Navigating the Digital World as Humans Do: UNIVERSAL VISUAL GROUNDING FOR GUI AGENTS

Boyu Gou[1] **Ruohan Wang**[1] **Boyuan Zheng**[1] **Yanan Xie**[2] **Cheng Chang**[2] **Yiheng Shu**[1]
Huan Sun[1] **Yu Su**[1]
[1]The Ohio State University [2]Orby AI
`{gou.43, sun.397, su.809}@osu.edu, yanan@orby.ai`

ABSTRACT

Multimodal large language models (MLLMs) are transforming the capabilities of graphical user interface (GUI) agents, facilitating their transition from controlled simulations to complex, real-world applications across various platforms. However, the effectiveness of these agents hinges significantly on the robustness of their grounding mechanisms. Prevalent GUI agents predominantly utilize text-based inputs such as HTML or accessibility trees, which, despite their utility, often introduce noise, incompleteness, and increased computational overhead. In this paper, we propose SeeAct-V, a generic vision-only framework for building GUI agents. It involves an MLLM as a planner to determine the next action, as well as a grounding model to retrieve the coordinates of target elements from screenshots. We introduce UGround, a universal pixel-level visual grounding model developed specifically for GUIs. This model, trained on 1.3 million diverse samples, is designed to ground open-ended element descriptions directly via pixel coordinates, and to function across different operating systems. Our comprehensive evaluation across six benchmarks, including desktop, mobile, and web platforms, demonstrates that UGround not only outperforms existing visual grounding models, but also matches or exceeds the performance of state-of-the-art methods that rely on HTML or accessibility trees. These results underscore UGround's practicability in significantly advancing the field of vision-based GUI agents, illustrating its ability to navigate digital environments with human-like perception and precision.

1 INTRODUCTION

GUI (graphical user interface) agents, autonomous agents acting in the digital world via operating on GUIs, have been rapidly co-evolving with large language models (LLMs). On the one hand, the general multimedia understanding and production capability of (multimodal) LLMs empower GUI agents to generalize beyond simple simulated settings (Shi et al., 2017; Humphreys et al., 2022) to diverse and complex real-world environments, including the web (Deng et al., 2023; Zhou et al., 2023; Yao et al., 2022), desktop (Xie et al., 2024; Wu et al., 2024) and mobile operating systems (Rawles et al., 2023; 2024; Yan et al., 2023). On the other hand, GUI agents have become one of the most important testbeds for LLMs, providing both the necessary breadth and depth for driving continued development as well as a pathway to many commercially viable automation applications.

Most humans perceive the digital world visually and act via keyboards, mice, or touchscreens. In principle, the embodiment of a GUI agent should already be *complete* if it can 1) visually perceive the GUI renderings, and 2) have effectors equivalent to a keyboard for typing and equivalent to a mouse or touchscreen for pixel-level operations like clicking and hovering.[1] However, current GUI agents assume more than that. For perception, most current agents rely on reading the underlying text-based representations such as HTML or accessibility (a11y) trees (Deng et al., 2023; Gur et al., 2024; Zhou et al., 2023).[2] Only with the recent advances in multimodal LLMs (MLLMs) does visual perception become broadly viable, but text-based representations are still used jointly (Zheng et al., 2024; Koh et al., 2024; Zhang et al., 2024a). For effectors, most current agents act via selecting from

[1] Except for auditory perception, which is out of scope of this study. [2] The a11y tree is a compact yet informative representation intended for assistive technologies to facilitate people with disabilities, e.g., visual impairment.

a list of options, e.g., HTML elements (Deng et al., 2023; Zheng et al., 2024) or labeled bounding boxes (He et al., 2024; Zhang et al., 2024a) instead of pixel-level operations on the GUI. Obtaining those options in turn require access to text-based representations and/or separate models for detecting objects and text (Wang et al., 2024; Kapoor et al., 2024).

However, there is no free lunch, and these additional requirements come with their limitations. On the one hand, *text-based representations are noisy and incomplete*. The full HTML contains a considerable amount of irrelevant information. The a11y tree is more compact and mainly contains semantic information, but similar to other voluntary meta annotations, it widely suffers from incomplete and incorrect annotations.[1] In contrast, visual renderings, by design, are information-complete and only contains information relevant to users. On the other hand, *the additional input increases latency and inference costs*. Zheng et al. (2024) found that HTML can consume up to 10 times more tokens to encode than the corresponding visual. Meanwhile, obtaining the a11y tree can be time-consuming in itself, especially in OS environments. The added latency and cost at every step are further compounded in the long-horizon agent tasks, significantly compromising user experience and practicality.

