

GROUNDING MULTIMODAL LARGE LANGUAGE MODEL IN GUI WORLD

Anonymous authors

Paper under double-blind review

ABSTRACT

Recent advancements in Multimodal Large Language Models (MLLMs) have accelerated the development of Graphical User Interface (GUI) agents capable of automating complex tasks across digital platforms. However, precise GUI element grounding remains a key challenge for accurate interaction and generalization. In this work, we present an effective GUI grounding framework, which includes an automated data collection engine that gathers extensive GUI screenshots and annotations to ensure broad generalization. We also propose a lightweight and flexible GUI grounding module designed to efficiently localize UI elements by pre-training on the collected data, and introduce a novel method to integrate this module with MLLMs for the effective execution of GUI tasks. Our approach demonstrates superior performance in task accuracy and adaptability, as validated by benchmarks such as ScreenSpot, MiniWob, AITW, and Mind2Web.

1 INTRODUCTION

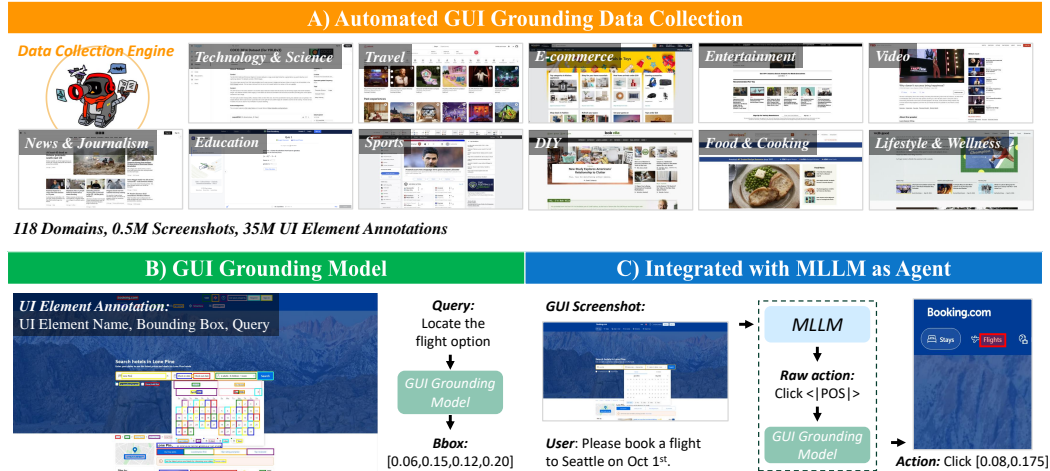


Figure 1: **Overview of our pipeline for building a vision-based GUI agent.** (A) A data collection engine gathers a diverse set of GUI screenshots spanning a broad range of topics. (B) A GUI grounding model trained on the collected data with dense elements and annotations. (C) The grounding model is integrated with MLLMs to form a vision-based agent.

Recent advancements in Large Language Models (LLMs)(Achiam et al., 2023; Touvron et al., 2023a;b; Jiang et al., 2023; Chiang et al., 2023) and Multimodal Large Language Models (MLLMs) (Li et al., 2023; Zhu et al., 2023; Liu et al., 2024; Bavishi et al., 2023; Hong et al., 2023) have significantly improved multimedia comprehension, logical reasoning, and decision-making capabilities, driving the evolution of Graphical User Interface (GUI) agents. These agents are designed to automate complex tasks on digital platforms such as PCs and mobile devices, reducing human effort in performing monotonous duties(Cheng et al., 2024; Gao et al., 2024; Shi et al., 2017; Zhou et al., 2023; Koh et al., 2024; Deng et al., 2023; Chen et al., 2024a;b; Lu et al., 2024a; Yan et al., 2023; Wang et al., 2024; Yang et al., 2023; Shaw et al., 2023).

054 Some pioneering works (Deng et al., 2023; Gur et al., 2023; Zheng et al., 2023; Kim et al., 2024)
055 have designed text-only agents that feed structured plain text representations of screen content, such
056 as HTML and DOM, to LLMs for reasoning, planning, and executing specific GUI actions. However,
057 plain text representations of GUIs have inherent limitations. First, structured text is not always
058 readily available, restricting its use. Second, raw text metadata—often extensive and noisy—requires
059 substantial preprocessing and filtering, which increases computational latency and inference costs
060 and may compromise model accuracy.

061 Humans interact with digital interfaces by visually perceiving content, making decisions, and ex-
062 ecuting actions accordingly. Drawing inspiration from this natural interaction with GUIs, another
063 approach adopted by GUI agents is to use visual renderings as inputs. This approach circumvents
064 the limitations associated with text metadata processing and is better suited for generalization across
065 various platforms. Several pretrained MLLMs (Bavishi et al., 2023; Hong et al., 2023) included UI
066 data during pre-training and have demonstrated their ability to understand basic elements within
067 UI screenshots. A recent work, SeeClick (Cheng et al., 2024), trained a visual GUI agent using
068 screenshots from web and mobile interfaces, enhancing vision-based capabilities in GUI tasks. De-
069 spite these advancements, their performance in GUI automation tasks still falls short compared to
070 text-based models.

071 We identify a primary bottleneck is GUI element grounding, which involves aligning the LLM-
072 generated textual plan with precise screen locations. Robust GUI agents require strong grounding
073 capabilities, characterized by: 1) Precision in Targeting and Interaction: The agent must accurately
074 identify and interact with the necessary elements at each step of a task, ensuring the reliable execution
075 of complex tasks. 2) Generalization Across GUI Scenarios: The agent should effectively generalize
076 across diverse real-world scenarios, such as online shopping, entertainment, and software-based office
077 tasks. 3) Flexible Grounding Adaptation: The grounding model should seamlessly integrate with any
078 LLM planner, enabling easy invocation and adaptation for planning and decision-making.

