ScreenSpot-Pro: GUI Grounding for Professional High-Resolution Computer Use

Anonymous ACL submission

Abstract

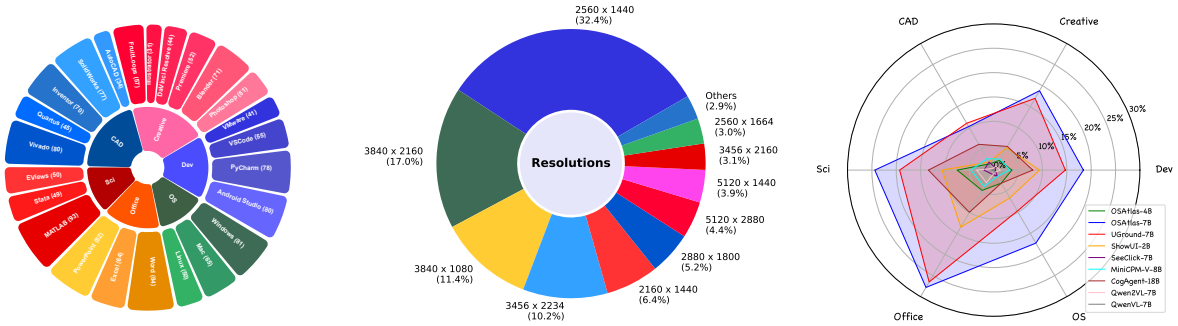
Recent advancements in Multi-modal Large Language Models (MLLMs) have led to significant progress in developing GUI agents for general tasks such as web browsing and mobile phone use. However, their application in professional domains remains under-explored. These specialized workflows introduce unique challenges for GUI perception models, including high-resolution displays, smaller target sizes, and complex environments. In this paper, we introduce **ScreenSpot-Pro**, a new benchmark designed to rigorously evaluate the grounding capabilities of MLLMs in high-resolution professional settings. The benchmark comprises authentic high-resolution images from a variety of professional domains with expert annotations. It spans 23 applications across five industries and three operating systems. Existing GUI grounding models perform poorly on this dataset, with the best model achieving only 18.9%. Our experiments reveal that strategically reducing the search area enhances accuracy. Based on this insight, we propose **ScreenSeekeR**, a visual search method that utilizes the GUI knowledge of a strong planner to guide a cascaded search, achieving state-of-the-art performance with 48.1% without any additional training. We hope that our benchmark and findings will advance the development of GUI agents for professional applications.

1 Introduction

Imagine a future where the everyday burdens of repetitive computer tasks are lifted, unleashing people's full productivity and creativity. A GUI agent capable of taking over the mundane operations of complex professional applications like Visual Studio Code, AutoCAD, Photoshop, could greatly enable computer users to focus exclusively on the work that truly matters. Recent advancements in Multi-modal Large Language Models (MLLMs) (OpenAI, 2024; Wang et al., 2024a; Chen et al., 2023; Li et al., 2024a) 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).

However, many existing studies primarily address general and easy tasks, such as general computer control (Humphreys et al., 2022; Cheng et al., 2024), web browsing (Koh et al., 2024; Deng et al., 2024; Yao et al., 2022), lifestyle and utility apps (Yang et al., 2023; Li et al., 2024b). In contrast, professional applications remain largely unexplored, with only a few works featuring specialized tasks such as coding in VSCode (Xie et al., 2024). These software are designed to provide a comprehensive suite of advanced features, catering to specialized tasks and workflows, and are thus fundamental in productivity and creative industries.

This paper focuses on GUI grounding in professional high-resolution environments: given a natural language instruction and a screenshot, the goal is to ground the instruction to the precise location of the target UI element. The primary challenge of applying GUI grounding models 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 2; (3) professional users frequently rely on additional documents and external tools to complement their workflows, further complicating the screen. Consequently, even if the MLLMs are able to understand user instructions and the user interfaces, it is difficult for them to ground the instructions into precise locations in such complex screenshots.



(a) Software categories and the number of tasks.

(b) Resolution distribution.

(c) Results of representative GUI Grounding models across 6 categories of applications.

Figure 1: Task distribution and benchmark results of ScreenSpot-Pro.