In this work, we are interested in *how far a human-like vision-only embodiment with visual input and pixel-level operations can go*. There have been a few attempts (Shaw et al., 2023; Hong et al., 2024; Cheng et al., 2024), but they are rarely adopted in state-of-the-art solutions. We find that a major bottleneck is *grounding*, i.e., mapping textual plans generated by an LLM to the precise locations on the GUI. There are three desiderata for GUI agent grounding: 1) *High accuracy*. A single grounding error can get an agent stuck and fail the whole task. 2) *Strong generalization*. It should work on different GUIs: desktop (Windows, Linux, macOS), mobile (Android, iOS), different websites, etc. 3) *Flexibility*. It should plug and play in different MLLMs instead of being tightly coupled with a certain model. Existing visual grounding methods for GUI agents (Shaw et al., 2023; Hong et al., 2024; Cheng et al., 2024) fall short in meeting these desiderata, hindering the advances towards vision-only GUI agents.

The main contributions of this work are three-fold:

1. We make careful arguments and a strong case for human-like vision-only embodiment for GUI agents that perceive the digital world entirely visually and take pixel-level operations on the GUI.

2. Through a series of data and modeling innovations, we develop UGround, a universal pixel-level visual grounding model for GUI agents that is highly accurate and works universally across different GUIs and with different MLLMs.

3. We present the most comprehensive evaluation for GUI agents to date, covering six benchmarks spanning three categories: grounding (desktop/mobile/web), offline agent evaluation (desktop/mobile/web), and online agent evaluation (mobile/web). The results demonstrate: 1) UGround substantially outperforms existing visual grounding models for GUI agents across the board, by up to 28% absolute. 2) With a strong visual grounding model like UGround, GUI agents can achieve on par or better end-to-end performance than state-of-the-art agents that use additional text-based input, clearly demonstrating the promise of GUI agents that navigate the digital world as humans do.

2 Method

2.1 Overview

As shown in Figure 1, we advocate a two-module framework, SeeAct-V to build GUI agents. Similar to previous work (Li et al., 2020a; Zheng et al., 2024), it has: 1) A "planner", which generates a textual description about the target element or area involved in the next step.[2] 2) A grounding model, which takes the textual plan generated by the planner and outputs the precise target element on the GUI to interact with. This two-module framework is universal across all GUI platforms. As mentioned earlier, existing GUI agents leverage text-based representations such as HTML or accessibility trees for either the planner or the grounding module or both and suffer from a number of limitations. In

[1] A 2024 survey over the top one million websites found that 95.9% of the home pages had accessibility conformance errors such as missing alternative text for images or missing form input labels, with an average of 56.8 errors per page (WebAIM, 2024). [2] Here we use "planner" to refer to a *textual* plan/description about the next step, rather than a directly executable plan as in traditional planning (Kambhampati et al., 2024).

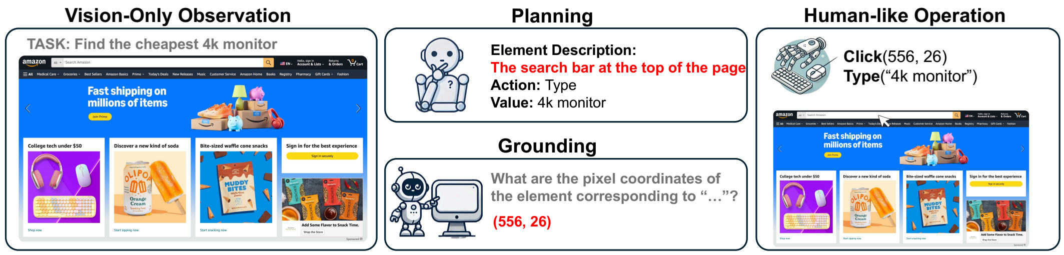

Figure 1: **SeeAct-V Framework.** SeeAct-V is designed to operate using screenshots as the sole input source, without relying on HTML or accessibility trees. It involves an MLLM as a planning model, and a visual grounding model to retrieve coordinates from screenshots. The screenshot and the task are first analyzed by the planner, such as GPT-4, to determine the next action. It is prompted to generate an element description if the action involves an element or a specific area. Based on the description, a grounding model identifies the coordinates of the element from the screenshot. Finally, we performs human-like interactions at the coordinates.

contrast, we advocate for vision-only GUI agents here, which do not require accessibility trees, OCR, or icon detection. With various pretrained MLLMs such as GPT-4V and LlaVA series applicable as planners, our main mission in this work is to develop a universal visual grounding model to empower the second module of SeeAct-V.

Towards this mission, we develop UGround, which addresses critical challenges of visual grounding in GUI agents: handling large variable image sizes and small icons, understanding intricate and noisy UIs, and comprehending diverse referring expressions. These challenges have not been effectively addressed by previous lower-resolution models, OCR or icon detection models, or traditional visual grounding models. We aspire to elevate the grounding performance to a new level of accuracy with UGround and make it a robust universal model for GUI visual grounding.

