SCREENSPOT-PRO: GUI GROUNDING FOR PROFESSIONAL HIGH-RESOLUTION COMPUTER USE

Kaixin Li[1] **Ziyang Meng**[2] **Hongzhan Lin**[3] **Ziyang Luo**[3] **Yuchen Tian**[3]
Jing Ma[3] **Zhiyong Huang**[1] **Tat-Seng Chua**[1]
[1]National University of Singapore [2]East China Normal University [3]Hong Kong Baptist University
likaixin@u.nus.edu

ABSTRACT

Multi-modal large language models (MLLMs) are rapidly advancing in visual understanding and reasoning, enhancing GUI agents for tasks such as web browsing and mobile interactions. However, these agents depend on reasoning skills for action planning but only rely on the model capability for UI grounding (localizing the target element). These grounding models struggle with high-resolution displays, small targets, and complex environments. In this work, we introduce a novel method to improve MLLMs' grounding performance in high-resolution, complex UI environments using a visual search approach based on visual reasoning. Additionally, we create a new benchmark, dubbed ScreenSpot-Pro, designed to comprehensively evaluate model capabilities in professional high-resolution settings. This benchmark consists of real-world high-resolution images and expert-annotated tasks from diverse professional domains. Our experiments show that existing GUI grounding models perform poorly on this dataset, with the best achieving only 18.9%, whereas our visual-reasoning strategy significantly improves performance, reaching 48.1% without any additional training.

1 INTRODUCTION

Recent advancements in Multi-modal Large Language Models (MLLMs) (OpenAI, 2024; Wang et al., 2024; Chen et al., 2023; Li et al., 2024) have significantly invigorated this pursuit, driving intensive research efforts in creating pure-vision based GUI agent models that can directly interact with electronic devices that are integral to modern life (You et al., 2024; Hong et al., 2023). These models are capable, to some extent, of directly perceiving device screens in a manner similar to humans, and making decisions on the operations based on the observations.

Many existing studies on GUI agents focus primarily on simple, everyday tasks like general computer control, web browsing, and lifestyle apps. These tasks are typically performed on devices with lower resolution and simpler interface elements, which makes them easier for GUI agents to handle. However, in real-world scenarios, where high-resolution displays and data-rich, complex interfaces are common, GUI agents often struggle. The intricate details, high-definition visuals, and complex layouts of professional applications create significant challenges for these agents, impacting their perception, comprehension, and interaction capabilities. This gap highlights the limitations of current approaches in handling more demanding environments.

The primary challenge of applying current GUI agents to these professional applications is threefold: (1) the significantly greater complexity of professional applications, compared to general-use software, often necessitates the use of higher resolutions that exceed the effective handling capacity of current MLLMs; (2) the increased resolution results in smaller relative target sizes in the screenshot, where GUI grounding models generally exhibit worse performance, as demonstrated in Figure 1; (3) professional users frequently rely on additional documents and external tools to complement their workflows, further complicating the screen. Consequently, even if the GUI agents[1] are able to understand user instructions and the user interfaces in the professional work environment, it is difficult for them to ground the instructions into executable actions in such complex screenshots.

[1]In this work, we use the terms "GUI agent" and "GUI model" interchangeably to refer to the MLLMs, as the primary focus of this work is on the grounding capabilities of these models.

This paper explores the key challenge in GUI grounding in professional high-resolution environments. Given a natural language instruction and a screenshot, the models are asked to ground the instruction to a precise location of the target UI element. We introduce an Visual-Reasoning method to handle GUI grounding in high-resolution environments. This method leverages the natural hierarchies in GUI screenshots, along with the extensive GUI-related knowledge embedded in the MLLM to perform iterative searching within the pixel space of high-resolution screenshots. At each step, it reasons on the user's instructions and predicts the most probable areas, rather than directly locating the target in the screenshot. These areas are then cropped in successive search steps to eliminate irrelevant distractions, until the grounding model is applied to a sufficiently simplified region.

