output Process (Two Steps):
Step 1: Chain-of-Thought Reasoning
First, you must articulate your reasoning process in plain text. Analyze the input and think step-by-step. Your reasoning should cover the following points:
Instruction Analysis:
What specific element does the instruction describe? Identify its key features (text, function, location, etc.), it is important you should locate the target UI element according to the blue hollow circle and the red bbox.
Scan the entire screenshot. Are there any other elements that could match this description, even partially?
Conclude whether the instruction is unique or ambiguous based on this scan.
Bounding Box Analysis:
What is the target element identified by the instruction? Does it have the blue hollow circle in the center of the box?
Does the red box tightly enclose this entire target element?
Does the box cut off any part of the element?
Does the box include significant empty space or other unrelated elements?
Conclude whether the bounding box is appropriately sized, too large, or too small.
Step 2: Final JSON Output
After you have completed your reasoning, provide the final answer as a single, well-formed JSON object. This JSON should be the very last part of your response. Do not add any text after the JSON object.

```
{
  ”instruction_evaluation”: {
    ”reasoning”: ”<A concise summary of your reasoning from Step 1 about the instruction’s uniqueness.>”
    ”is_unique”: <true_or_false>,
  },
  ”bbox_evaluation”: {
    ”reasoning”: ”<A concise summary of your reasoning from Step 1 about the bounding box size.>”,
    ”is_appropriately_sized”: <true_or_false>
  }
}
```

Now, please perform this two-step evaluation for the following data.
Grounding Instruction:
<instruction_here>

C.2 Instruction quality refinement

To verify and filter the quality of both the original and the newly generated diverse instructions, we prompted GPT-4.1 to assess whether each instruction uniquely corresponded to a single element in the GUI screenshot. To mitigate potential model hallucinations during this verification process, we visually grounded the task by overlaying the ground-truth annotation directly onto the input image. This filtering stage significantly improved the quality of our instruction set. A manual inspection of 1,542 instructions confirmed this improvement, revealing that the error rate was reduced from over 23% to under 8%, as detailed in Tab. 8, each data was checked by two experenced annotators.

C.3 Data Quality Improvement

To further validate the quality of our data processing pipeline, we adapt an ablation study via SFT. We use about 210k original samples and result in about 180k cleaned samples after our pipeline process, we use them train Qwen2.5-VL-7B. The results shown in Tab. 11 indicate the high quality of our pipeline.

Table 11: Data pipeline ablation study.

Data Pipeline	MMBench-GUI L2	UI-I2E	Showdown	SS.Pro	SS.V2
✗	72.3	63.5	**60.5**	33.0	88.1
✓	**74.3**	**66.3**	60.3	**33.7**	**90.2**

D Implementation Details

D.1 Implementation Details

We employ the state-of-the-art vision-language foundation models Qwen2.5-VL-7B and Qwen2.5-VL-32B as our backbone architectures. The training procedure consists of two stages:

- **SFT Stage** We fine-tune the models on approximately 283k instances for one epoch. For each instance, we randomly select one instruction as the input command, contextualized by four analytical perspectives (i.e., appearance, spatial, function and goal). The target reasoning process is another randomly sampled instruction from the same instance. We use a global batch size of 256 and a learning rate of 5e-6.
- **RL Stage** The GRPO training utilizes 33k instances, expanded to approximately 100k training samples by generating a sample per valid instruction. The prompt excludes analytical perspectives to promote unconstrained reasoning. We train for one epoch with a learning rate of 1e-6 and 8 rollouts. The batch size is set to 256 for the 7B model and 128 for the 32B model.

D.2 Evaluation Metrics

Following prior works (Yang et al., 2025; Liu et al., 2025c; Tang et al., 2025), we evaluate GUI Grounding performance using the point-in-box accuracy. A prediction is considered correct if the predicted coordinate point $p = (x_p, y_p)$ falls within the ground-truth bounding box $b = (x_l, y_l, x_r, y_r)$, where the (x_l, y_l) denotes the top-left corner and (x_r, y_r) represents the bottom-right corner. The accuracy over a test set of size N is formally defined as: Accuracy $= \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(p_i \in b_i)$, where $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 if the condition is true and 0 otherwise.