079 The main challenge of building such a system is developing this capability in the absence of large-scale
080 annotations available on the internet, similar to a relevant topic, segmentation (Kirillov et al., 2023a).
081 Therefore, a carefully tailored approach to data curation and methodical model training is essential.
082 From the data curation, the grounding model should be trained on a comprehensive and diverse set
083 of GUI screenshots and annotations to ensure strong generalization to various GUI applications.
084 Currently, public datasets lack dense, diverse, and richly annotated GUI grounding data. To resolve
085 this, we have created an automated GUI data collection engine to gather relevant screenshots and
086 grounding annotations, as shown in Figure 1 (A). We also perform data cleaning to maintain high
087 quality and augment query expression for different GUI elements to enhance generalizability.

088 From a model perspective, our goal is to develop a lightweight yet powerful grounding model and
089 seamlessly integrate it into an MLLM to perform complex GUI tasks, as shown in Figure 1 (B) &
090 (C). Therefore, we first design a GUI Grounding model with a robust image encoder for generating
091 embeddings and a language encoder for processing queries, combined in a lightweight decoder to
092 predict bounding boxes. This well-trained model is then integrated with the MLLM for downstream
093 tasks like navigation. We add a $\langle |POS| \rangle$ token to the MLLM’s vocabulary, which triggers grounding
094 when needed. Its hidden embedding is passed to the GUI grounding model to locate the corresponding
095 position for operation. This design separates reasoning and grounding into modular components,
utilizing the MLLM for reasoning and the grounding model for precise localization.

096 We evaluated our approach on the ScreenSpot (Cheng et al., 2024) benchmark, demonstrating
097 the efficacy of our GUI grounding pre-training methods. Additionally, we integrated our model
098 with MLLMs to function as a comprehensive GUI agent. This agent has been adapted for various
099 mobile and web-based tasks, including MiniWob (Shi et al., 2017), AITW (Rawles et al., 2024), and
100 Mind2Web (Deng et al., 2023). Our evaluations show significant improvements in task accuracy and
101 adaptability compared to previous methodologies.

102 103 2 RELATED WORK 104

105
106 **Datasets for GUI Understanding.** Several noteworthy benchmarks have been developed as testbeds
107 for GUI agents, each incorporating test data or GUI environments for evaluating the performance of
models. Examples include MiniWob (Shi et al., 2017), MiniWob++Liu et al. (2018), Mind2Web(Deng

et al., 2023), and VisualWebArena Koh et al. (2024), which provide environments for web-based navigation tasks. Additionally, OS-World (Xie et al., 2024) introduces a simulated environment for system operations, while AssistGUI (Gao et al., 2024) focuses on PC software navigation tasks.

In addition to these benchmarks, various publicly available datasets have contributed to training models for UI understanding. For instance, Widget Captioning (Li et al., 2020) offers a dataset that includes mobile UI elements across 88 tasks. RICO (Deka et al., 2017; Sunkara et al., 2022) provides approximately 66k mobile screenshots with view hierarchies, alongside around 500k human annotations that identify various icons by their shapes and semantics. Moreover, datasets such as Pix2Struct (Lee et al., 2023) and Pix2Act (Shaw et al., 2023) have been employed to train models with screen parsing capabilities, leveraging in-house data. Recently, SeeClick (Cheng et al., 2024) has gathered a set of pretraining data to develop an MLLM-based GUI agent.

Nevertheless, these existing datasets are still insufficient in terms of diversity and scale to support the development of a robust GUI grounding model (Lu et al., 2024b). To address this limitation, we have designed an automated data collection engine to assemble a dataset for GUI element grounding, characterized by its high diversity and dense annotations.

Multimodal LLM for GUI Understanding. Multimodal Large Language Models (MLLMs) (Liu et al., 2024; Zhu et al., 2023; OpenAI, 2023b; Abdin et al., 2024; Bavishi et al., 2023) have recently made significant advancements, integrating visual perception and language generation to perform tasks such as image captioning (Li et al., 2023; Wang et al., 2020), visual question answering (Hudson & Manning, 2019), and particularly in OCR-free image text comprehension (Luo et al., 2024). The Qwen-VL series (Bai et al., 2023b;a) adds grounding capabilities, allowing models to localize image regions based on language input. LISA (Lai et al., 2024) adds segmentation capabilities to MLLMs by introducing an additional segmentation modules to MLLMs.

These newly added capabilities enable models to better observe the details of images, which supports tasks in GUI environments. As a result, some models (Hong et al., 2023; You et al., 2024; Shaw et al., 2024; Cheng et al., 2024) have been proposed to generate actions in these environments. GUI-based tasks, often requiring the understanding of high-resolution images, are addressed by models like (Cheng et al., 2024; Hong et al., 2023), enabling interaction with GUIs through visual grounding, thus improving accessibility and widget localization.

While existing models exhibit strong object grounding capabilities, accurately localizing GUI elements in screenshots remains challenging due to their typically small size and similar visual features, such as icons. To overcome this, we designed a specialized decoder, inspired by LISA (Lai et al., 2024), dedicated to grounding GUI elements. Unlike previous approaches that rely solely on pure Transformer architectures to predict bounding boxes for GUI elements (Cheng et al., 2024; Hong et al., 2023; Bavishi et al., 2023), our method addresses the unique challenges posed by GUI screenshots.