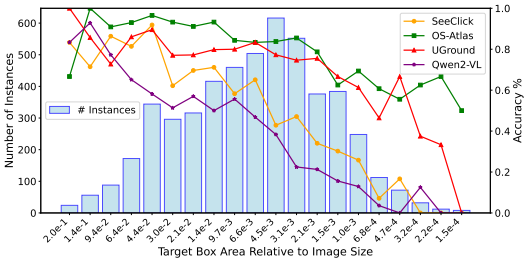

Figure 1: Performance of the expert GUI grounding models SeeClick (Cheng et al., 2024), OS-Atlas (Wu et al., 2024), UGround (Gou et al., 2024), and the generalist MLLM Qwen2-VL (Wang et al., 2024) on the ScreenSpot-v2 GUI grounding benchmark (Wu et al., 2024). The elements on the x-axis are arranged in logarithmically decreasing order, representing their relative size in the entire image. There is a universal decrease in accuracy as the target bounding box size becomes smaller.

Besides, we introduce ScreenSpot-Pro, a new GUI grounding benchmark that includes 23 applications in 5 types of industries, as well as common usages in 3 operating systems. It contains 1,581 instructions, each paired with a unique screenshot, captured by professional users. These tasks are further categorized into ScreenSpot-Pro differentiates itself from previous grounding benchmarks (Cheng et al., 2024; Liu et al., 2024b) in that it includes authentic high-resolution images and tasks captured from a variety of professional applications and domains, thus reflecting the complexity and diversity of real-world scenarios

Our contribution is summarized as follows:

- We propose SeeClick-Pro, a visual reasoning based search method for high-resolution complex scenarios.
- We present ScreenSpot-Pro, a new benchmark for GUI grounding designed to facilitate comprehensive evaluation with authentic tasks collected from various high-resolution professional desktop environments.
- We offer a comprehensive evaluation and comparison of common GUI grounding models.

2 Preliminaries on Visual Search

Several approaches have been proposed to tackle the challenge of processing high-resolution images, including resolution scaling (Chen et al., 2023) and simple cropping (Liu et al., 2024a; Hong et al., 2023). However, these methods struggle to perform effectively at ultra-high resolutions (above 2K) due to inherent model limitations, such as short context lengths and low-resolution training data. For instance, UGround (Gou et al., 2024) supports resolutions up to 1344×1344, while QwenVL (Bai et al., 2023) operates at 448×448. Scaling input resolutions further demands novel model architectures and substantial computational resources for retraining. A promising alternative lies in leveraging visual search techniques.

Humans naturally rely on contextual information to identify likely areas of interest when searching for targets in complex scenes. Inspired by this, the V* algorithm (Wu & Xie, 2023) adopts a top-down search guided by a segmentation module to process high-resolution images. At the core of the framework are a search queue q and a MLLM with two additional decoders: a detection head, which serves as the Target Localization Decoder, and a segmentation head, which functions as a Search Cue Localization Decoder. We denote the MLLM as f_{MLLM}, and the MLLM with the decoder heads as f_{DET} and f_{SEG}. Given a high-res image I and the search target t, the algorithm starts with an empty q, and it initially attempts to locate the target:

$$(d, c) = f_{DET}(I, t) \tag{1}$$

Here, d represents the detection box, and c denotes the associated confidence score. If c is sufficiently high, d is accepted as a result. If not, the algorithm finds where t is likely to appear in the image based on the search cue heatmap h, obtained by segmenting t. The core idea is that, if the segmentation is uncertain, the algorithm retries to segment a new object which is the most likely to co-occur with the target. Specifically, the new target t' is obtained by prompting the MLLM as $t' = f_{MLLM}(I, t)$, if the maximum of h falls below a threshold σ:

$$h = \begin{cases} f_{SEG}(I, t) & \text{if } \max(s) > \sigma \\ f_{SEG}(I, t') & \text{if } \max(s) \leq \sigma \end{cases} \tag{2}$$

Then the image is split into four uniform patches I_1, I_2, I_3, I_4 and enqueued into q for subsequent search within. The priority of the patches are determined by h in the respective areas. Finally, the algorithm takes a patch from q, and iteratively executes the search process. By implementing this recursive partitioning and priority scoring strategy, the V* algorithm effectively handles high-resolution images and complex visual information.

3 SeeClick-Pro: GUI Grounding via Visual Reasoning and Visual Search

Professional applications are designed to provide a comprehensive suite of advanced features, catering to specialized tasks and workflows. These features often involve intricate details, such as high-definition visuals, precise layouts, or data-dense interfaces, which demand a high-resolution display to be fully effective. Unlike natural images, the UI of these applications typically follows a well-defined hierarchy. For example, menus, tools, and properties are often organized within subpanels or child windows, providing clear cues on where to search for a UI target.