E Experiment Prompts

E.1 SFT Stage

In the Supervised Fine-Tuning (SFT) stage, we utilized a dataset of approximately 290,000 instances, with the data distribution detailed in Tab. 9. For each training instance, we randomly sampled a single refined instruction to serve as the input, and subsequently, another instruction was randomly selected from the remaining set to function as the reasoning. Each instance was used

Table 12: Performance comparison on the **MMBench-GUI L2** benchmark. We report aggregated accuracy (%) for details. We report aggregated accuracy (%) in detail. We use'-' to denote unavailability, and '*' to denote the results evaluated by us.

Model	Windows		MacOS		Linux		iOS		Android		Web		Avg.
	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	Basic	Adv.	
GPT-4o	1.5	1.1	8.7	4.3	1.1	1.0	5.1	3.3	2.5	1.4	3.2	2.9	2.9
Claude-3.7	1.5	0.7	12.5	7.5	1.1	0.0	13.7	10.6	1.4	1.4	3.2	2.3	4.7
Qwen-Max-VL	43.9	36.8	58.8	56.1	53.9	30.1	77.4	59.1	79.5	70.1	74.8	58.8	58.0
ShowUI-2B	9.2	4.4	24.1	10.4	25.1	11.7	29.0	19.7	17.4	8.7	22.9	12.7	16.0
Qwen2.5-VL-7B	31.4	16.5	31.3	22.0	21.5	12.2	66.6	55.2	35.1	35.2	40.3	32.5	33.9
Qwen2.5-VL-72B	55.7	33.8	49.9	30.1	40.3	20.9	56.1	28.2	55.6	25.4	68.4	45.8	41.8
OS-Atlas-Base-7B	36.9	18.8	44.4	21.7	31.4	13.3	74.8	48.8	69.6	46.8	61.3	35.4	41.4
Aguvis-7B-720P	37.3	21.7	48.1	33.3	33.5	25.0	67.5	65.2	61.0	51.0	61.6	45.5	45.7
UI-TARS-1.5-7B	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-TARS-72B-DPO	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
UGround-V1-7B	66.8	39.0	71.3	48.6	56.5	31.1	92.7	70.9	93.5	71.0	88.7	64.6	65.7
InternVL3-72B	70.1	42.6	75.7	52.3	59.2	41.3	93.6	80.6	92.7	78.6	90.7	65.9	72.2
InfiGUI-G1-7B	82.7	61.8	83.8	63.9	72.3	52.0	94.9	89.4	95.2	85.6	93.5	76.3	80.8
GTA1-7B*	76.8	57.4	80.3	63.9	68.6	53.6	93.9	83.3	96.3	84.5	90.3	74.7	78.5
GTA1-32B*	82.3	66.9	**89.0**	74.0	**73.3**	52.0	96.2	88.2	95.8	88.5	**95.2**	79.9	83.4
UI-Ins-7B	82.7	64.7	87.2	**75.1**	71.7	51.5	94.9	89.7	95.8	89.0	93.2	80.8	83.1
UI-Ins-32B	**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**

for training only once. To better align the training data with the model's in-context learning capabilities, our training prompt provided four predefined analytical perspectives as context. The prompt structure is detailed as follows:

Training Example We provide a SFT training example as following, we mark the **Instruction as Reasoning** in red.

E.2 RL Stage

In the Reinforcement Learning (RL) stage, we trained the model on a total of 32,851 instances, with the data sources and distribution detailed in Tab. 10. To ensure the model performs robustly across all diverse instructions for a given instance, we trained on every remaining refined instruction for each sample. Furthermore, to encourage the model to discover potentially optimal reasoning pathways, we removed the predefined analytical perspectives from the prompt. This allowed the model, already fine-tuned during the SFT stage, to freely explore various thinking formats and strategies.