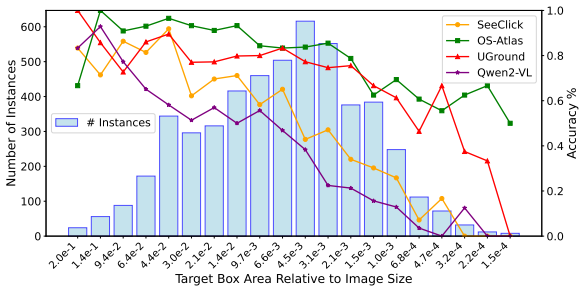


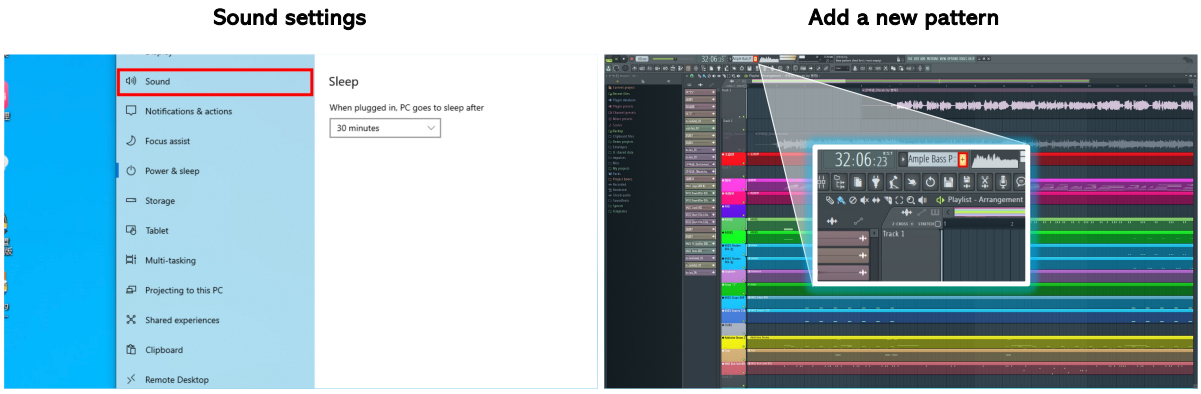
Figure 2: Performance of the expert GUI grounding models SeeClick (Cheng et al., 2024), OS-Atlas (Wu et al., 2024), UGround (Gou et al., 2024), and the generalist MLLM Qwen2-VL (Wang et al., 2024a) on the ScreenSpot-v2 GUI grounding benchmark (Wu et al., 2024). The elements on the x-axis are arranged 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.

However, previous benchmarks such as ScreenSpot (Cheng et al., 2024) neglect these factors and only evaluate GUI grounding in cropped screenshot areas in easy usages, as shown in Figure 3. In this paper, we introduce ScreenSpot-Pro, a novel GUI grounding benchmark that includes 1,581 instructions, each in a unique screenshot. They are sourced from 23 applications in 5 types of industries, as well as common usages in 3 operating systems. ScreenSpot-Pro differentiates itself from previous grounding benchmarks (Cheng et al., 2024; Liu et al., 2024b) in that: i) ScreenSpot-Pro 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; ii) ScreenSpot-Pro is annotated by professional users, ensuring rigorous quality control to maintain the validity of test samples, guaranteeing reliable and meaningful evaluation results.

Through extensive experiments, we found that strategically narrowing the search area within an image leads to significant performance improvements. Building on this insight, we introduce ScreenSeekeR, an agentic framework designed as a baseline approach for GUI grounding in high-resolution environments. ScreenSeekeR leverages the inherent hierarchical structures in GUI screenshots and the rich GUI-related knowledge within the MLLM planner to iteratively refine the search process. Instead of directly identifying the target UI element, it systematically reasons over user instructions to predict the most probable regions. These regions are progressively cropped to remove irrelevant distractions, allowing the grounding model to operate on a simplified subset of the image. With this approach, ScreenSeekeR boosts the OS-Atlas-7B model's performance from 18.9% to 48.1%, achieving a 254% relative improvement.

Our contribution is summarized as follows:

- We present ScreenSpot-Pro, a novel benchmark for GUI grounding with authentic tasks collected from various high-resolution professional desktop environments.
- We identify key challenges in GUI grounding and introduce baseline methods to tackle the difficulties posed by high-resolution image inputs.
- We propose ScreenSeekeR, an agentic framework for adapting existing GUI grounding models to perform visual searches in high-resolution screenshots, achieving state-of-the-art performance.



(a) ScreenSpot

(b) ScreenSpot-Pro (Ours)

Figure 3: ScreenSpot (Cheng et al., 2024) (left) vs ScreenSpot-Pro (right). ScreenSpot-Pro features screenshots of the entire screen, while ScreenSpot contains unrealistic screenshots cropped to local areas. Targets are highlighted in red boxes.

2 Related Works

2.1 GUI Grounding

The aspiration to build autonomous agents that assist humans in daily tasks has long captivated researchers. Recently, Multimodal Large Language Models (OpenAI, 2023, 2024; Wang et al., 2024a) have demonstrated remarkable progress in image understanding and reasoning. These advancements have greatly inspired applications in GUI agents to process both visual and linguistic inputs, allowing them to handle a wider range of tasks (Zhang et al., 2024; Yang et al., 2023; Ziyang et al., 2024; Gur et al., 2024; He et al., 2024). A fundamental aspect of GUI agents is grounding, which translates high-level plans into executable actions located on the screen. Leveraging the capabilities of MLLMs, GUI grounding models (Cheng et al., 2024; Gou et al., 2024; Wu et al., 2024; Yang et al., 2024) are fine-tuned on large-scale text-position pairs extracted from screenshots. This process significantly enhances their ability to align language commands with visual elements, improving the accuracy and effectiveness of GUI agents in real-world applications.

To evaluate GUI grounding abilities, previous benchmarks have primarily focused on simple tasks such as web browsing and mobile interactions. However, these benchmarks oversimplify the problem. For instance, ScreenSpot (Cheng et al., 2024) facilitates artificial screenshots by cropping regions from full-screen images, while VisualWebBench (Liu et al., 2024b) reformulates location prediction as a multiple-choice task by providing candidate targets. Furthermore, these benchmarks overlook the importance of productivity tools in professional settings. To address these limitations, we introduce ScreenSpot-Pro, a benchmark designed to provide a more rigorous evaluation of GUI grounding in high-resolution professional environments. Therefore, in this work, we introduce our benchmark that builds on the idea of ScreenSpot (Cheng et al., 2024) to address tasks in high-resolution professional scenarios.