Based on the observation, we propose SeeClick-Pro, adopting the idea of visual reasoning and visual search to address the problem of GUI grounding in professional high-resolution computer screens. The algorithm is summarized in Algorithm 1 and an example is visualized in Figure 2.

The proposed algorithm operates through a recursive search process that incrementally refines the localization of the target. Given a text instruction T and an image I, the algorithm begins the search over the entire image and progressively narrows the search area based on inferred positions and visual cues.

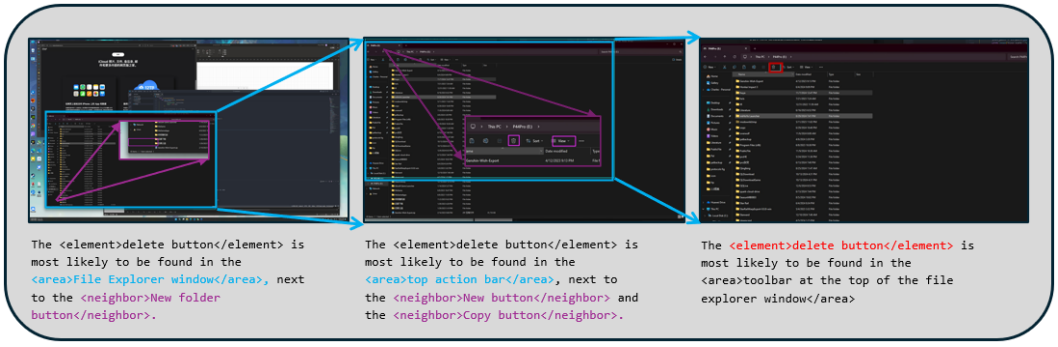

Figure 2: A demonstration of our proposed method. Instruction: "delete file or folder"

Position Inference The core of the algorithm lies in **Position Inference**, where GPT-4 analyzes the instruction T to predict the potential locations of the target. Initially, it identifies the approximate location of the target UI and predicts a series of *areas* that likely enclose the target. It then leverages common knowledge to infer possible *neighboring* UI elements in proximity to the target, such as the "cut" button typically appearing near the "copy" button. This allows the model to generate a set of candidate regions in the image that are most likely to contain the target.

Algorithm 1 V*: LLM-guided Visual Search

Input: Instruction T, Image I_{img}, Max Depth D_{max}, Min Size S_{min}
Output: Target Bounding Box b
function VISUALSEARCH($T, I, D_{\text{max}}, S_{\text{min}}, d$)
    $d \leftarrow 0$, $viewport \leftarrow (0, 0, 1, 1)$
    if $depth \geq D_{\text{max}}$ or I_{img} too small **then**
        return DIRECTGROUNDING($I, viewport$)
    end if
    $candidates \leftarrow$ POSITIONINFERENCE(Y, I)
    $patches \leftarrow$ GROUND($candidates$)
    $dilated\_patches \leftarrow$ DILATE($patches, S_{\text{min}}, R_{\text{max}}$)
    $scores \leftarrow$ SCOREPATCHES($nms\_patches$)
    $nms\_patches \leftarrow$ NMS($dilated\_patches$)
    $sorted\_patches \leftarrow$ SORT($nms\_patches, scores$)
    for each $patch \in sorted\_patches$ **do**
        $sub\_image \leftarrow$ CROPIMAGE($I, patch$)
        $terminate, b \leftarrow$ VISUALSEARCH($T, sub\_image, d + 1$)
        if $terminate$ **then**
            return b
        end if
    end for
    return `None`
end function

Patch Scoring The grounded bounding boxes are often noisy, so we apply box dilation to expand them into larger candidate areas. Subsequently, the candidates are scored based on a Gaussian distribution of the distance between their center points and those of all grounded boxes (including both the target areas and their neighbors):

$$S(x', y', \sigma) = \begin{cases} \exp\left(-\frac{(x'-0.5)^2+(y'-0.5)^2}{2\sigma^2}\right), & \text{if } 0 \leq x' \leq 1 \text{ and } 0 \leq y' \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$x' = \frac{x - x_1}{x_2 - x_1}, \quad y' = \frac{y - y_1}{y_2 - y_1} \tag{4}$$

where (x, y) is the center of a voting box, and (x_1, y_1, x_2, y_2) represent the coordinates of the candidate area. Candidates with more voting boxes closer to their center receive higher scores, while those further away are assigned progressively lower scores. This centrality-based approach emulates human visual attention, and avoids the prioritization of larger candidates, which would otherwise slow down the search process.