2.2 Data Construction

To train UGround effectively, we create a comprehensive and diverse dataset that simulates the variability and complexity of real-world GUI referring expressions. Here we first describe our data construction process, especially how to synthesize data from the web, and then introduce the open-source data as well as GPT-generated data to enrich the dataset.

2.2.1 Data Synthesis

To train a good visual grounding model that can generalize across applications, websites, and diverse referring expressions, we need a large and high-quality set of <screenshot, element description, element coordinate> triplets. We begin with synthesizing diverse GUI visual grounding data at a low cost from abundant metadata of webpages.

Previous work has demonstrated the effectiveness of training with metadata of webpage elements (Hong et al., 2024; Cheng et al., 2024). However, their performance still falls short of a desired level for SeeAct-V. GUI referring expressions can be very diverse, which places high demands on the training data. For example, an element can be described in great detail, or simply as "*the button at the top right corner*". Simply training on HTML attributes of the element does not cover the latter description, not to mention even more complex cases. Nevertheless, on the other hand, referring expressions of GUI elements actually have distinct patterns, which can help design better data to fit and generalize to the highly diverse expressions used in practice.

Based on our analysis, three types of attributes are commonly used to refer GUI elements (illustrative examples are shown in Figure 2):

Visual Features. Salient visual features displayed in screenshots, such as text content, image content, element types (e.g., button, input fields, select menu), shapes, colors, etc.

Positions. Absolute and relative positions to other elements. Besides straightforward positional information, contextual references (e.g., "*the 'Add to Bag' button for 'item A'*", "*the link 'text' under the section 'Title'*") often pose more challenges during grounding. To understand this type of referring expressions, the grounding model needs to understand the positional relationship of elements and the their context in the screenshots, which make it impossible to address by a simple OCR or icon detection model.

Functionality. The roles of elements. They are particularly important for non-textual elements, as they are rarely described solely through visual features in real-world use. Here we also interpret app names as roles (which is to open the apps).

We create a generation template to simulate the above listed attributes, which is a random combination of the following fields:

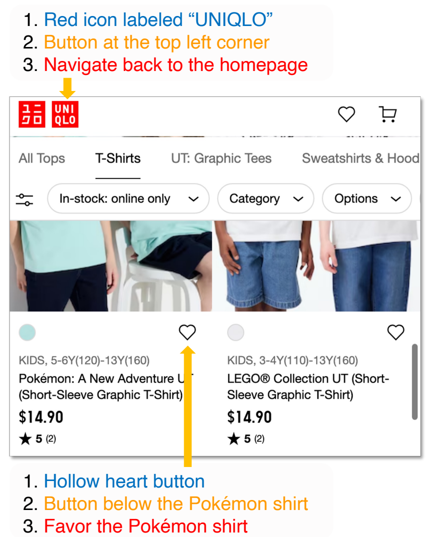

Figure 2: Examples of referring expressions of the three types: visual features, positions, and functionality

Main Description. We use either the visual or functional information of an element as its main description. HTML attributes like `inner-text`, `alt-text` are used as primary visual clues, and accessibility labels like `alt`, `aria-label` are used as roles of elements. However, HTML elements do not always contain high-quality accessibility labels. To address this challenge, we use an open-source MLLM (LLaVA) to uncover visual and functional clues about elements through element screenshots paired with key HTML attributes. Similar to (Lai et al., 2023), we employ an LM (Llama 3) to revise these interpretations into concise element descriptions.

Absolute Position. We generate absolute positional descriptions according to the absolute coordinates of elements in screenshots, by a simple rule to cover most positions.

Relative Position. We first find the neighborhood of elements (for example, the element below or above), and generate corresponding relative position descriptions. Contextual references are more difficult to synthesize, and are always difficult cases in practice. On the other hand, they are very crucial to help the model understand the context of elements, not simply detecting text and icons. We heuristically create textual labels of common and salient elements, including radio buttons, checkboxes, input fields, and select boxes, by their HTML attributes and nearby elements. (for example, a radio button labeled "on"). Some structures of HTML DOM trees (for instance, an element inside of an title element) are also leveraged to create contextual references.

Another challenge we encounter is the mismatch of the correctness and clarity between training and inference. Due to limitations in grounded understanding, even advanced MLLMs like GPT-4o tend to hallucinate on positional descriptions. Hence, it is crucial to teach the model to deal with suboptimal or ambiguous but acceptable descriptions. Otherwise we experimentally find that the model often takes shortcuts by relying heavily on the positional information while neglecting other descriptions, which makes it difficult for the model to generalize to slightly ambiguous or incorrect descriptions provided by MLLMs. We randomly add small portions of suboptimal or wrong positional descriptions to simulate this phenomenon (for example, describing a faraway element as neighborhoods, or randomly change the positional description).