2.2 Processing High Resolution Images

Though several approaches have been proposed to tackle the challenge of processing high-resolution images in MLLMs, including resolution scaling (Chen et al., 2023) and simple cropping (Liu et al., 2024a; Hong et al., 2023), these methods struggle to perform effectively at ultra-high resolutions 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 $\times$ 1344, while QwenVL (Bai et al., 2023) operates at 448 $\times$ 448. Further increasing input resolutions necessitates innovative model architectures and significant computational resources for retraining. An alternative approach involves utilizing visual search techniques (Wu and Xie, 2023; Wang et al., 2024b). However, these methods depend on predefined splitting strategies, which constrain search flexibility and may result in missing contextual information in GUI environments.

3 ScreenSpot-Pro: Benchmarking GUI Grounding for Professional High-Resolution Computer Use

In this section, we introduce the data collection range, criteria, processing procedure, quality control measures, and provide a statistical overview of ScreenSpot-Pro.

3.1 Scope of Data Collection

ScreenSpot-Pro includes six distinct application genres, with a primary focus on four types of professional applications. Additionally, it features office productivity software and common operating system tasks. These categories include:

- **Development and Programming:** Software for coding, debugging, and testing.
- **Creative Software:** Tools for image, audio, and video production.
- **CAD and Engineering:** Software for 2D/3D design and simulation.
- **Scientific and Analytical:** Tools for data analysis, computation, and simulations.
- **Office Software:** Productivity applications for document processing, data organization, and presentations.
- **Operating System Commons:** General-purpose operating systems for computing, file management, and system utilities.

A detailed list of the collection can be found in Table 1.

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 1: List of software collected in ScreenSpot-Pro.

3.2 Collection Method and Criteria

ScreenSpot-Pro captures realistic tasks in real-world challenges across various platforms and applications. Experts with at least five years of experience using the relevant applications were invited 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 $\times$ 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.

3.3 Quality Control

ScreenSpot-Pro has undergone strict quality control to ensure its high-quality in two notable aspects.

Task Validity. Each instance is reviewed by at least two annotators to ensure correct instructions and target bounding boxes. Ambiguous instructions are resolved, guaranteeing only one target.

Target Box Precision. Annotators precisely verify the interactable regions of the GUI elements, excluding irrelevant areas. This ensures bound-

ing boxes closely match each element for accurate representation and minimal ambiguity.

3.4 ScreenSpot-Pro Statistics

Table 1 summarizes the collected GUI data, encompassing 23 applications across 3 operating systems, offering a level of diversity unmatched by previous benchmarks. The icons constitute 61.8% of the elements, with the remainder being texts. Notably, targets in ScreenSpot-Pro occupy 0.07% of the screenshot area on average, a significant reduction compared to 2.01% of ScreenSpot (Cheng et al., 2024).

4 Methods

The main challenge lies in the large resolution of the screenshots and the small size of the UI targets. Therefore, our aim is to develop strategies that reduce the search space. To achieve more accurate predictions, we propose several intuitive baselines for performing multi-round grounding, which effectively downsize the image.

4.1 Planner-Free Methods

Iterative Zooming. Inspired by V*'s iterative approach (Wu and 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. We always use a 2 row $\times$ 2 column splitting strategy.

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. 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.

4.2 ScreenSeekeR: An Agentic Grounding Framework

Unlike natural images, the UI of applications typically follows a well-defined hierarchy. For example, menus, tools, and properties are often organized within sub-panels or child windows, providing potential cues on where to search for a UI target. Based on the observation, we propose ScreenSeekeR, adopting the idea of visual search to address the problem of GUI grounding in professional high-resolution computer screens.

The core idea behind ScreenSeekeR is to utilize the GUI knowledge of a strong planner (GPT-4o) to generate possible areas to guide the search. 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. First, the planner proposes the most possible areas to search within based on the screenshot. The candidate areas are filtered and scored using the predictions of the grounder model. Then, the planner continues to search recursively or terminate if it thinks the target is found. The algorithm is summarized in Algorithm 1 and an example is visualized in Figure 4.

Position Inference The core of the algorithm lies in Position Inference, where GPT-4o 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. For example, a "new" button typically appears near the "delete" button. This allows the model to generate a set of candidate regions in the image that are likely to contain the target. The prompts can be found in Appendix C.

Candidate Area Scoring The grounded bounding boxes are often noisy, so we apply box dilation to expand smaller ones into larger candidate areas, reducing the risk of missing the target. Next, candidates are ranked based on the sum of their scores across all grounded boxes to determine the search order. Each candidate's score from a given box is computed using a predefined function that considers the distance between their center points:

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

$$x' = \frac{x - x_1}{x_2 - x_1}, \quad y' = \frac{y - y_1}{y_2 - y_1} \tag{2}$$

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. σ is set to 0.3 in all experiments.