Recursive Search The candidates are subjected to non-maximum suppression (NMS) to decrease redundant or overlapping regions, ensuring that the remaining search areas are distinct. The algorithm then recursively searches each candidate patch by cropping out a sub-image, which is passed into the recursive search function, $VisualSearch(I, sub\_image, d + 1)$. The grounder model is invoked if the patch size is sufficiently small (a hyperparameter set to 1280 pixels), and GPT-4o verifies the correctness of the bounding box. This recursive process continues until GPT-4o determines that the target has been found or until the maximum search depth is reached.

4 EXPERIMENTS

4.1 THE SCREENSPOT-PRO DATASET

We create ScreenSpot-Pro, aiming to reflect realistic tasks in real-world challenges across various platforms and applications[2]. To achieve this, it is crucial to capture the authentic workflows of

[2]It is important to note that constructing an interactive environment to distribute similar to OSWorld (Xie et al., 2024) is not feasible due to licensing restrictions.

professionals. We invited a total of 14 experts with at least five years of experience using the relevant applications to record the data. They were instructed to perform their regular work routine to ensure the authenticity of the tasks whenever possible. To minimize disruptions to their workflow, we developed a silently running screen capture tool, accessible through a shortcut key. When activated, this tool takes a screenshot and overlays it on the screen, allowing experts to label the bounding boxes and provide instructions directly. This method enhances the consistency and quality of the annotations, as experts can label tasks in real-time without the need to recall the purposes and context of their actions in hindsight. To obtain authentic high-resolution images, we prioritized screens with a resolution greater than 1080p (1920×1080), a configuration commonly found among annotators. Monitor scaling was disabled. In dual-monitor setups, images were captured to span both displays.

Following SeeClick (Cheng et al., 2024), we also specify the type of the target element, categorizing it as either *text* or *icon*. We refined the classification criteria to better discriminate ambiguous cases where icons are accompanied by text labels, which is common in AutoCAD and Office suites. Specifically, a target is classified as *icon* only when no text hints are present. If text labels are present, the target is labeled as *text*, even if an icon is included.

4.2 Settings and Metrics

We experimented on several VLMs that support GUI Grounding: QwenVL-7B (Bai et al., 2023), Qwen2VL-7B (Wang et al., 2024), MiniCPM-V-2.6 (8B) (Yao et al., 2024), CogAgent (18B) [3] (Hong et al., 2023), SeeClick (7B) (Cheng et al., 2024), UGround (7B) (Gou et al., 2024), OSAtlas-4B, OSAtlas-7B (Wu et al., 2024), ShowUI (2B) (Lin et al., 2024) and Aria-UI (MOE, 3.9B active) (Yang et al., 2024). We precisely evaluate whether the model's predictions align with the annotated ground truth boxes. Formally, a prediction is considered correct if $x_{min} \leq x_i \leq x_{max}$ and $y_{min} \leq y_i \leq y_{max}$, where x_i, y_i are the predictions, and $x_{min}, x_{max}, y_{min}, y_{max}$ is the ground truth box. For models generating bounding box outputs, we calculate the center point as its prediction.

4.3 Baseline Methods

We hypothesize that the main challenge for the models is the large resolution of the screenshots. Therefore, we come up with several intuitive baselines to perform multi-round grounding to shrink the image size for a more accurate prediction.

Iterative Zooming. Inspired by V*'s iterative approach (Wu & Xie, 2023), Iterative Zooming first performs grounding directly on the whole screenshot, and splits the screenshot into smaller patches. At each step, it chooses the patch the prediction falls into to continue searching within. For the splitting strategy, we always use a 2 row $\times$ 2 column split.