We collect metadata of elements (salient HTML attributes, bounding boxes, screenshots) from Common Crawl[1], and then apply our curated template and rules to the raw data to get the grounding data. Elements on the same webpages are merged to multi-turn conversations of single queries to accelerate training speed by tens of time. We finally use 773K webpages, of which 120K contain elements captioned by LLaVA and Llama 3.

[1] https://commoncrawl.org/

2.2.2 HUMAN AND GPT GENERATED DATA

Despite the rich visual features and functionalities extracted from web raw data and element captions, capturing high-quality visual clues and functional roles remains challenging, particularly for the latter. While the synthetic web data effectively teaches precision and positional descriptions, it still lacks diversity, functionality coverage of other platforms, and nuance of free-form descriptions encountered in real-world applications. To address these gaps, we incorporate human and GPT-4 generated data to cover more diverse and nuanced descriptions not fully represented in our synthetic dataset. Additionally, by including more data from Android, we enhance the model's performance on mobile UI.

First, we gather open-source datasets. We use 5K referring expression data from UIBert (Bai et al., 2021) and 13K from GUIAct-Thoughts (Chen et al., 2024), and 30K functionality data of Android elements from Widget Caption (Li et al., 2020b). We also collect command-style data from AITZ-Thoughts (Zhang et al., 2024b), Android Control-Instructions(Li et al., 2024), and GUIAct-Questions ((Zhang et al., 2024b; Li et al., 2024)), which provide more free-form descriptions and often contain element roles within the commands.

To further expand our dataset, we use GPT-4 to generate 150K descriptions and 257k commands for arbitrary elements from webpages. This includes generating actions or commands corresponding to the elements and detailed descriptions of the elements, adding substantial diversity and complexity to our training data.

We intentionally do not let humans revise the GPT-4 generated data, since in practice, our grounding model needs to ground textual plans generated by MLLMs including GPT-4; this helps simulate the less accurate and suboptimal descriptions in real deployment, ensuring that the model learns to handle descriptions of varied qualities, enhancing its robustness and practical usefulness.

2.3 MODEL DESIGN

We adopt a widely used open-source model architecture, LLaVA-NeXT (Liu et al., 2024), as our backbone model. We make a few adaptations to make it better work for our goals.

Input-Output Formulation. We always let the model anwer "*What are the pixel coordinates of the element corresponding to* {*Description*}" with the screenshot. Following previous work, we represent output coordinates in natural language. Most previous work uses normalized coordinates (percentiles or per thousand). However, while normalization is effective for models with fixed square input sizes, it is less suited for variable-sized image inputs or coordinates requiring higher accuracy. Thus, we opt for natural pixel coordinates (e.g., "*(1344, 1344)*").

Image Resolution. GUI screenshots are significantly larger than natural images, often requiring a resolution of about 1000px for clear screen readability. LLaVA is initially built for 336px images, and later scaled up to at most 772px by `AnyRes` (Cheng et al., 2023; Gao et al., 2024; Liu et al., 2024; Xu et al., 2024), which resizes and splits a large image into small slices, encoding them independently with vision encoders, and adds a special token symbolizing the end of each row to help the LM understand image shapes. To balance the training cost and resolution, we enlarge the allowed input sizes to 36 ViT (Dosovitskiy et al., 2020) slices, and use CLIP@224px (Radford et al., 2021) as the image encoder, pushing the upper bound to 1344*1344 (landscape) and 896*2016 (portrait) during training and inference. We only resize the image by width to preserve the original aspect ratios and pad white strips to the bottom as needed. Additionally, we substitute the language backbone with Vicuna-1.5-7b-16k (Zheng et al., 2023) to handle long visual contexts. We empirically find the low-resolution image fusion commonly used in `AnyRes` to be ineffective in GUI grounding, as 224px is too small to provide informative global context, and hence remove it from the model design.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

To comprehensively evaluate the GUI grounding performance of UGround, and the performance when incorporated with a planner, we evaluate our model on both GUI grounding and GUI agent benchmarks.

We begin with ScreenSpot (Cheng et al., 2024), which evaluates the single-step GUI grounding capability across Web, Mobile and OS (Android, iOS, MacOS, Windows, etc.) We further incorporate UGround into agent frameworks on five agent benchmarks, namely Multimodal-Mind2Web, AndroidControl, OmniACT, Mind2Web-Live, and AndroidWorld, to demonstrate its practicability in real use.