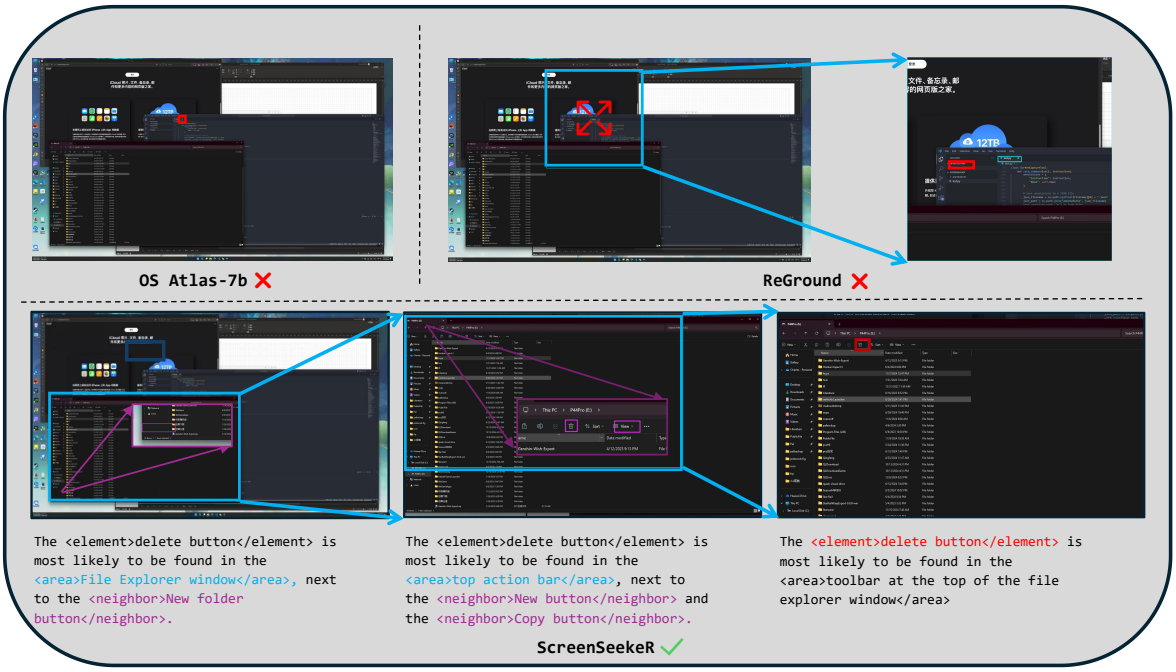


Figure 4: Comparing ScreenSeekeR (bottom) with the plain model prediction (top left) and ReGround (top right). The task is "delete file or folder". Grounding results are marked in the same color as the text references under the screenshots. Final results are drawn in red boxes.

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 mitigates the bias towards large areas, which would otherwise slow down the search process.

The candidates are then subjected to non-maximum suppression (NMS) to decrease overlapping regions. When two boxes overlap greatly, the one with a higher score is kept.

Recursive Search The algorithm recursively searches each candidate area 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 the planner verifies the correctness of the bounding box. This recursive process continues until the planner determines that the target has been found or until the maximum search depth is reached.

5 Experiments

With ScreenSpot-Pro, we rigorously evaluate the correctness whether the model's predictions fall into the annotated ground truth boxes. For models inferencing boxes, we consider the center point of the generated box as the prediction.

End-to-end Models. We conduct the experiments on several MLLMs that support GUI Grounding: QwenVL-7B (Bai et al., 2023), Qwen2VL-7B (Wang et al., 2024a), MiniCPM-V-2.6 (8B) (Yao et al., 2024), CogAgent (18B) [1] (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 (Mixture of Experts, 3.9B active) (Yang et al., 2024). We handle the varying formats of the location outputs to ensure a fair comparison across models.

Multi-round Methods. We compare the four proposed methods, Iterative Zooming, Iterative Narrowing, ReGround and ScreenSeekeR with the same grounding model OS-Atlas-7B. To enable a fair comparison, the number of iterations in Iterative Zooming and Iterative Narrowing are both set to 3 following Nguyen (2024). For ScreenSeekeR, we utilize GPT-4o (OpenAI, 2024) as the planner.

[1] THUDM/cogagent-chat-hf

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

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

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($T, I, D_{\max}, S_{\min}, d$)
  $d \leftarrow 0$, $viewport \leftarrow (0, 0, 1, 1)$
  if $depth \geq D_{\max}$ 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($patches, S_{\min}, R_{\max}$)
  $scores \leftarrow$ 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

5.1 Results of End-to-end models

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.

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 6 in the Appendix); 2) these applications often assume users are familiar with the icons and buttons; and 3) the icons carry unique meanings within professional contexts that are rarely encountered in the web data, on which many models are primarily trained.

5.2 Results of Planner-Free methods

The simplest ReGround Method achieves the best result among planner-free methods. Interestingly, the simplest baseline ReGround achieved the highest performance with OS-Atlas-7B, 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.

Ablations on the crop size of ReGround. Table 5 examines the impact of crop size in ReGround on the two top-performing models, OS-Atlas-7B and UGround (7B). Both models exhibit peak performance within specific resolution ranges, with performance declining as image sizes devi-

Model	Dev	Creative	CAD	Scientific	Office	OS	Overall Text	Overall Icon	Overall 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**
- Recursive Search	40.8	35.5	33.3	44.5	58.7	43.4	51.8	16.2	41.9
- Neighbor Inference	46.8	41.6	33.3	44.9	63.0	**53.6**	62.4	20.4	46.4
- Patch Scoring	48.5	**42.8**	34.1	**47.6**	61.3	50.0	63.3	20.2	46.8
ScreenSeekeR	**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

Table 4: Comparison of methods on ScreenSpot-Pro with OS-Atlas-7B.