3 DATA COLLECTION ENGINE

A robust GUI grounding model means strong performance across diverse GUI applications. To achieve this, it is essential to collect a large, high-quality, and varied dataset. The most effective way to gather such a comprehensive dataset is through web screenshots, given their vast quantity and diversity in layouts and elements, which can be generalized to various application scenarios.

Therefore, we propose a data collection engine that automates the browsing of websites and collects high-quality annotations of GUI elements. The collected data includes *screenshots of graphical user interfaces, the corresponding GUI elements with basic details (including names and bounding boxes), and the hierarchical information of these elements*. The data we aim to collect through this engine is characterized by:

- **Dense Annotations:** Screens should have comprehensive annotations of elements, including but not limited to types, displayed content, and positional information.
- **Rich Elements:** This includes actionable elements, such as various buttons or controls, as well as static elements like embedded images and static text.
- **Diverse Domains:** The data should cover a broad range of styles and layouts, spanning multiple domains, to ensure versatility and the ability to generalize to various application scenarios.

To achieve these goals, our data engine is structured around the following stages:

Dataset	Task	Platform	Vis. Input	# Inst.	# Anno.	Anno. type
MiniWob (Shi et al., 2017)	Navigation	Web	Image	2.8K	9.8K	traj., pos.
Mind2Web (Deng et al., 2024)	Navigation	Web	(multi.) Image	2.3K	17K	traj., pos.
AITW (Rawles et al., 2023)	Navigation	Mobile	(multi.) Image	4.6K	23.6K	traj., pos.
AITZ (Zhang et al., 2024)	Navigation	Mobile	(multi.) Image	2.5K	18.6K	traj., pos.
Odyssey (Lu et al., 2024a)	Navigation	Mobile	(multi.) Image	7.7K	119K	traj., pos.
GUIWord (Chen et al., 2024a)	Understanding	Diverse	Video	12.4K	98K	cap., QA
ScreenSpot (Cheng et al., 2024)	Grounding	Web, Desk, Mobile	Image	1.3K	1.3K	type, cont., pos.
GUIAct (Chen et al., 2024b)	Navigation	Web, Mobile	(multi.) Image	79K	191K	type, cont., pos.
Screen2words (Wang et al., 2021)	Understanding	Mobile	Image	79K	79K	cap.,type,cont.,class,pos,hier.
GUIChat (Chen et al., 2024b)	Understanding	Web, Mobile	Image	50K	50K	chat, pos.
RICO (Deka et al., 2017)	Grounding	Mobile	Image	243K	763K	type, cont., pos.
GUIEnv (Chen et al., 2024b)	Grounding	Web, Mobile	Image	70K	589K	type, cont., pos.
AMEX (Chai et al., 2024)	Grounding	Mobile	Image	97K	885K	type, cont., pos.
SeeClick (Cheng et al., 2024)	Grounding	Web	Image	0.3M	3.0M	type, cont., pos.
Ours	Grounding	Web	Image	0.5M	35M	type,cont.,class,pos,hier.

Table 1: **Statistics for GUI related datasets.** The upper half datasets of smaller size are used for downstream evaluation. The lower half datasets of larger size can be used for pretraining. Understanding type includes chat and/or captioning data. “traj.”: trajectory, “pos.”: position, “cap.”: captioning, “QA”: question answer, “cont.”: displayed content, “class.”: class type defined by system, “hier.”: hierarchy info. Among these datasets, our collected data provides the most screenshots and element annotations.

Topic Selection. We follow previous work (Cheng et al., 2024; Hong et al., 2023), sourcing URLs from the Common Crawl. However, many web pages in Common Crawl only contain plain texts with simple layouts, such as pages consisting of only a few lines of text, a few links, or simple images, thus offering few interactive elements. This simplicity creates a significant gap when compared to the complex GUI structures in common GUI tasks. To collect a large number of interactive elements, we selected 11 topics and identified 118 domains related to those topics. These domains were then used to filter URLs from the Common Crawl dataset. The selection of topics and website domains was guided by GPT-4 (Achiam et al., 2023), which proposed a diverse set of topics relevant to real-world applications. This domain list then extracts relevant web page instances from the original Common Crawl dataset. In the Appendix, we display the selected topics and their associated domains.

GUI Data Collection. Given the URLs for the selected websites, we built a system that automatically loads these web pages in a web browser, such as Chrome. Since web pages are often lengthy, our system simulates user interaction by pressing the “Page Down” key to scroll through the entire content until the bottom of the page is reached. During this process, we capture screenshots of the web pages along with corresponding element data. The element data, which includes attributes, class names, display contents, and bounding boxes, is obtained using UI Automation (UIA). This approach allows us to comprehensively capture both the visual and structural aspects of the user interface, ensuring that we accurately document and analyze the interactive elements present on each webpage.

Data Cleaning. The goal of the data cleaning stage is to ensure the quality and usability of the collected data. Following the gathering of screenshots and element data from various websites, we implement several critical cleaning steps to filter out unusable or irrelevant data, resulting in a high-quality dataset. First, we eliminate websites that are either empty or inaccessible as many URLs may lead to pages with no meaningful content, broken links, or require authentication that cannot be handled automatically. This step is crucial for maintaining the reliability of our dataset. Next, we clean the element data by removing entries with erroneous or missing element names. We also address cases where the element hierarchy is ambiguous or misrepresented by restructuring it to accurately reflect the parent-child relationships between GUI elements. By performing these steps, we refine the dataset to ensure it contains accurate, meaningful, and well-structured data, which is essential for further processing and the effective training of robust GUI grounding models.