RL Training Examples We provide a RL training example as following.

SFT Training Example

 # System Prompt: You are a GUI agent. You are given a task and your action history, with screenshots. You need to perform the next action to complete the task.
Output Format
Return a json object with function name and arguments within <tool_call></tool_call> XML tags:

<think>
...
<think>
<tool_call>
{"name": "grounding", "arguments": <args-json-object>}
<tool_call>

<args-json-object> represents the following item of the action space:
Action Space
{"action": "click", "coordinate": [x, y]}

Your task is to accurately locate a UI element based on the user's instructions. The screenshot resolution is height {height} and width {width}.
First, you should carefully examine the screenshot and analyze the user's instructions from one of the following different perspectives:
Appearance Perspective: You should prioritize visually scanning the user interface to find an element that precisely matches the label, text color, background, shape, and its relative position to other elements as described in the appearance-based instruction. Your decision-making will be driven by direct pattern matching of these visual attributes.
Function Perspective: You should identify which UI element is responsible for achieving the specified system state change or executing the described action, such as switching the media display mode. Your decision-making will be guided by understanding the intended effect or utility of interacting with the element.
Spatial Perspective: You should first locate any specified reference points (e.g., another UI element or screen edge) and then precisely pinpoint the target element based on its exact directional and positional relationship within the UI layout. Your decision-making will be based on geometric and relational positioning within the interface.
Goal Perspective: You should identify the UI element or sequence of actions that directly leads to the fulfillment of the user's stated ultimate objective. Your decision-making will be driven by finding the most direct and efficient path to achieve the desired end-state for the user.
You should translate the user's instruction into the corresponding instruction, and then provide the final action.
 # User Instruction:
Click on the CSDN bookmark in the bookmarks bar to access the CSDN website.
 # Assistant:
<think>
I will analyze this instruction from Appearance-Based perspective, the user's instruction can be represented as : Click the bookmark with the red 'C' icon and the label 'CSDN' in the bookmarks bar.
</think>
<tool_call>
{"name":"grounding","arguments":{"action":"click","coordinate":[588,67]}}
</tool_call>

RL Training Example

 # System Prompt: You are a GUI agent. You are given a task and your action history, with screenshots. You need to perform the next action to complete the task.
Output Format
Return a json object with function name and arguments within <tool_call></tool_call> XML tags:

<think>
...
<think>
<tool_call>
{"name": "grounding", "arguments": <args-json-object>}
<tool_call>

<args-json-object> represents the following item of the action space:
Action Space
{"action": "click", "coordinate": [x, y]}

Your task is to accurately locate a UI element based on the user's instructions. The screenshot resolution is height {height} and width {width}.
First, you should carefully examine the screenshot and analyze the user's instructions in <think>...<think> tags and then output the coordinate.
 # User Instruction:
Click on the CSDN bookmark in the bookmarks bar to access the CSDN website.
 # Assistant:
<think>
...
</think>
<tool_call>
...
</tool_call>

F Details of results

F.1 Evaluation details

We present the performance of our models on four benchmarks: MMBench-GUI L2, UI-I2E-Bench, ScreenSpot-Pro, and ScreenSpot-V2. The results demonstrate that both UI-Ins-7B and UI-Ins-32B achieve excellent performance across diverse operating systems and software categories, performing favorably against models of similar parameter counts. Ultimately, UI-Ins-32B establishes a new state-of-the-art. Furthermore, analysis of the MMBench-GUI L2 and UI-I2E-Bench results reveals that our models consistently improve performance across various instruction types. Notably, they exhibit substantial gains on advanced and implicit instructions, which demand a higher level of semantic understanding, significantly outperforming peer models of a similar size.