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

Table 5: Ablation of crop size in ReGround.

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

5.3 Results of ScreenSeekeR

ScreenSeekeR achieves SOTA on ScreenSpot-Pro. Table 4 demonstrates the superior performance of ScreenSeekeR on ScreenSpot-Pro. While the base model, OS-Atlas-7B, achieves only 18.9% accuracy, our method successfully boosts it to 48.1% without any additional training. Interestingly, although the planner model GPT-4o scores only 0.9% direct grounding accuracy on the benchmark, it powers the entire search process. This suggests that models with strong screenshot understanding, even if not optimized for grounding, can still be leveraged to enhance grounding performance.

ScreenSeekeR generates intuitive and explainable search traces. As shown in the example in Figure 1, given the task of "delete file or folder," ReGround was misled into grounding the file tab in the background VSCode window, as its initial grounding attempt was too far off. In contrast, ScreenSeekeR not only successfully grounds the target UI but also generates a natural search trajectory. It first locates the button in the open Explorer window, then searches the top action bar before identifying the target, closely aligned with a human user's thought process. This feature is not only effective but also essential for interpreting the model, as it provides a clear and understandable explanation of the search process.

Ablation on key designs. To evaluate the impact of each key component, we conducted ablation studies on ScreenSeekeR, with the results summarized in the bottom part of Table 4. Removing subsequent searches and retaining only the first planner decision led to the most significant performance drop, reducing accuracy to 41.9%. When neighbor inference was ablated, limiting the planner to simply identifying the target's location, performance decreased by 1.7%. Additionally, substituting the patch scoring method with a simple majority vote strategy resulted in a performance drop to 46.8%. These results underscore the crucial role each design element plays in the effectiveness of ScreenSeekeR.

6 Conclusion

The growing capabilities of Multi-modal Large Language Models (MLLMs) present new opportunities for GUI automation in professional domains, yet existing models struggle with the unique challenges of high-resolution interfaces. To address this, we introduced ScreenSpot-Pro, a benchmark that rigorously evaluates GUI grounding in complex professional environments. Our evaluation showed that current models perform poorly, highlighting the need for improved strategies. Inspired by our findings, we proposed ScreenSeekeR, a search-based method that enhances accuracy by refining the search space, achieving a substantial performance boost without additional training. Our work underscores the gap in MLLM-driven GUI agents for specialized applications and provides a foundation for future advancements in this domain.

Limitations

This benchmark is specifically designed to evaluate GUI grounding, excluding agent planning and execution tasks like those in OSWorld (Xie et al., 2024). This decision was made to mitigate potential legal risks arising from licensing restrictions associated with these softwares.

References

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

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. 2023. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. *arXiv preprint arXiv:2312.14238*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *Preprint*, arXiv:2401.10935.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for gui agents. *Preprint*, arXiv:2410.05243.

Izzeddin Gur, Hiroki Furuta, Austin V. Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A real-world webagent with planning, long context understanding, and program synthesis. In *ICLR*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

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. 2023. Cogagent: A visual language model for gui agents. *Preprint*, arXiv:2312.08914.

Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. 2022. A data-driven approach for learning to control computers. In *International Conference on Machine Learning*, pages 9466–9482. PMLR.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *ACL*.

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

Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024b. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*.

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

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

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

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

OpenAI. 2023. Gpt-4v(ision) system card. https://cdn.openai.com/papers/GPTV_System_Card.pdf. Accessed: 2024-02-03.

OpenAI. 2024. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/. 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. 2024a. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, and Dacheng Tao. 2024b. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. *arXiv preprint arXiv:2408.15556*.

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

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

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. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

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

Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

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

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *Preprint*, arXiv:2404.05719.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. 2024. Ufo: A ui-focused agent for windows os interaction. *arXiv preprint arXiv:2402.07939*.

Meng Ziyang, Yu Dai, Zezheng Gong, Shaoxiong Guo, Minglong Tang, and Tongquan Wei. 2024. VGA: Vision GUI assistant - minimizing hallucinations through image-centric fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1261–1279, Miami, Florida, USA. Association for Computational Linguistics.

A ScreenSpot-Pro-CN

In the case of professional scenarios, it is common for non-English speakers to operate in both their native languages and English. Consequently, it is essential for GUI agents to efficiently manage tasks that involve switching between languages, while accurately interpreting context and instructions across these languages. To reflect this, every task in the benchmark also includes a Chinese instruction translated by GPT-4 and reviewed by the authors who are fluent in both languages. This allows an assessment of the performance and utility of the GUI agent across different language environments.

Chinese Instructions Pose Greater Challenges. As shown in Table 6, 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.

B More Details on the Collected Software

Development and Programming. Development and programming software supports the entire lifecycle of software development, from writing code to debugging and testing applications. These tools provide integrated environments that enhance productivity and collaboration, offering features like syntax highlighting, version control integration, and debugging tools. The applications in this category include **VSCode** (code editor), **PyCharm** (Python IDE), **Android Studio** (Android app development), and **Quartus** (FPGA programming). Additionally, virtualization is critical for creating scalable computing solutions and managing virtual environments, so we also include **VMware Fusion** (virtual machine management).