Augmenting GUI Grounding Queries. To ensure the model’s generalization capabilities, it must adapt to various query types and handle a range of tasks such as element identification, grounding, and referring expression. However, the metadata collected primarily provides basic information like class names and display contents, which are insufficient for generating diverse queries.

We address this by generating instructions through a set of defined templates that guide the model to focus on specific aspects of the GUI data. For example, a button labeled “go back” could be augmented to a query like “Where is the button to go back to the previous page?” These templates, created using GPT-4 after categorizing the collected elements, enable us to generate a diverse set of

queries that encompass a wide range of interactions and contexts. Some examples of constructed templates are shown in Appendix. This augmentation process ensures that the model can effectively manage various tasks, improving its ability to generalize across different application scenarios and enhancing its practical usability and robustness.

Data Overview. In total, we collected 0.5M screenshots with 35M UI element annotations. Our collected data covers a broad range of topics and features rich elements with dense annotations. We provide a comparison of our collected dataset with previous GUI data in Table 1.

4 MODEL ARCHITECTURE

Within our scope, the GUI grounding model should exhibit generalized capabilities and be easily adaptable for seamless integration with current MLLMs, requiring minimal modifications and ensuring minimal performance loss.

We propose AssistGUIGrounder (AGG), a GUI grounding model designed with these objectives in mind. For the GUI grounding model itself, it takes a GUI image and a textual query about the target element as inputs and outputs the position of the target element corresponding to the query. In terms of integrating the GUI grounding model with an MLLM, it should excel in vision-based GUI agent tasks. The inputs include a GUI image representing the current status, a text description of the user goal, and optionally, a history of past actions. The MLLM, acting as a “planner”, generates the representation of the action and target element for the next step. Meanwhile, the GUI grounding model extracts visual features from the input image and combines them with the decision information from the MLLM, thus outputting the position of the target element when required.

In the following sections, we will outline the model architecture of the GUI grounding model and discuss how it can be effectively integrated with MLLMs.

4.1 GUI GROUNDING MODEL

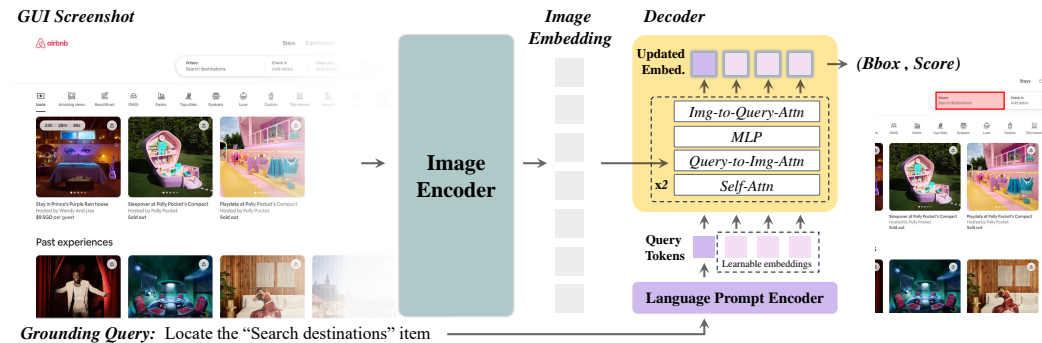


Figure 2: **Overview of GUI Grounding model.** The model consists of three main components: an image encoder, a language prompt encoder, and a lightweight decoder. The image encoder extracts visual features from the input GUI image, while the language prompt encoder transforms the textual query into query embeddings. The lightweight decoder then combines these visual features and query embeddings to predict the position of the target GUI element.

Our GUI grounding model AGG comprises an image encoder I , a language prompt encoder L , and a lightweight decoder D , as illustrated in Fig. 2. As an overview, the image encoder extracts the visual features of a GUI image X_{img} , while the language prompt encoder learns to represent the textual query X_{query} . The lightweight decoder then fuses these visual features and the textual query representation, learning to predict the position of the target GUI element, denoted as \hat{b} . This can be formulated as: $\hat{b} = D(I(X_{img}), L(X_{query}))$. Below is a detailed description of these components:

Image Encoder. We leverage the scalable pretraining benefits by employing the SAM (Kirillov et al., 2023b) pretrained Vision Transformer (ViT)(Dosovitskiy et al., 2021). This choice is made due to its demonstrated effectiveness in promptable point-to-mask segmentation tasks, showcasing its strong visual perception capabilities. For handling high-resolution GUI inputs, we utilize windowed attention

along with four equally spaced global attention blocks, following the methodology in (Kirillov et al., 2023b; Li et al., 2022).

Language Prompt Encoder. The language encoder is applied to maps text queries into query embeddings. Specifically, we employ a language transformer model (Devlin et al., 2019) as the prompt encoder. We prepend $\langle \text{CLS} \rangle$ to each text query and use its output hidden states from the final transformer layer as the query embedding.

Lightweight decoder. The decoder’s role is to process the embeddings from both the image encoder and the language prompt encoder, mapping them to the output bounding boxes which indicate the precise locations of GUI elements. Our decoder architecture is inspired by the design principles outlined in SAM (Kirillov et al., 2023b). Before inputting the embeddings into the decoder, we introduce a set of learnable embeddings and concatenate them with the query embeddings from the prompt encoder. These combined embeddings are collectively referred to as “query tokens”.

The decoder comprises two layers, each performing a series of operations: (1) Bi-directional self-attention on the query tokens, allowing the model to convey information within the tokens; (2) Cross-attention from query tokens (as *query*) to the image embedding; (3) An MLP updates the query tokens; (4) Cross-attention from the image embedding (as *query*) to the query tokens, which updates the image embeddings with contextual information derived from the queries. The next decoder layer then takes the updated query tokens and the updated image embeddings from the previous layer. Finally, the output of the query tokens from the last decoder layer is used for grounding prediction.