F.2 Qualitive results

Generalization analysis We performed a detailed classification of the model's reasoning process by first manually defining ten distinct analytical perspectives. We then utilized GPT-4.1 to examine 500 responses generated by UI-Ins-7B on the UI-I2E benchmark based on this taxonomy. As a single response can incorporate multiple reasoning perspectives, the 500 responses ultimately corresponded to 1,950 distinct instances of reasoning. We compiled statistics on these perspectives, the results of which are presented in Tab. 7. The taxonomy can be seen as following:

Taxonomy of Reasoning Perspectives

1. Appearance
Abbreviation: app
Definition: Describes the static visual properties of a UI element, including its color, shape, icon, style, and the literal text it displays.
2. Functionality
Abbreviation: func
Definition: Describes the element's purpose, its action, or what happens when a user interacts with it.
3. Location
Abbreviation: loc
Definition: Describes the element's spatial position on the screen or in the viewport, which can be absolute (e.g., "top-left") or relative to other elements (e.g., "below the title").
4. Intent
Abbreviation: intent
Definition: Describes the high-level user goal or plan that motivates the entire action. It is often the starting point of a reasoning chain.
5. Structural Relationship
Abbreviation: struct
Definition: Describes the element's position within the UI's layout hierarchy (like a DOM tree), emphasizing its parent, child, or sibling relationship to other elements or containers.
6. State
Abbreviation: state
Definition: Describes the current dynamic condition of an element, such as whether it is interactive, active, selected, disabled, or checked.
7. Component Type
Abbreviation: ctype
Definition: Identifies the element as a standard, reusable design pattern or component, rather than just describing its appearance.
8. Sequential Position
Abbreviation: seq
Definition: Describes the element's order or temporal place within a multi-step user task or workflow.
9. Salience
Abbreviation: salience
Definition: Describes the element's degree of visual prominence, which is often determined by its size, contrast, unique styling, or animation.

10. Accessibility
Abbreviation: a11y
Definition: Describes non-visual properties provided for assistive technologies, such as screen readers. This includes ARIA labels, roles, and other accessibility attributes.

Visualization Here we present the grounding results of UI-Ins-32B across various platforms and software applications. As shown in Fig. 7, UI-Ins-32B demonstrates robust performance on diverse platforms.

Failure cases We analyzed the failure cases of our model on the MMBench-GUI benchmark and identified two primary categories of errors. The first category stems from an insufficient understanding of diverse UI layouts, as shown in Fig. 8. The second category involves model hallucinations, as illustrated in Fig. 9.

Table 13: Performance comparison on the **ScreenSpot-Pro** benchmark. We report aggregated accuracy (%) in detail. We use'-' to denote unavailability, and '*' to denote the results evaluated by us.

Model	CAD		Dev.		Creative		Scientific		Office		OS		Avg.
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	
GPT-4o	2.0	0.0	1.3	0.0	1.0	0.0	2.1	0.0	1.1	0.0	0.0	0.0	0.8
Claude Comp. Use	14.5	3.7	22.0	3.9	25.9	3.4	33.9	15.8	30.1	16.3	11.0	4.5	17.1
SeeClick	2.5	0.0	0.6	0.0	1.0	0.0	3.5	0.0	1.1	0.0	2.8	0.0	1.1
Qwen2-VL-7B	0.5	0.0	2.6	0.0	1.5	0.0	6.3	0.0	3.4	1.9	0.9	0.0	1.6
CogAgent-18B	7.1	3.1	14.9	0.7	9.6	0.0	22.2	1.8	13.0	0.0	5.6	0.0	7.7
UI-R1-3B	11.2	6.3	22.7	4.1	27.3	3.5	42.4	11.8	32.2	11.3	13.1	4.5	17.8
ZonUI-3B	31.9	15.6	24.6	6.2	40.9	7.6	54.8	18.1	57.0	26.4	19.6	7.8	28.7
GUI-R1-7B	23.9	6.3	49.4	4.8	38.9	8.4	55.6	11.8	58.7	26.4	42.1	16.9	31.0
UI-TARS-7B	20.8	9.4	58.4	12.4	50.0	9.1	63.9	31.8	63.3	20.8	30.8	16.9	35.7
UI-AGILE-7B	49.2	14.1	64.3	15.2	53.0	9.8	72.9	25.5	75.1	30.2	45.8	20.2	44.0
GUI-G^2-7B	55.8	12.5	68.8	17.2	57.1	15.4	77.1	24.5	74.0	32.7	57.9	21.3	47.5
InfiGUI-G1-7B	57.4	23.4	74.7	24.1	64.6	18.2	80.6	31.8	75.7	39.6	57.0	29.2	51.9
GTA1-7B	53.3	17.2	66.9	20.7	62.6	18.9	76.4	31.8	82.5	50.9	48.6	25.9	50.1
GTA1-32B	43.7	23.4	82.5	**28.3**	69.2	14.7	79.9	31.8	80.8	43.4	**70.1**	32.6	53.6
OpenCUA-7B	-	-	-	-	-	-	-	-	-	-	-	-	50.0
OpenCUA-32B	-	-	-	-	-	-	-	-	-	-	-	-	55.3
SE-GUI-7B	51.3	**42.2**	68.2	19.3	57.6	9.1	75.0	28.2	78.5	43.4	49.5	25.8	43.2
GUI-Actor-7B	-	-	-	-	-	-	-	-	-	-	-	-	44.6
UI-Ins-7B	**60.9**	20.3	75.3	18.6	65.2	**18.9**	81.3	29.1	79.7	37.7	57.0	25.8	52.2
UI-Ins-32B	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**