Creative Software. Creative software includes applications designed for the creation and editing of visual, audio, and video content. These tools are essential in industries such as graphic design, video production, and music composition, enabling professionals to produce high-quality media for various platforms. The tools in this category include **Photoshop** (image editing), **Premiere** (video editing), **Illustrator** (vector graphic design), **FruitLoops Studio** (music production), **DaVinci Resolve** (color grading and video editing), **Unreal Engine** (game engine and 3D simulation), and **Blender** (3D modeling and animation).

Computer-Aided Design (CAD) and Engineering. CAD and engineering software are used to design and model physical objects and systems. These applications are vital in fields such as engineering, architecture, and product manufacturing, where precision design and simulation are required. They enable professionals to create detailed 2D drawings, 3D models, and simulate the behavior of mechanical structures. The tools in this category include **AutoCAD** (2D/3D design), **SolidWorks** (3D CAD and simulation), **Inventor** (mechanical design), and **Vivado** (circuit design and FPGA programming).

Scientific and Analytical. Scientific and analytical software is designed for data analysis, numerical computation, and mathematical modeling. These applications are indispensable in fields like research, engineering, and data science, providing robust environments for analyzing large datasets, solving complex mathematical problems, and running simulations. The software in this category includes **MATLAB** (numerical computation and algorithm development), **Origin** (data analysis and scientific visualization), **Stata** (statistical analysis), and **EViews** (econometric modeling).

Office Software. Office software includes applications designed to facilitate productivity in tasks such as document creation, data analysis, communication, and presentation. These tools are widely used across various industries to manage workflows and support collaborative environments. Key applications in this category include **Word** (word processing), **Excel** (spreadsheets and data analysis), **PowerPoint** (presentation design).

Operation System Commons. Apart from professional software, ScreenSpot-Pro also includes basic operating system operations to evaluate models in high-res environments. These samples are referred to as Operating System Commons, encompassing the general use and interaction with an OS. These include file management, system utilities, etc., that are fundamental to day-to-day tasks on any OS. For this category, we include **Windows**, **macOS**, and **Linux**.

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

Table 6: Performance of GUI grounding models with Chinese instructions. The abbreviations used in the table are defined in Table 1.

C Prompts

Position Inference Prompt

I want to identify a UI element that best matches my instruction. Please help me determine which region(s) of the screenshot to focus on and list the UI elements that might appear next to the target.
If the target does not exist in the screenshot, please output "No target".
Output Requirements:
1. List the possible regions in descending order of probability.
2. Always make specific, clear and unique references to avoid ambiguity. References such as "Other icons" and "window" are NOT allowed.
3. Use the following XML tags to describe items in the screenshot:
- <element>: Wrap a specific UI element.
- <area>: Describe an area of the UI containing multiple elements.
- <neighbor>: Describe a UI element that may appear around the target.
Example Output:
The <element>shortcut link</element> is most likely to be found in the <area>Settings window</area>, in the <area>tools panel</area>, next to the <neighbor>Search button</neighbor>.
Important Notes:
- The target UI element is guaranteed to be present in the screenshot.
Do not speculate about operations that could change the screenshot.
Instruction:
{instruction}

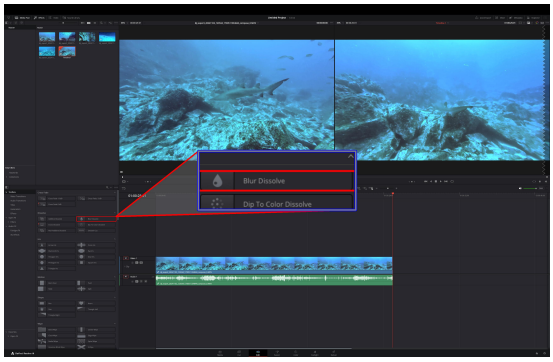
Table 7: Position Inference Prompt

Result Checking Prompt

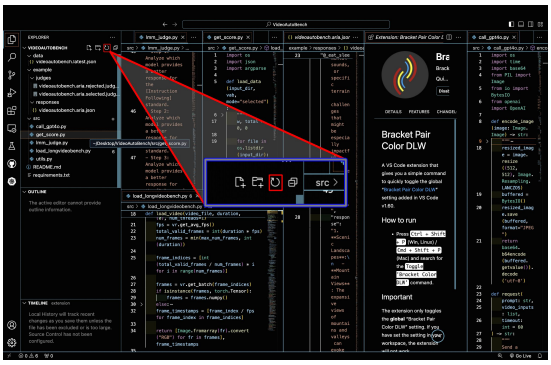
You are given a cropped screenshot. Your task is to evaluate whether the marked element in the red box matches the target described in my instruction.
Please follow these steps:
1. Analyze the screenshot by describing its visible content and functionalities.
2. Determine which of the following applies:
- 'is_target': The marked element is the target.
- 'target_elsewhere': The marked element is not the target, but it exists elsewhere.
- 'target_not_found': The marked element is not the target, and it does not exist.
3. If the target exists, rewrite the instruction to make it clearer.
After your analysis, provide the result in JSON format:
- "result": (str) One of 'is_target', 'target_elsewhere', or 'target_not_found'.
- "new_instruction": (str, default null) A clearer version of the instruction.
Here is my instruction:
{instruction}