Training for Grounding. For the GUI image input, we follow (Kirillov et al., 2023b) to rescale the input image to 1024×1024 , by adjusting the long side while maintaining the aspect ratio and padding the short side. For the textual query input, to enhance the generalizability of the grounding model, we augment the text queries by sampling templates from a predefined set for different elements. For instance, the metadata from the data collection engine might only provide structured information such as $\{ \text{"class_name": "button", "content": "go back", "level-info": ["navigation bar", "go back"], \dots} \}$ for a button. Instead of merely concatenating this information into "button: go back," we generate more diverse queries like “Seek the go back button and click”, or “Help me find the go back button in the navigation bar”.

Following object detection methods (Carion et al., 2020), we add prediction heads to the decoder outputs for bounding box prediction. Each output from the learnable embeddings predicts both a score and bounding box coordinates. The predicted score is optimized to match the IoU of the predicted and ground-truth boxes using mean-square-error loss \mathcal{L}_{mse} . For bounding box regression, we use a linear combination of l_1 loss \mathcal{L}_{l_1} and the generalized IoU loss $\mathcal{L}_{\text{GIoU}}$ (Rezatofighi et al., 2019). The overall training loss for GUI grounding is given by: $\mathcal{L} = \lambda_{\text{mse}}\mathcal{L}_{\text{mse}} + \lambda_{l_1}\mathcal{L}_{l_1} + \lambda_{\text{GIoU}}\mathcal{L}_{\text{GIoU}}$, where λ_{mse} , λ_{l_1} , and λ_{GIoU} are hyper-parameters. Unlike Kirillov et al. (2023b), which backpropagates only from the lowest loss, we found that backpropagating from all predictions improves performance in GUI grounding. During inference, bounding boxes are ranked by score, and the top-1 box is selected as the final prediction.

4.2 INTEGRATING GROUNDING MODEL WITH MLLM AS AGENT

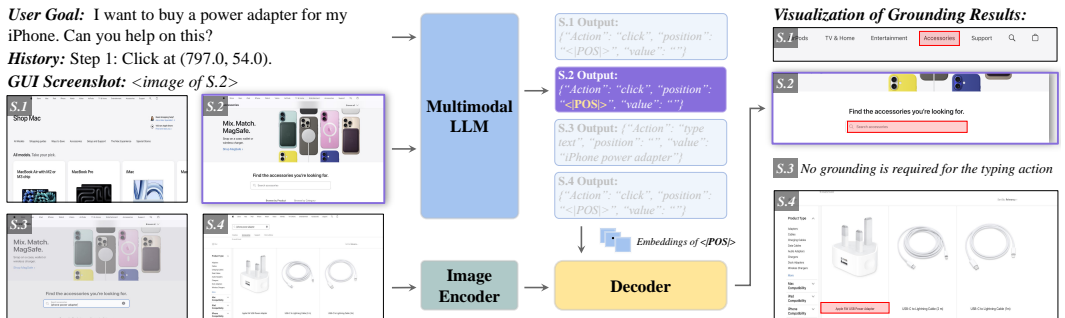


Figure 3: **Integrate AGG with MLLM as a GUI agent.** The proposed agent integrates an MLLM with a trained GUI grounding model. MLLM processes textual inputs (user goals, action history) and visual inputs (GUI screenshot) to generate a response. A special token triggers GUI grounding. The image encoder extracts visual features, then the lightweight decoder predicts the target GUI element’s position. The figure illustrates the process of task completion and highlights the step 2.

For vision-based agents, completing a complex GUI task typically involves several steps, each requiring the model to: (1) Perceive and understand: gather and interpret information from GUI images, user goals, and action history to understand the current state. (2) Reason and plan: analyze the current state to decide on the next action. (3) Execute: perform the desired action on the target GUI element. In this work, we propose the simple yet effective embeddings-as-position paradigm to integrate the GUI grounding model’s capabilities into an MLLM, thereby creating a vision-based agent tailored for these tasks. We detail the design below.

Architecture for the integrated agent. The proposed agent consists of an MLLM M and a well-trained GUI grounding model. Inspired by Lai et al. (2024), this approach involves several key steps. Initially, we add a new token $\langle |POS| \rangle$ to the original MLLM vocabulary, indicating a request for a GUI grounding output. Given the textual input X_{usr} , which includes the user goal and optionally, the action history, along with the visual input of the GUI image X_{img} , these inputs are fed into the MLLM, which generates a text-form response \hat{o}_{usr} . This process is formulated as $\hat{o}_{usr} = M(X_{img}, X_{usr})$. When the MLLM aims to generate a bounding box, the text-form output includes a $\langle |POS| \rangle$ token. We then extract the last layer embedding corresponding to the $\langle |POS| \rangle$ token and apply a linear projection layer to it. We denote this output as h_{pos} . Meanwhile, the image encoder from the grounding model extracts visual features from the input GUI image. Finally, these visual features and the projected embedding are fed into the lightweight decoder of the grounding model to predict the position P of the desired GUI element, formulated as $P = D(h_{pos}, I(X_{img}))$.

Training for agent tasks. The model is trained using the text token prediction loss \mathcal{L}_{text} and the grounding loss \mathcal{L}_{grd} . The overall objective for training the vision-based agent is: $\mathcal{L} = \lambda_{text}\mathcal{L}_{text} + \lambda_{grd}\mathcal{L}_{grd}$, where λ_{text} and λ_{grd} are hype-parameters. Especially, when the form of the target position is a bounding box, \mathcal{L}_{grd} is the same as that used in GUI grounding training. When the form of the target position is point, we slightly modify the prediction head upon the decoder output to predict the normalized distance to the ground truth point as a score indicator and predict the 2D point coordinates as the position output.