Table 14: Performance comparison on the **UI-I2E-Bench** benchmark. We report aggregated accuracy (%) in detail. We use'-' to denote unavailability, and '*' to denote the results evaluated by us.

Model	Grouped by Platform			Grouped by Implicitness		Overall
	Web	Desktop	Mobile	Explicit	Implicit	
Qwen2.5-VL-7B	56.9	41.6	61.7	58.4	51.0	53.8
Qwen2.5-VL-72B	49.0	47.2	55.3	49.6	52.5	51.4
OS-Atlas-4B	54.6	19.9	58.6	51.5	39.9	44.3
OS-Atlas-7B	52.2	48.9	68.1	63.2	55.8	58.6
Aguvis-7B	45.1	47.6	60.3	61.1	48.4	53.2
Uground-V1-2B	66.4	49.5	59.9	72.9	47.9	57.4
Uground-V1-7B	70.8	65.7	73.5	81.3	63.6	70.3
Uground-V1-72B	74.7	74.6	78.2	84.5	71.3	76.3
UI-TARS-2B	62.2	54.0	66.7	74.1	54.5	62.0
UI-TARS-7B	56.5	58.0	65.7	71.4	55.3	61.4
UI-TARS-1.5-7B	79.5	68.8	74.1	81.3	68.2	73.2
UI-TARS-72B	77.1	69.8	75.5	80.9	69.4	73.7
UI-I2E-VLM-4B	60.9	38.9	61.4	61.9	48.3	53.4
UI-I2E-VLM-7B	62.1	64.0	76.2	72.0	67.9	69.5
InfiGUI-G1-7B	84.6	66.3	83.0	85.0	72.7	77.4
GTA1-7B*	77.5	71.3	83.5	87.0	72.8	78.2
GTA1-32B*	93.3	77.6	84.4	91.4	78.7	83.5
UI-Ins-7B	90.5	72.8	83.8	88.9	76.3	81.1
UI-Ins-32B	**95.7**	**81.9**	**88.2**	**92.9**	**83.9**	**87.3**

Table 15: Performance comparison on the **ScreenSpot-V2** benchmark. We report aggregated accuracy (%) in detail. We use'-' to denote unavailability, and '*' to denote the results evaluated by us.