Table 8: Result checking Prompt

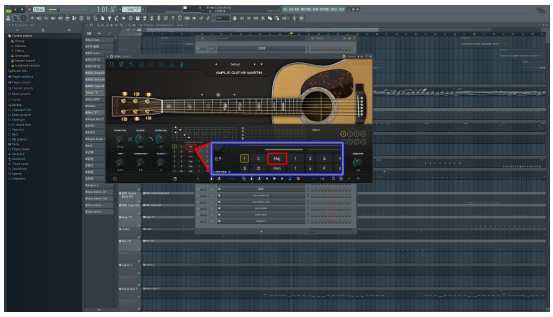
D Data Examples



Instruction: *Blur Dissolve.*
Application: *davinci*
Type: *icon*
Bounding Box: *[460, 1344, 709, 1375]*

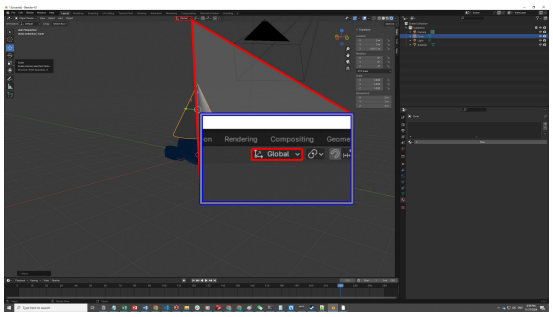


Instruction: *Refresh the file explorer.*
Application: *vscode*
Type: *icon*
Bounding Box: *[473, 183, 503, 219]*

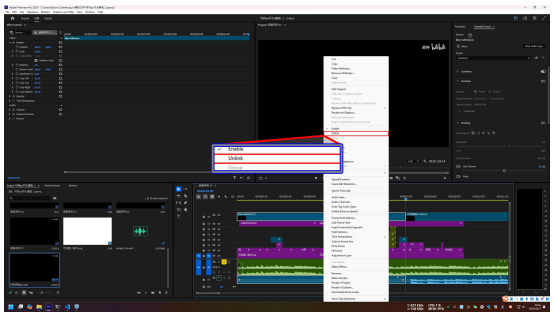


Instruction: *choose chord type for 1.*
Application: *fruitloops*
Type: *text*
Bounding Box: *[853, 652, 897, 677]*

Instruction: *Execute Python scripts.*
Application: *unreal engine*
Type: *text*
Bounding Box: *[246, 2035, 377, 2054]*

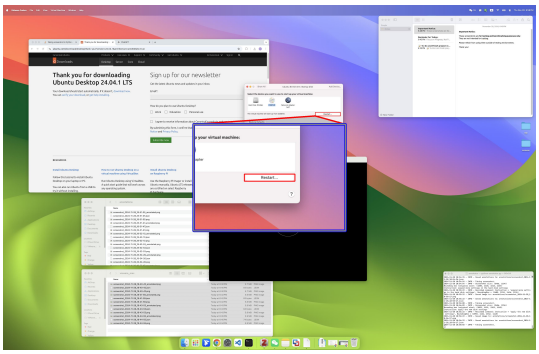


Instruction: *Change the coordinate mode of the object.*
Application: *blender*
Type: *icon*
Bounding Box: *[803, 54, 882, 71]*

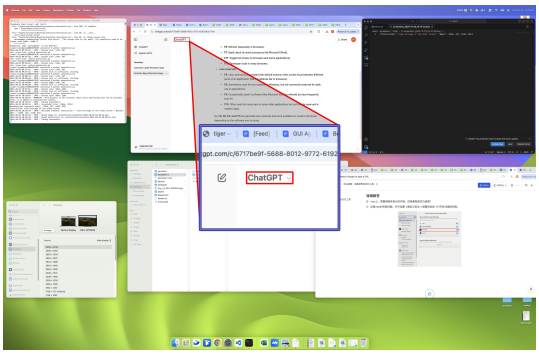


Instruction: *unlink audio and video.*
Application: *premiere*
Type: *text*
Bounding Box: *[1499, 592, 1801, 613]*

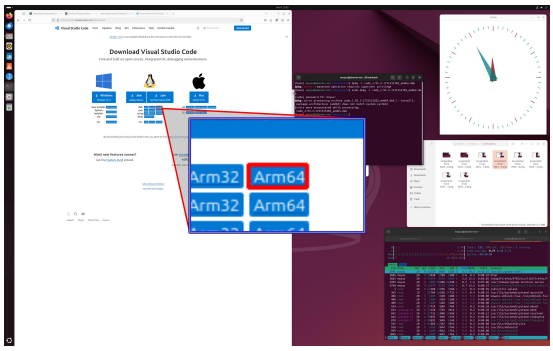
Figure 5: Examples of tasks in ScreenSpot-Pro.