5 EXPERIMENT

5.1 EXPERIMENTAL SETUP

Training data for GUI Grounding. We train our GUI grounding model on two datasets: web UI data collected by our engine and mobile UI data reorganized from public sources. For web UI, we applied a data smoothing strategy (Hudson & Manning, 2019), resulting in 0.36M images with 22M element annotations. For mobile UI, following (Cheng et al., 2024), we used 27K screenshots and 275K query-bbox annotations from the WidgetCaption (Li et al., 2020) and RICO (Deka et al., 2017).

Evaluation on Grounding and Agent Benchmarks. We evaluate our GUI grounding model on the ScreenSpot (Cheng et al., 2024) benchmark, which tests grounding capabilities across platforms and applications. We then integrate the model with various MLLMs using the embeddings-as-position approach and evaluate on tasks like MiniWob (Shi et al., 2017), AITW (Rawles et al., 2023), and Mind2Web (Deng et al., 2024). Following (Cheng et al., 2024), we pretrain Qwen-VL (Bai et al., 2023b) with our data and a mixed public split to build an MLLM for GUI tasks, and further combine it with our trained grounding model. Instructions and previous action memory are provided as per (Cheng et al., 2024) during training and inference.

Main implementation details. We configure the image encoder and lightweight decoder of the GUI grounding model using the ViT-SAM-Large (Kirillov et al., 2023b) and the language prompt encoder with a BERT-based (Devlin et al., 2019) setup. Training runs on 32 V100 GPUs with a global batch size of 128 for 150K steps. The AdamW (Loshchilov, 2017) optimizer is used with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and a weight decay of $1e-4$. A Cosine Annealing scheduler manages the learning rate, starting with a warm-up over 200 steps. The max learning rate is $1e-3$, dropping to $5e-5$. Hyperparameters for the training objective are set as $\lambda_{mse} = 10$, $\mathcal{L}l_1 = 5$, and $\lambda_{GIoU} = 2$. For vision-based agent tasks, we fine-tune with 2 A100 GPUs, applying LoRA (Hu et al., 2021) tuning (rank 8, alpha 16) for the language model. The decoder parameters are unfrozen, while the image encoder remains frozen. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and no weight decay. The training objective hyperparameters are $\lambda_{text} = 1$ and $\mathcal{L}_{grd} = 1$.

5.2 EXPERIMENTAL RESULTS

Results on ScreenSpot. ScreenSpot is a benchmark comprising over 600 interface screenshots and more than 1200 annotations for query-element pairs. It spans mobile platforms (iOS, Android), desktop platforms (macOS, Windows), and web platforms.

For this benchmark, we compare our GUI grounding model with generalist MLLMs capable of recognition and reasoning (Chen et al., 2023; Bai et al., 2023b; Yan et al., 2023; Abdin et al., 2024), as well as recent MLLMs that have incorporated GUI-related tasks during pretraining (Bavishi et al., 2023; Hong et al., 2023; Cheng et al., 2024).

The results, shown in Table 2, highlight that while generalist MLLMs possess extensive knowledge about natural images, their GUI grounding performance on ScreenSpot is subpar due to the significant differences between GUIs and natural images. Even GPT-4V struggles to accurately locate screen elements. GUI-specific MLLMs demonstrate better performance. In particular, Our AGG achieves the highest performance across different platforms and for both text and icon elements, even with fewer model parameters. This underscores the advantages of our collected data for GUI grounding training, allowing AGG to effectively handle diverse queries and rich elements encountered during training. With rich elements and diverse query seen during training, AGG possess the capability for GUI grounding, even for the very less scenarios like iOS. Compared to text element grounding, the model’s performance is less strong on icon-based tasks, indicating that grounding non-text elements in GUIs remains challenging.

Method	GUI.S	Model size	Mobile		Desktop		Web		Avg.
			Text	Icon	Text	Icon	Text	Icon	
MiniGPTv2 (Chen et al., 2023)	✗	7B	8.4	6.6	6.2	2.9	6.5	3.4	5.7
Qwen-VL (Bai et al., 2023b)	✗	9.6B	9.5	4.8	5.7	5.0	3.5	2.4	5.2
Phi-3.5-V (Abdin et al., 2024)	✗	4.2B	1.7	1.7	4.1	0.7	4.7	2.0	2.5
GPT-4V OpenAI (2023a)	✗	-	22.6	24.5	20.2	11.8	9.2	8.8	16.2
Fuyu (Bavishi et al., 2023)	✓	8B	41.0	1.3	33.0	3.6	33.9	4.4	19.5
CogAgent (Hong et al., 2023)	✓	18B	67.0	24.0	74.2	20.0	70.4	28.6	47.4
SeeClick (Cheng et al., 2024)	✓	9.6B	78.0	52.0	72.2	30.0	55.7	32.5	53.4
AGG (Ours)	✓	0.4B	86.1	62.8	81.8	46.2	74.2	48.4	66.6

Table 2: Performance of various models on ScreenSpot. “GUI.S” indicates “GUI Specific”, denoting whether the model was trained using domain-specific data. The best results in each column are highlighted in bold.

Results on MiniWob. MiniWob Shi et al. (2017) features around 100 distinct web automation tasks, requiring the agent to navigate a simplified web environment and execute human-given instructions. In line with the approach used in (Cheng et al., 2024), we conduct 2.8K episode rollouts for training.