Iterative Narrowing. This baseline operates in the same ground-and-zoom procedure as Iterative Zooming, but the patches are cropped to center the prediction. The patch size is set to half the width and height of the image at each step, and the number of iterations is set to 3 to enable a fair comparison with Iterative Zooming. This approach closely aligns with a concurrent work (Nguyen, 2024).

ReGround. We assess a simple baseline that crops the region surrounding the initial prediction to re-ground and make a final determination. The size of the crop can be manually configured based on the optimal input size of the models.

4.4 Results and Findings

Models struggle on ScreenSpot-Pro, even the specialist models. The full results of the GUI grounding models are presented in Table 2. OS-Atlas-7B leads the performance with an accuracy of 18.9%, closely followed by UGround and AriaUI. None of the other models achieved an accuracy above 10%. Notably, GPT-4o, despite its advanced capabilities, scored only 0.9%, highlighting its limitations for the GUI grounding task.

[3] THUDM/cogagent-chat-hf

Table 1: List of software collected in ScreenSpot-Pro.

Icon	Abbr.	Application	Edition & Version	OS	Icons	Texts
		Development and Programming				
	VSC	Visual Studio Code	1.95	macOS	22	33
	PyC	PyCharm	2023.3	macOS	38	40
	AS	Android Studio	2022.2	macOS	44	36
	Qrs	Quartus	II 13.0 SP1	Windows	32	13
	VM	VMware	Fusion 13.6.1	macOS	9	32
		Creative				
	PS	Photoshop	2020	Windows	25	26
	PR	Premiere	2025	Windows	24	28
	AI	Adobe Illustrator	2025	Windows	19	12
	Bl	Blender	4.0.2	Windows	15	56
	FL	FruitLoops Studio	20.8.3	Windows	31	26
	UE	Unreal Engine	5.4.4	Windows	6	29
	DR	DaVinci Resolve	19.0.3	macOS	23	21
		CAD and Engineering				
	CAD	AutoCAD	Mechanical 2019	Windows	7	27
	SW	SolidWorks	Premium 2018 x64	Windows	14	63
	Inv	Inventor	Professional 2019	Windows	11	59
	Vvd	Vivado	2018.3	Windows	32	48
		Scientific and Analytical				
	MAT	MATLAB	R2022b	Windows	19	74
	Org	Origin	2018	Windows	43	19
	Stt	Stata	SE 16	Windows	41	8
	Evw	EViews	10	Windows	7	43
		Office Suite				
	Wrd	Word	Office 365 (16.90)	macOS	15	69
	PPT	PowerPoint	Home and Student 2019	Windows	25	57
	Exc	Excel	Office 365 (16.82)	macOS	13	51
		Operating System Commons				
	Win	Windows	11 Professional	-	47	34
	mac	macOS	Sonoma 14.5	-	23	42
	Lnx	Linux	Ubuntu 24.04	-	19	31

Table 2: Model Performance by Software. The abbreviations used in the table are defined in Table 1.