We mainly investigate GPT-4[1] and GPT-4o as planners, as they are demonstrated to be SOTA models on many benchmarks. For grounding, we compare with SeeClick (Cheng et al., 2024), the prior SOTA model on ScreenSpot, as another potential flexible grounding model. We also compare our method to prior prompt-only SOTA methods, either variations of textual choices or SoM, with either text or image as inputs. We following most of prompts of baseline methods, we always replace the input to pure screenshots, and make small changes to incorporate UGroundto baseline prompts and frameworks.

Table 1: Results on ScreenSpot.

Planner	Grounding	Mobile		Desktop		Web		Average
		Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
Original Instr.	GPT-4	22.6	24.5	20.2	11.8	9.2	8.8	16.2
	GPT-4o	20.2	24.9	21.1	23.6	12.2	7.8	18.3
	CogAgent	67.0	24.0	74.2	20.0	70.4	28.6	47.4
	SeeClick	78.0	52.0	72.2	30.0	55.7	32.5	53.4
	UGround	**82.8**	**60.3**	**82.5**	**63.6**	**80.4**	**70.4**	**73.3**
LLaVA-NeXT-72b	SeeClick	74.7	41.1	69.1	27.1	40.9	24.3	46.2
	UGround	84.6	47.2	83.5	50.0	78.3	57.3	66.8
GPT-4	SeeClick	76.6	55.5	68.0	28.6	40.9	23.3	48.8
	UGround	90.1	70.3	87.1	55.7	85.7	64.6	75.6
GPT-4o	SeeClick	81.0	59.8	69.6	33.6	43.9	26.2	52.3
	UGround	**93.4**	**76.9**	**92.8**	**67.9**	**88.7**	**68.9**	**81.4**

3.2 GUI Grounding: ScreenSpot

ScreenSpot is a benchmark designed specifically for GUI grounding, consisting of 1.2K single-step instructions and coordinates of the target elements. It covers many mobile, desktop and web environments on many mainstream platforms, and categorizes element types into text and icons.

We evaluate on two settings, with different MLLMs as planners:

Original Instructions: Most of original labeled queries are of instruction style (for example, "view iPhone storage"), not exactly element description. They are often close to functional referring expressions in our training dataset.

Referring Expression: During online running, we do not have human-labeled ground-truth action instructions. Therefore, we perceive the original instructions as simple one-step tasks, and use different MLLMs as the planner to generate element referring expressions for the elements to interact with, and then evaluate all the grounding models on the referring expressions.

We omit grounding performance of general grounding models trained on natural images except GPT-4 and GPT-4o, since they are already well studied and perform very poorly on ScreenSpot.

3.3 Offline Agent Evaluation

Web Agents: We use Multimodal-Mind2Web (Zheng et al., 2024), the multimodal version of the large-scale web agent dataset Mind2Web (Deng et al., 2023), for our evaluation on web. The SOTA approach SeeAct in (Zheng et al., 2024) also split each step as two steps. MLLMs like GPT-4 firstly generate the action by screenshots, then it select the target elements from either SoM choices or textual HTML elements from top-50 element filtered by another LM trained on Mind2Web.

[1] Due to the deprecation of the name GPT-4V and the frequent abbreviation of GPT-4-Turbo as GPT-4, we refer to GPT-4V, GPT-4-Turbo, and GPT-4 collectively as GPT-4 in this paper. Detailed endpoint names in baselines are provided in the appendix.

Table 2: Element Accuracy on Multimodal-Mind2Web. Models marked with * denote checkpoints finetuned on Mind2Web. The SoM result from (Zheng et al., 2024) is tested on subsets of 30 tasks for each split. 'I' stands for Image, 'T' stands for textual elements, and 'I(R&S)' stands for raw images plus images with Set-of-Marks.

Input	Planner	Grounding	Cross-Task	Cross-Website	Cross-Domain	Avg.
Image	CogAgent*	-	**54.2**	**50.0**	**54.7**	**52.3**
Image	SeeClick*	-	23.8	15.3	16.2	18.4
I + T	GPT-4	Choice	46.4	38.0	42.4	42.3
I(R&S) + T	GPT-4	SoM	29.6	20.1	27.0	25.6
Image	GPT-4	SeeClick	29.6	28.5	30.7	29.6
	GPT-4	UGround	45.1	44.7	44.6	44.8
	GPT-4o	SeeClick	32.1	33.1	33.5	32.9
	GPT-4o	UGround	**47.7**	**46.0**	**46.6**	**46.8**

Similar to prior evaluation in SeeAct, we divide the full webpage screenshots into viewport-sized blocks and simulate scrolling down by heuristically turning empty actions to scrolling. We also add CogAgent and SeeClick-Mind2Web which are trained or finetuned on Mind2Web, to compare with unified visual GUI agents. We report element accuracy here, since that is the most related metric to our grounding comparison.

Table 3: Step Accuracy on AndroidControl.