Method	Modality	Dataset	Score
CC-Net(SL) (Humphreys et al., 2022)	DOM+Image	2.4M	36.5
WebN-T5 (Gur et al., 2022)	HTML	12K	55.2
WebGUM (Furuta et al., 2024)	HTML+Image	2.8K	65.5
WebGUM (Furuta et al., 2024)	HTML+Image	347K	86.1
SeeClick Cheng et al. (2024)	Image	2.8K	73.6
MLLM ◦ AGG (Ours)	Image	2.8K	75.8

Table 3: Results on MiniWob over 45-tasks split.

Method	Modality	Dataset	Score
CC-Net(SL) (Humphreys et al., 2022)	Image	2.4M	23.4
Pix2Act (Shaw et al., 2024)	Image	12K	55.2
Qwen-VL (Bai et al., 2023b)	HTML+Image	2.8K	48.4
SeeClick Cheng et al. (2024)	Image	2.8K	67.0
MLLM ◦ AGG (Ours)	Image	2.8K	69.3

Table 4: Results on MiniWob over-35 tasks split.

We compared our method with models using image-only inputs, text-only inputs, and a combination of both. Note that with text input from the environment, the model selects the HTML element from candidates as the action target.

Due to variations in the evaluation task sets used by different methods, we follow (Cheng et al., 2024) and report performance on two groups of data splits. We compute the success rate over 50 random seeds for each task and then calculate the mean success rate across all tasks as the final score.

Results are presented in Table 3 and Table 4. We denote our method as “MLLM ◦ AGG”. The results show that our vision-based agent surpasses previous methods. Notably, with 2.8K training episodes, it outperforms WebGUM, which uses both HTML and GUI images as input and exceeds the vision-based Pix2Act, which used less than 0.3% as much training data. Furthermore, our method outperforms MLLM-involved models like Qwen-VL and the SOTA SeeClick, demonstrating the efficacy of our framework.

Results on AITW. Android In The Wild (AITW) (Rawles et al., 2023) dataset is a mobile navigation benchmark including 30K instructions and 715K corresponding operation trajectories. Following the approach detailed in (Cheng et al., 2024), we apply the same train/test split based on instructions, retaining a single trajectory per instruction and ensuring no overlap between the training and test sets.

We compare our method to API-based LLMs and previous MLLM-based methods. We use the screen-wise action matching score as the main metric, as outlined in (Rawles et al., 2023), and follow (Cheng et al., 2024) to compute click accuracy to evaluate grounding capabilities.

Results are illustrated in Table 5. Our method achieved the best performance across all tasks, surpassing both the API-based LLMs and the previous GUI data pretrained MLLMs. Notably, our method exhibits higher click accuracy than other methods, demonstrating stronger GUI grounding capabilities in this scenario.

Method	Modality	General	Install	GoogleApps	Single	WebShopping	Overall	ClickAcc
ChatGPT-CoT (Zhang & Zhang, 2024)	Text	5.9	4.4	10.5	9.4	8.4	7.7	-
PaLM2-CoT (Rawles et al., 2023)	Text	-	-	-	-	-	39.6	-
GPT-4V (Yan et al., 2023)	Image	41.7	42.6	49.8	72.8	45.7	50.5	-
Qwen-VL Bai et al. (2023b)	Image	49.5	59.9	46.9	64.7	50.7	54.3	57.3
SeeClick Cheng et al. (2024)	Image	54.0	66.4	54.9	63.5	57.6	59.3	66.4
MLLM \circ AGG (Ours)	Image	56.4	67.8	56.6	67.2	63.1	62.2	68.1

Table 5: Performance of different models on AITW. The upper half corresponds to the performance of API-based LLMs.

Method	PT	Cross-Task			Cross-Website			Cross-Domain		
		Ele.Acc	Op.F1	Step.SR	Ele.Acc	Op.F1	Step.SR	Ele.Acc	Op.F1	Step.SR
MindAct (Gen)	-	20.2	52.0	17.5	13.9	44.7	11.0	14.2	44.7	11.9
MindAct (Deng et al., 2024)	-	55.1	75.7	52.0	42.0	65.2	38.9	42.1	66.5	39.6
GPT-3.5-Turbo	-	20.3	56.6	17.4	19.3	48.8	16.2	21.6	52.8	18.6
GPT-4 (OpenAI, 2023a)	-	41.6	60.6	36.2	35.8	51.1	30.1	37.1	46.5	26.4
Qwen-VL (Bai et al., 2023a)	✗	15.9	86.7	13.3	13.2	83.5	9.2	14.1	84.3	12.0
SeeClick (Cheng et al., 2024)	✓	28.3	87.0	25.5	21.4	80.6	16.4	23.2	84.8	20.8
MLLM \circ AGG (Ours)	✓	41.2	87.4	36.8	39.2	84.8	33.8	38.9	85.2	34.2

Table 6: Performance of different models on Mind2Web. “Ele.Acc” denotes element accuracy, “Op.F1” denotes operation F1 score and “Step.SR” denotes step success rate. “PT” indicates if the MLLM is pretrained on GUI data.

Results on Mind2Web. Mind2Web (Deng et al., 2024) is a benchmark that evaluates an agent’s capabilities in web navigation tasks. It encompasses over 2000 open-ended tasks sourced from 137 real website URLs, each accompanied by high-level instructions and corresponding human action trajectories. The original Mind2Web was originally designed for text-based agents, which select actionable elements from filtered and simplified HTML elements in each step. In this work, following (Cheng et al., 2024), we use parsed screenshots and target element bounding boxes from the raw HTML data of Mind2Web.