Model	Mobile		Desktop		Web		Avg.
	Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
SeeClick	78.4	50.7	70.1	29.3	55.2	32.5	55.1
OS-Atlas-Base-7B	95.2	75.8	90.7	63.6	90.6	77.3	85.1
UI-TARS-7B	96.9	89.1	95.4	85.0	93.6	85.2	91.6
UI-TARS-72B	94.8	86.3	91.2	87.9	91.5	87.7	90.3
GUI-G^2-7B	98.3	**91.9**	95.4	**89.3**	94.0	87.7	93.3
UI-R1-3B	98.2	83.9	94.8	75.0	93.2	83.7	89.5
Qwen2.5-VL-7B	97.6	87.2	90.2	74.2	93.2	81.3	88.8
Qwen2.5-VL-32B	97.9	88.2	98.5	79.3	91.2	86.2	91.3
UGround-v1-7B	83.6	90.5	85.8	86.3	95.5	83.2	87.7
UI-Tars-1.5-7B	92.2	81.5	91.0	84.2	95.5	84.5	89.0
InfiGUI-G1-7B	**99.0**	**91.9**	94.3	82.1	**97.9**	89.2	93.5
GTA1-7B	**99.0**	88.6	94.9	**89.3**	92.3	86.7	92.4
GTA1-32B	98.6	89.1	96.4	86.4	95.7	88.7	93.2
Phi-ground-7B	90.2	76.4	93.6	75.9	96.5	62.0	83.8
OpenCUA-7B	-	-	-	-	-	-	92.3
OpenCUA-32B	-	-	-	-	-	-	93.4
GUI-Actor-7B	97.6	88.2	96.9	85.7	93.2	86.7	92.1
SE-GUI-7B	-	-	-	-	-	-	90.3
LPO	97.9	82.9	95.9	86.4	95.6	84.2	90.5
UI-Ins-7B	**99.0**	90.5	97.9	81.4	97.4	91.6	94.0
UI-Ins-32B	98.6	90.0	**99.0**	87.9	97.0	**93.1**	**94.9**