Model	Development					Creative						CAD						Scientific				Office			OS			Avg
	AS	PyC	VSC	VM	UE	PS	Bl	PR	DR	AI	FL	CAD	SW	Inv	Qrs	Vvd	MAT	Org	Evw	Stt	PPT	Exc	Wrd	Lnx	mac	Win	
OS-Atlas-7B	8.8	15.4	25.5	34.1	22.9	17.6	22.5	17.3	27.3	3.2	10.5	2.9	3.9	2.9	13.3	26.3	23.7	11.3	54.0	12.2	22.0	12.5	44.0	20.0	20.0	12.3	18.9
UGround (7B)	7.5	7.7	21.8	31.7	20.0	21.6	25.4	17.3	11.4	0.0	14.0	2.9	0.0	7.1	15.6	28.7	23.7	6.5	46.0	0.0	25.6	15.6	36.9	18.0	12.3	2.5	16.5
AriaUI (MOE, 3.9B active)	0.0	3.8	21.8	2.4	0.0	27.5	26.8	17.3	2.3	0.0	12.3	0.0	1.3	1.4	20.0	17.5	21.5	1.6	44.0	6.1	6.1	1.6	36.9	2.0	3.1	2.5	11.3
ShowUI (2B)	3.8	7.7	5.5	22.0	11.4	5.9	7.0	5.8	0.0	3.2	3.5	0.0	0.0	1.4	15.6	5.0	8.6	12.9	16.0	6.1	9.8	6.3	22.6	4.0	10.8	4.9	7.7
CogAgent (18B)	2.5	5.1	16.4	9.8	2.9	11.8	7.0	7.7	0.0	0.0	5.3	0.0	1.3	0.0	11.1	18.8	16.1	1.6	34.0	2.0	6.1	0.0	21.4	2.0	4.6	2.5	7.7
OS-Atlas-4B	1.3	1.3	12.7	2.4	0.0	0.0	2.8	1.9	2.3	3.2	5.3	0.0	0.0	1.4	2.2	3.8	7.5	3.2	20.0	0.0	4.9	0.0	8.3	6.0	0.0	3.7	3.7
MiniCPM-V (7B)	0.0	2.6	9.1	2.4	0.0	3.9	0.0	3.8	0.0	0.0	0.0	0.0	0.0	0.0	6.7	11.3	2.2	1.6	18.0	0.0	4.9	0.0	3.6	0.0	3.1	3.7	3.0
Qwen2-VL-7B	0.0	0.0	5.5	0.0	2.9	2.0	0.0	0.0	0.0	0.0	1.8	0.0	0.0	0.0	2.2	1.3	2.2	0.0	12.0	2.0	2.4	0.0	6.0	2.0	0.0	0.0	1.6
SeeClick (7B)	0.0	0.0	0.0	2.4	0.0	0.0	1.4	1.9	0.0	0.0	0.0	2.9	0.0	5.7	0.0	0.0	0.0	0.0	8.0	2.0	0.0	0.0	2.4	2.0	1.5	1.2	1.1
GPT-4o	0.0	1.3	0.0	2.4	2.9	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	2.9	0.0	1.3	2.2	0.0	2.0	0.0	0.0	1.6	1.2	0.0	0.0	0.0	0.8
QwenVL-7B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1

Icons targets are more difficult to ground than texts. Table 3 demonstrates that the benchmarked models struggle significantly in identifying and grounding icon elements in the GUI, a consistent finding with (Cheng et al., 2024). The challenge is exacerbated by the specialization required for professional applications, which introduces several issues: 1) the sheer number of functions makes comprehensive text-based descriptions impractical, e.g. Origin's toolbar (see Figure **??** in the Appendix); 2) these applications often assume users are familiar with the icons and buttons; and 3)

Table 3: Performance breakdown of various models across application categories on ScreenSpot-Pro.

Model	Development			Creative			CAD			Scientific			Office			OS			Avg		
	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg
OSAtlas-7B	33.1	1.4	17.7	28.8	2.8	17.9	12.2	4.7	10.3	37.5	7.3	24.4	33.9	5.7	27.4	27.1	4.5	16.8	28.1	4.0	18.9
UGround (7B)	26.6	2.1	14.7	27.3	2.8	17.0	14.2	1.6	11.1	31.9	2.7	19.3	31.6	11.3	27.0	17.8	0.0	9.7	25.0	2.8	16.5
AriaUI (MOE, 3.9B active)	16.2	0.0	8.4	23.7	2.1	14.7	7.6	1.6	6.1	27.1	6.4	18.1	20.3	1.9	16.1	4.7	0.0	2.6	17.1	2.0	11.3
CogAgent (18B)	14.9	0.7	8.0	9.6	0.0	5.6	7.1	3.1	6.1	22.2	1.8	13.4	13.0	0.0	10.0	5.6	0.0	3.1	12.0	0.8	7.7
ShowUI (2B)	16.9	1.4	9.4	9.1	0.0	5.3	2.5	0.0	1.9	13.2	7.3	10.6	15.3	7.5	13.5	10.3	2.2	6.6	10.8	2.6	7.7
OSAtlas-4B	7.1	0.0	3.7	3.0	1.4	2.3	2.0	0.0	1.5	9.0	5.5	7.5	5.1	3.8	4.8	5.6	0.0	3.1	5.0	1.7	3.7
MiniCPM-V (7B)	7.1	0.0	3.7	2.0	0.0	1.2	4.1	1.6	3.4	8.3	0.0	4.7	2.8	3.8	3.0	3.7	1.1	2.6	4.5	0.7	3.0
Qwen2-VL-7B	2.6	0.0	1.3	1.5	0.0	0.9	0.5	0.0	0.4	6.3	0.0	3.5	3.4	1.9	3.0	0.9	0.0	0.5	2.5	0.2	1.6
SeeClick (7B)	0.6	0.0	0.3	1.0	0.0	0.6	2.5	0.0	1.9	3.5	0.0	2.0	1.1	0.0	0.9	2.8	0.0	1.5	1.8	0.0	1.1
GPT-4o	1.3	0.0	0.7	1.0	0.0	0.6	2.0	0.0	1.5	2.1	0.0	1.2	1.1	0.0	0.9	0.0	0.0	0.0	1.3	0.0	0.8
QwenVL-7B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1