Input	Planner	Grounding	Step Accuracy	
			High-Level	Low-Level
Acc. Tree	GPT-4	Choice	42.1	55.0
Image	GPT-4	SeeClick	39.4	47.2
	GPT-4	UGround	46.2	58.0
	GPT-4o	SeeClick	41.8	52.8
	GPT-4o	UGround	**48.4**	**62.4**

Mobile Agents: We utilize AndroidControl (Li et al., 2024), a large scale Android interaction dataset comprises 15k unique tasks over 833 apps. Following experiments in Li et al. (2024), we use 500 random steps from the test set. We compare with the SOTA zero-shot method, the text-only version of M3A (Rawles et al., 2024). We also follow the two settings of High-Level tasks and Low-Level tasks, which are to evaluate with the high-level instructions or both high-level and human-labeled low-level command at each step.

OS Agents: We use OmniACT (Kapoor et al., 2024) to evaluate our model's performance on OS tasks. The dataset encompasses a variety of desktop applications and web tasks across different operating systems. It contains 9802 tasks, each containing natural language instructions, UI screens, a list of UI elements (labels and bounding box coordinates), and corresponding PyAutoGUI code scripts.

Table 4: Action Scores (AS) on OmniACT.

Inputs	Planner	Grounding	AS
T	GPT-4	DetACT	11.60
T + I	GPT-4	DetACT	17.02
Image	GPT-4	SeeClick	28.92
	GPT-4	UGround	31.07
	GPT-4o	SeeClick	29.59
	GPT-4o	UGround	**32.77**

The baseline method DetACT uses an OCR module, an icon model, and a color module to extract UI elements and their coordinates, and then pass them to LLMs and MLLMs to generate a sequence of actions in PyAutoGUI codes. We incorporate UGround by replacing input to pure screenshots, and let MLLMs generate element descriptions instead of coordinates when calling PyAutoGUI functions. And then call UGround to locate the coordinates from the screen. Following the method in (Kapoor et al., 2024), we keep the same prompt and five in-context examples retrieved from training set by task similarity. We report the action score, which measures the correctness of action sequences along with penalizing wrong click area, and type or press values.

3.4 Online Agent Evaluation

We further evaluate our model in an end-to-end manner, on two online benchmarks from Android and Web. Generally, without backup grounding methods as a hybrid approach, pure visual grounding can be much harder than SoM or other text-only approach, because at least these methods ensure effective actions at each step. In contrast, a failure of the grounding model can leads to a repeatative clicking to blank area, making agents stuck at a point.

Online Web Agents: We use Mind2Web-Live (Pan et al., 2024) test set as our evaluation on web. It adds functional evaluation metrics to tasks in Mind2Web (Deng et al., 2023). Specifically, key nodes (sub-steps) are annotated to the tasks, to verify the completion of the tasks. The baseline agent is text-only, perceives and interact with webpages by hundreds of textual HTML elements. To completely avoid using HTML, we make the following changes to the action space: 1. We only allow the model to see the current viewport and add *scroll_up* and *scroll_down*. 2. In the baseline agent, an additional judgment model is used to determined whether to press enter after input through HTML[1]. We remove this judgment and change the actions to *type* and *press_enter*, letting the agent to make its own decisions. 3. We disable API-based *select*, which forces agents to select options merely through clicking and makes the action more challenging. We report completion rate and task success rate here, which measure the completion of key nodes, and the full completion of tasks.

Table 5: Completion Rate and Task Success Rate (SR) on Mind2Web-Live.

Inputs	Planner	Grounding	Completion Rate	Task SR
HTML	GPT-4	Choice	44.3	21.1
	GPT-4o	Choice	47.6	22.1
Image	GPT-4	UGround	55.3	**23.1**
	GPT-4o	UGround	**56.0**	19.2

Online Android Agents: We use AndroidWorld (Rawles et al., 2024), an online benchmark running in Android emulators as our online evaluation on Mobile UI. It spreads 116 reproducible tasks across 20 apps, and provides accurate evaluation according to the states of the device. The baseline agent M3A receive both raw and SoM images, together with textual UI elements and element status as input to reason and decide the next action in ReAct-style (Yao et al., 2023). It also combines self-reflection (Shinn et al., 2024) in the pipeline to help agents summarize the current move and facilitate the following steps. The text-only variation of M3A, which use only Android a11y tree, has a higher success rate on Android World, which only use textual elements

Table 6: Task Success Rates (SR) on AndroidWorld.