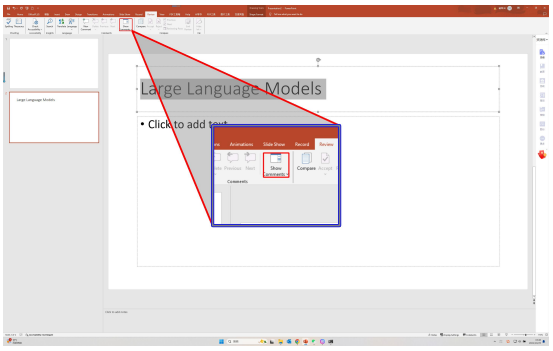
Instruction: *restart from CD.*
Application: *VMWare*
Type: *text*
Bounding Box: *[2024, 695, 2188, 718]*



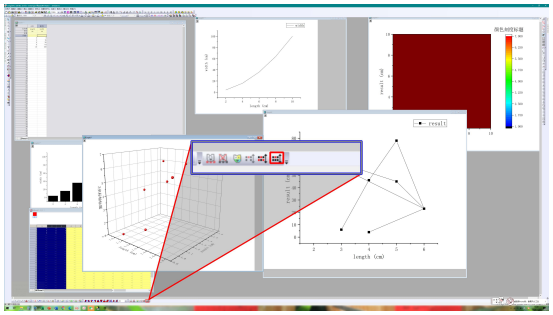
Instruction: *Change model.*
Application: *macOS*
Type: *text*
Bounding Box: *[1109, 211, 1209, 236]*



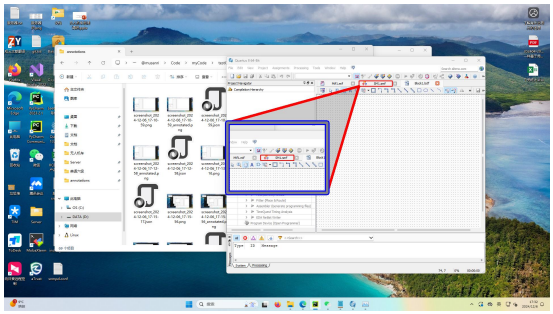
Instruction: *select the correct deb package to download according to the error message in the terminal.*
Application: *linux common*
Type: *text*
Bounding Box: *[960, 639, 1001, 655]*



Instruction: *Show comments.*
Application: *powerpoint*
Type: *text*
Bounding Box: *[614, 72, 681, 136]*



Instruction: *disable masking.*
Application: *origin*
Type: *icon*
Bounding Box: *[998, 2078, 1021, 2097]*



Instruction: *select the SM1.smf file in Quartus window.*
Application: *quartus*
Type: *text*
Bounding Box: *[1248, 270, 1365, 289]*

Figure 6: More examples of tasks in ScreenSpot-Pro.

E Annotator Example

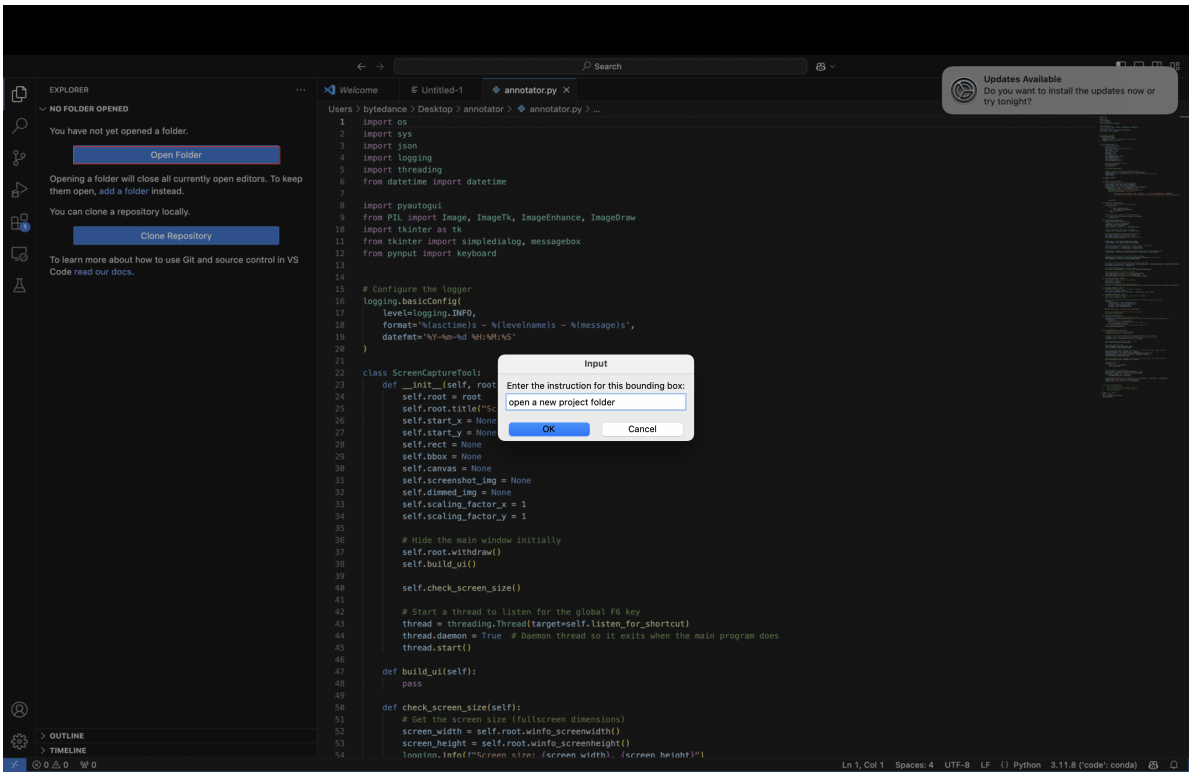


Figure 7: An example of the annotation tool. When activated, the tool captures a screenshot and overlays it on the screen, allowing experts to drag to label the bounding box (the red box around “Open Folder”) and input the instruction in the popup dialog directly.