Table 4: Performance of GUI grounding models with Chinese instructions. The abbreviations used in the table are defined in Table 1.

Model	Development					Creative						CAD					Scientific				Office			OS			Avg
	AS	PyC	VSC	VM	UE	PS	Bl	PR	DR	AI	FL	CAD	SW	Inv	Qrs	Vvd	MAT	Org	Evw	Stt	PPT	Exc	Wrd	Lnx	mac	Win	
OS-Atlas-7B	11.3	15.4	21.8	34.1	22.9	11.8	23.9	21.2	11.4	6.5	14.0	5.9	3.9	2.9	8.9	23.8	14.0	11.3	44.0	12.2	17.1	10.9	36.9	16.0	16.9	14.8	16.8
AriaUI (MOE, 3.9B active)	0.0	3.8	18.2	2.4	0.0	23.5	12.7	11.5	0.0	0.0	10.5	0.0	0.0	0.0	13.3	18.8	19.4	1.6	52.0	6.1	2.4	0.0	20.2	2.0	6.2	2.5	9.0
UGround (7B)	3.8	2.6	10.9	14.6	8.6	9.8	11.3	3.8	9.1	3.2	7.0	0.0	0.0	4.3	6.7	12.5	10.8	4.8	30.0	2.0	12.2	4.7	6.0	12.0	7.7	3.7	7.7
ShowUI (2B)	3.8	6.4	5.5	22.0	5.7	7.8	4.2	3.8	0.0	0.0	3.5	5.9	2.6	1.4	15.6	7.5	9.7	11.3	18.0	10.2	9.8	1.6	8.3	4.0	10.8	6.2	7.0
CogAgent (18B)	0.0	5.1	10.9	4.9	0.0	5.9	5.6	5.8	0.0	3.2	3.5	0.0	1.3	0.0	6.7	5.0	7.5	1.6	14.0	2.0	1.2	0.0	2.4	4.0	3.1	2.5	3.7
OS-Atlas-4B	0.0	1.3	7.3	0.0	0.0	2.0	2.8	0.0	4.5	0.0	7.0	5.9	0.0	1.4	0.0	3.8	5.4	4.8	12.0	0.0	4.9	1.6	2.4	4.0	0.0	2.5	2.8
MiniCPM-V (7B)	1.3	2.6	3.6	0.0	0.0	0.0	0.0	1.9	0.0	0.0	3.5	0.0	1.3	0.0	4.4	8.8	0.0	0.0	28.0	0.0	3.7	3.1	0.0	0.0	1.5	2.5	2.5
Qwen2-VL-7B	0.0	0.0	3.6	0.0	0.0	2.0	1.4	3.8	0.0	0.0	1.8	0.0	0.0	0.0	4.4	1.3	2.2	1.6	22.0	6.1	2.4	0.0	2.4	0.0	1.5	0.0	2.0
GPT-4o	2.5	0.0	0.0	0.0	2.9	2.0	1.4	3.8	0.0	0.0	0.0	0.0	2.6	1.4	0.0	0.0	2.2	0.0	2.0	0.0	1.2	0.0	0.0	0.0	1.5	0.0	0.9
SeeClick (7B)	0.0	2.6	0.0	0.0	0.0	0.0	2.8	0.0	0.0	0.0	0.0	0.0	0.0	4.3	0.0	0.0	1.1	0.0	8.0	0.0	1.2	0.0	1.2	0.0	1.5	0.0	0.9
QwenVL-7B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	1.5	0.0	0.2

the icons carry unique meanings within professional contexts that are rarely encountered in the web data, on which many models are primarily trained.