Input	Planner	Grounding	SR
Acc. Tree	GPT-4	Choice	30.6
I (R&S) + T	GPT-4	SoM	25.4
Image	GPT-4	UGround	31.0
	GPT-4o	UGround	**32.8**

4 Results and Analysis

4.1 Results

Universal GUI Grounding. As shown in Table 1, UGround demonstrates state-of-the-art performance on the ScreenSpot benchmark, surpassing previous GUI grounding models across all platforms. This superiority is particularly pronounced with icon elements in desktop and web user interfaces, where the icons are usually small. UGround also demonstrate flexible acceptance to either original GUI commands, or referring expressions generated by MLLMs on ScreenSpot, both substantially outperform SeeClick by large margins.

Offline Agent Evaluation. As shown in Table 2, Table 3, and Table 4, UGround achieves SOTA performance on the three environments, outperforming prior text-only or SoM baselines with pure raw screenshots. Comparing with unified models, UGround with GPT-4 largely outperforms fnetuned SeeClick-Mind2Web model, but fails to outperform CogAgent on Mind2Web. It shows that with a larger model size and sufficient training, unified models still is possible to work better than UGround. But we do observe that training on Mind2Web limits the flexibility of the models in realistic use, since the models always generate action for a screenshot, never suggesting scrolling down. This hinders realistic use of such finetuned models.

[1] The original agent leverages an LLM to decide between *fill_form* and *fill_search* in the original action space, where the difference is whether to perform a *press_enter* after typing.

Online Agent Evaluation. Online end-to-end evaluation is the hardest setting for all grounding methods, as it assesses the success of whole tasks. However, with UGround as the visual grounding model, we get either higher or comparable performance on Mind2Web-Live and AndroidWorld, as shown in Table 5 and Table 6. Specifically, it outperforms SoM method in AndroidWorld by a large margin, even though Android environments have the least dense UI layout which should be suitable for the SoM method.

4.2 Error Analysis

We investigate action-wise error types in our experiments on ScreenSpot (GPT-4o as the planner), AndroidControl, and Multimodal-Mind2Web, to understand the performance on the three platforms. We sample 60 failure cases from each benchmark (or benchmark sub-split), to analyze whether an error is caused by wrong element descriptions, or grounding failures of UGround. For the convenience of statistics, we skip errors caused by ambiguous or wrong ground truth answer, or an alternative action at the step, only considering the two error types to understand the grounding performance.

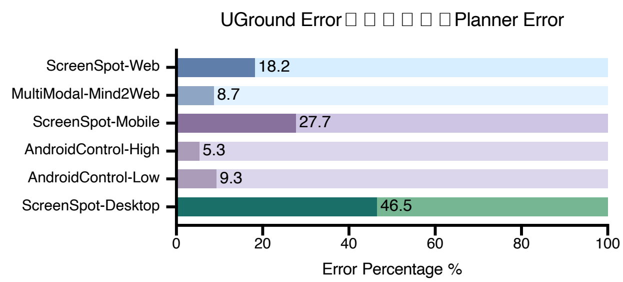

Figure 3: Error analysis of planning and grounding errors on ScreenSpot (with GPT-4o generated referring expressions), AndroidControl and Multimodal-Mind2Web.

As shown in Figure 3, planning errors are the main reasons for the failures on every benchmark. The most common cases are the planner generates element descriptions of other elements, showing poor understanding of the given tasks and the elements in the pages. Some other errors include hallucinate on the screenshot, generating non-existent elements; generating overly generic descriptions, not precisely describing the target area.

In addition, although UGround perform strongly on ScreenSpot-Mobile and ScreenSpot-Desktop, many failures from these two platforms are grounding errors, not planning errors. As discussed in Section 2.2.2, functionalities of elements, especially those of less-frequent elements, are one of main difficulties when building the model. OS has the most icons, where many of them are special to UGround because of a lack of OS data. Hence UGround perform relatively poorly on OS and Mobile on ScreenSpot. However, when running for realistic Android tasks, most of the actions turn out to be not very hard, leading to very few grounding error caused by UGround, which align with the fact that the model works well and still outperform previous SOTA on AndroidWorld.

4.3 Data Analysis

We analyze two key aspects of our training data: the scale of the web synthetic data and the diversity of data sources. Our objective is to understand the impact of various training data types on the performance of GUI visual grounding.

Synthetic Data Scaling. To investigate the effect of our web synthetic data, we train UGround on ramdonly sampled 50K, 100K, 200K, 400K and the entire data, and compare with their performance on ScreenSpot with GPT-4o generated referring expressions. The results on the three platforms and their average are illustrated in Figure 4, as well as the baseline of SeeClick.