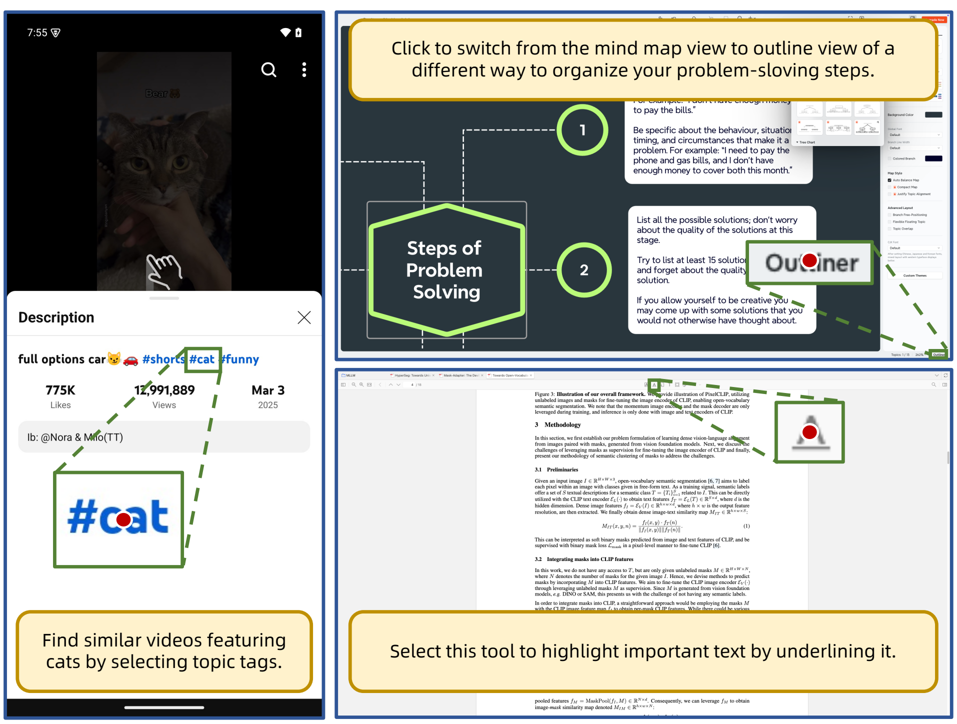


Figure 7: Success Examples of UI-Ins-32B

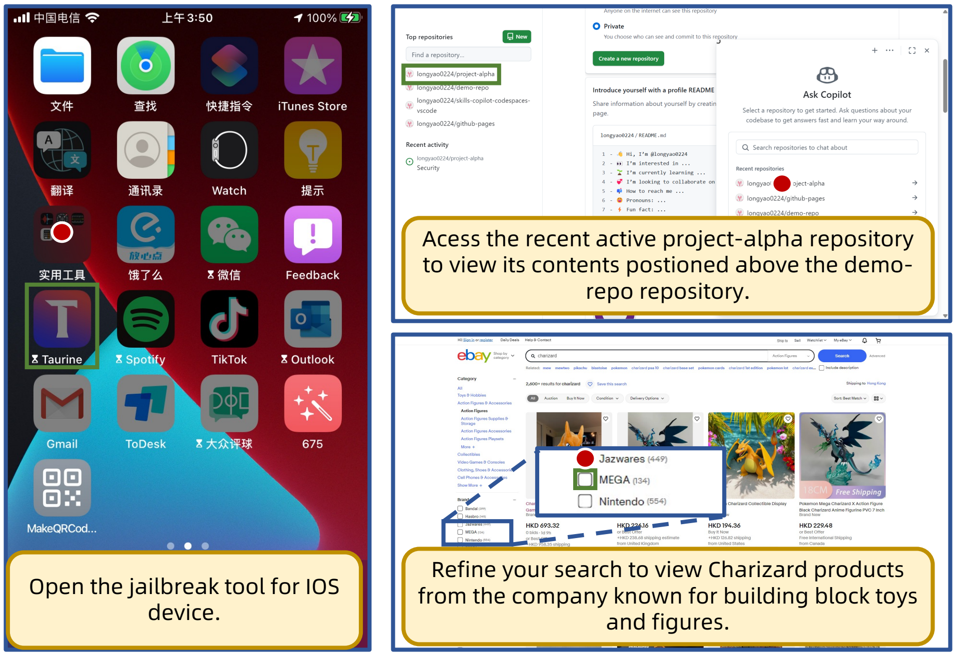


Figure 8: Failure Examples of UI-Ins-32B, these examples need more layout knowledge of corresponding app.

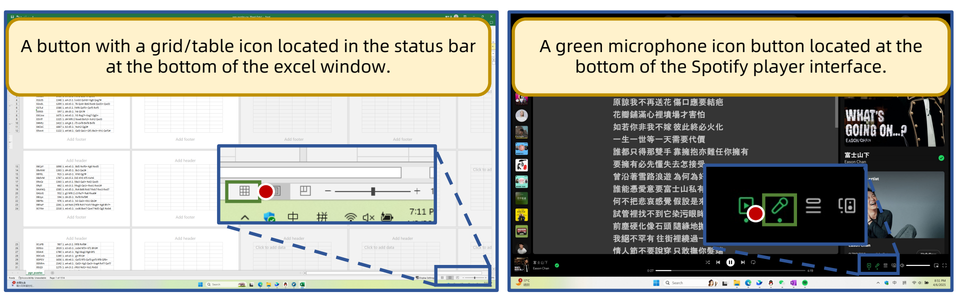


Figure 9: Failure Examples of UI-Ins-32B, these examples casued by hallucination.

Table 16: Performance comparison across different reasoning perspectives. 'App.' is Appearance, 'Func.' is Functionality, 'Spa.' is Spatial, 'Goa.' is Goal, and 'No.' indicates no specific perspective provided.

Perspective	**MMBench-GUI L2**	**UI-I2E**	**Showdown**	**SS.Pro**	**SS.V2**
Appearance (App.)	75.7	69.8	66.4	**37.1**	**91.1**
Functionality (Func.)	75.4	68.6	65.7	35.7	89.7
Spatial (Spa.)	74.7	68.0	66.4	34.5	90.6
Goal (Goa.)	74.7	67.8	65.7	35.8	90.1
No Perspective (No.)	**76.3**	**70.1**	**67.5**	**37.1**	90.6