Chinese Instructions Pose Greater Challenges. As shown in Table 4, most models experienced a significant performance drop when switching to Chinese instructions, with the SOTA model OS-Atlas-7B achieving only 16.8%. Among these, UGround-7B saw the most severe decline, dropping from 16.4% to 7.7%, emphasizing its limitations in bilingual contexts. Interestingly, the performance of GPT-4o and QwenVL-7B improved, although this increase appears insignificant given their overall low scores.

Table 5: Comparison of methods on ScreenSpot-Pro with OS-Atlas-7B.

Model	Dev	Creative	CAD	Scientific	Office	OS	Overall Text	Overall Icon	Avg
OS-Atlas-7B	17.7	17.9	10.3	24.4	27.4	16.8	28.1	4.0	18.9
Iterative Focusing	33.1	27.3	23.8	25.2	43.9	36.2	43.5	10.8	31.0
Iterative Narrowing	34.4	27.3	20.3	29.5	40.9	43.9	43.5	13.1	31.9
ReGround	37.5	38.1	33.3	37.8	59.1	37.8	55.7	15.1	40.2
Ours	**49.8** +32.1	**41.9** +24.0	**37.9** +27.6	**47.2** +22.8	**64.3** +36.9	**52.0** +35.2	**64.1** +36.0	**22.4** +18.4	**48.1** +29.2

Simple Shrinking of Image Size achieves impressive result. Interestingly, we found that shrinking the screenshot size strategically demonstrates impressive performance gains. The ReGround method with OS-Atlas-7Bachieved the highest performance, reaching 40.2%. Iterative Narrowing slightly outperformed Iterative Focusing, likely due to its superior image-splitting strategy when the target is positioned near the center of the x or y axes.

Table 6 examines the impact of crop size in ReGround on the two top-performing models, OS-Atlas-7B and UGround. Both models exhibit peak performance within specific resolution ranges, with performance declining as image sizes deviate. OS-Atlas-7B achieves its best score with 1024×1024 crops, while UGround performs optimally with 768×768 crops. This behavior is expected: when images are too

Table 6: ReGround Crop size ablation on ScreenSpot-Pro.

Crop Size	512 × 512	768 × 768	1024 × 1024	1280 × 1280
OS-Atlas-7B	25.1	34.2	**40.2**	40.1
UGround (7B)	27.0	**28.8**	28.2	26.3

small, crucial context is lost (Nguyen, 2024), whereas images that are too large exceed the model's processing capacity.

State-of-the-Art Performance of SeeClick Pro Agent. The proposed SeeClick Pro Agent achieves a state-of-the-art (SOTA) performance of 48.1%, outperforming all other approaches across all subcategories. Notably, the greatest improvements are observed in the Office and OS categories, whereas the Scientific category shows the least improvement. This discrepancy is likely due to the planning model's greater familiarity with commonly used software, which benefits from extensive documentation and user discussions. In contrast, scientific software typically has a more limited user base and fewer available resources, making its analysis more challenging for the model.

5 Conclusion

This paper presents a novel approach to improving the grounding performance of multi-modal large language models (MLLMs) in high-resolution, complex UI environments. By incorporating visual reasoning into the grounding process, we address the challenges posed by high-resolution displays, small targets, and intricate UI structures. The introduction of the ScreenSpot-Pro benchmark, with its expert-annotated tasks from professional domains, provides a comprehensive evaluation of model performance in real-world settings. Our experimental results demonstrate the limitations of existing GUI grounding models, which struggle to achieve high accuracy, and show that our proposed visual-reasoning strategy can substantially enhance performance, achieving a notable improvement without the need for additional training. This work paves the way for more effective and robust GUI agents capable of handling complex, professional UI environments.

References

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 1(2):3, 2023.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. *arXiv preprint arXiv:2312.14238*, 2023.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents, 2024. URL https://arxiv.org/abs/2401.10935.

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, 2024. URL https://arxiv.org/abs/2410.05243.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents, 2023. URL https://arxiv.org/abs/2312.08914.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465*, 2024.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26296–26306, 2024a.

Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *arXiv preprint arXiv:2404.05955*, 2024b.

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

OpenAI. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/, 2024. Accessed: 2024-11-01.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. *arXiv preprint arXiv:2312.14135*, 2023.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*, 2024.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms, 2024. URL https://arxiv.org/abs/2404.05719.