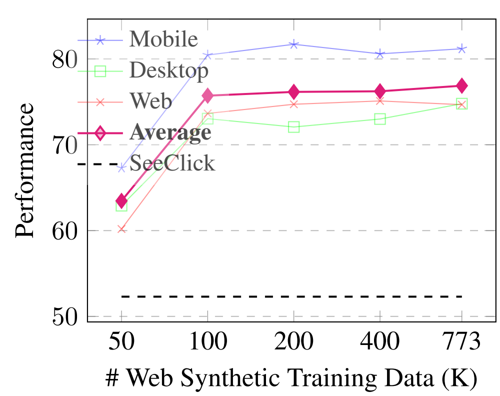

Figure 4: Performance of UGround trained on different data sizes of web synthetic data.

The average performance keeps increasing with data scaling up. But the main improvement happens from 0 to 100K data. And with only 50K webpages of synthetic data, UGround already largely surpass SeeClick by above 10%. To conclude, with only 100K synthetic data, the carefully curated synthetic data is able to teach UGround to understand referring expressions to a remarkable level. We observe that the remaining data mainly improves precision on less frequent element types, like radio, checkbox, very samll text links, etc.

Training Data Sources. To further dissect the influence of training data, we compare the performance of UGround when trained with only web synthetic data, only data generated by humans and GPT-4, and a combination of both (full model). We compare the performance differences on ScreenSpot with GPT-4o generated referring expressions.

As shwon in Table 7. The analysis reveals that while human and GPT-4 data are generally perceived as high-quality inputs, their standalone performance in GUI visual grounding does not surpass that of 100K synthetic data. Specifically, the model trained only on synthetic data (UGround-Synthetic) achieved a performance score of 76.9%, with notably higher scores on icon examples. This highlights the effectiveness of leveraging abundant *ARIA* labels on web, and the effectiveness of our referring expression generation template.

The combination of both data types (UGround) yields the highest performance at 81.4%. This suggests a synergistic effect where the diversity of training examples from both human, GPT-4 and synthetic sources enhances the model's ability to generalize across different GUI elements and actions.

Table 7: Results on ScreenSpot of UGround trained with different data sources.

Model	Mobile		Desktop		Web		Average
	Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
UGround-Synthetic	89.0	73.4	88.1	61.4	84.8	64.6	76.9
UGround-Human-GPT	92.3	71.2	84.5	46.4	87.0	59.2	73.4
UGround	93.4	76.9	92.8	67.9	88.7	68.9	81.4

Acknowledgments

We are grateful for the collaboration with the Orby AI team for their contribution on data collection and analysis, as well as for providing computing resources. We would also like to extend our appreciation to Kanzhi Cheng, Yulu Guo, Lizi Yang and colleagues from the OSU NLP group for their insightful comments. Special thanks to Yichen Pan, Christopher Rawles, Dehan Kong, Alice Li, and Raghav Kapoor for their assistance with evaluation. This work is supported in part by Orby AI and ARL W911NF2220144. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notice herein.

References

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. Uibert: Learning generic multimodal representations for ui understanding. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1705–1712, 2021.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*, 2024.

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

Siyuan Cheng, Bozhong Tian, Qingbin Liu, Xi Chen, Yongheng Wang, Huajun Chen, and Ningyu Zhang. Can we edit multimodal large language models? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13877–13888, 2023.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Peng Gao, Renrui Zhang, Chris Liu, Longtian Qiu, Siyuan Huang, Weifeng Lin, Shitian Zhao, Shijie Geng, Ziyi Lin, Peng Jin, et al. Sphinx-x: Scaling data and parameters for a family of multi-modal large language models. *arXiv preprint arXiv:2402.05935*, 2024.

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

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

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

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

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024.

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. *arXiv preprint arXiv:2402.17553*, 2024.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Zhengfeng Lai, Haotian Zhang, Wentao Wu, Haoping Bai, Aleksei Timofeev, Xianzhi Du, Zhe Gan, Jiulong Shan, Chen-Nee Chuah, Yinfei Yang, et al. From scarcity to efficiency: Improving clip training via visual-enriched captions. *arXiv preprint arXiv:2310.07699*, 2023.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on computer control agents. *arXiv preprint arXiv:2406.03679*, 2024.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile ui action sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8198–8210, 2020a.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5495–5510, 2020b.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL `https://llava-vl.github.io/blog/2024-01-30-llava-next/`.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control. In *Advances in Neural Information Processing Systems*, volume 36, pp. 59708–59728, 2023.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36: 34354–34370, 2023.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pp. 3135–3144. PMLR, 2017.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

WebAIM. The WebAIM Million. `https://webaim.org/projects/million/`, 2024. Accessed: 2024-08-04.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

Ruyi Xu, Yuan Yao, Zonghao Guo, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, Maosong Sun, and Gao Huang. Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images. *arXiv preprint arXiv:2403.11703*, 2024.

An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Qinghong Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian J. McAuley, Jianfeng Gao, Zicheng Liu, and Lijuan Wang. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *ArXiv*, abs/2311.07562, 2023.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

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

Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024b.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2023.