We compare our approach with both text-based web agents Deng et al. (2024); OpenAI (2023a) and vision-based agents (Bai et al., 2023b; Cheng et al., 2024). Mind2Act employs a two-step process: a small language model first generates candidate elements from raw HTML, followed by a large language model that selects the target via multiple-choice QA. Mind2Act(gen) directly generates the target element. GPT-3.5 and GPT-4 use the same multiple-choice QA formulation and include three demonstrations for in-context learning. Qwen-VL, SeeClick and our model are fine-tuned using trajectory annotations.

Following the evaluation metrics from (Cheng et al., 2024; Deng et al., 2024), we measure element accuracy, operation F1, and step success rate (StepSR). For vision-based models, a prediction is considered correct if the predicted point (or the center of the predicted bounding box) falls within the target element’s ground-truth bounding box.

The results presented in Table 6 demonstrate that pretraining on GUI grounding data significantly enhances agent performance. Our method also surpasses the previous state-of-the-art vision-based agent, SeeClick, by a large margin in terms of element accuracy, underscoring the robust grounding capabilities of our model. However, our model still falls short of the top-performing text-based

GUI agents, as grounding elements in images is inherently more complex than selecting an element from multiple choices. This highlights the challenges associated with GUI element grounding and suggests that there remains substantial room for improvement in deploying vision-based GUI agents for real-world applications.

5.3 ABLATION STUDY

In this section, we conduct ablation studies to analyze the effectiveness of various design elements in the grounding model and vision-based GUI agent.

Model Var	avg. ele.acc	Training config	avg.	# prompt emb.	avg.
Qwen-VL	14.4	#1. Random init, Full	41.1	1	46.2
Qwen-VL ◦ AGG	27.4	#2. SAM init, Lora	46.8	2	48.4
Qwen-VL(SC data)	24.3	#3. prop min	47.3	3	49.2
Qwen-VL(SC data) ◦ AGG	35.2	#4. prop max	48.6	4	49.0
Qwen-VL(our data)	30.7	#5. w/o aug	44.5	5	49.8
Qwen-VL(our data) ◦ AGG	39.8	#6. <i>default</i>	49.2		

Table 7: Effective factors for agent tasks: different data to pretrain the MLLM and whether incorporate AGG. The default setting is marked with a color box .

Table 8: Training configuration for AGG training. Default setting is marked with color box .

Table 9: Number of prompt embedding for decoder. Default setting is marked with color box .

Effective factors for improving agent tasks. We assess the impact of our collected data and AGG. Specifically, we evaluate the Qwen-VL (Bai et al., 2023b) MLLM without GUI data pretraining, trained with SeeClick’s pretraining data, and trained with our pretraining data. We then integrate AGG to these model variants. The average element accuracy on Mind2Web Deng et al. (2024) is reported. Results in Table 7 indicate that using our collected data during pretraining results in an MLLM with superior GUI grounding capabilities. Additionally, integrating AGG leads to further improvements, demonstrating the effectiveness of our proposed paradigm.

Training Configurations for the Image Encoder. We examine the impact of using SAM model weights for initialization and different tuning strategies for the image encoder during training. The results in Table 8(#1,2,6) demonstrate that initializing the image encoder with SAM model weights and fully tuning it during training yields the best performance.

Grounding Loss Propagation During Training. We examine the impact of loss propagation configurations for GUI grounding training. The results shown in Table 8(#3,4,6) indicate that propagating the loss from all prompt embeddings during training yields the best performance for GUI grounding.

Effectiveness of augmenting prompt queries. We investigate the effect of augmenting the query from metadata during GUI grounding training. Results in Table 8(#5,6) show that augmenting query significantly enhance the GUI grounding capability, which is important to build a generalized vision-based GUI agent.

Number of Learnable Prompt Embeddings for Decoder. We investigate the impact of the number of learnable prompt embeddings used in the decoder. Table 9 shows that increasing the number of prompts beyond 3 yields no significant improvement in grounding performance.

6 CONCLUSION

In this work, we investigated the critical challenge of precise GUI element grounding, which is essential for accurate interaction and generalization in GUI agents. We developed a simple yet effective GUI grounding framework that includes an automated data collection engine to gather extensive GUI screenshots and annotations, ensuring broad generalization. We introduced a lightweight GUI grounding module designed for efficient UI element localization by pre-training on the collected data. By integrating this module with MLLMs, we demonstrated superior performance in task accuracy and adaptability across multiple benchmarks, including ScreenSpot, MiniWob, AITW, and Mind2Web. Our results highlight the effectiveness of our approach in enhancing the capabilities of GUI agents for complex tasks on digital platforms.

REFERENCES

- 540
541
542 Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany
543 Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report:
544 A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.
- 545 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman,
546 Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report.
547 *arXiv preprint arXiv:2303.08774*, 2023.
- 548 Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge,
549 Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023a.
- 550
551 Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang
552 Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities.
553 *arXiv preprint arXiv:2308.12966*, 2023b.
- 554 Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and
555 Sağnak Taşlılar. Introducing our multimodal models, 2023. URL [https://www.adept.ai/
556 blog/fuyu-8b](https://www.adept.ai/blog/fuyu-8b).
- 557
558 Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey
559 Zagoruyko. End-to-end object detection with transformers. In *European conference on computer
560 vision*, pp. 213–229. Springer, 2020.
- 561 Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren,
562 and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv
563 preprint arXiv:2407.17490*, 2024.
- 564
565 Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong
566 Wang, Huichi Zhou, Yiqiang Li, et al. Gui-world: A dataset for gui-oriented multimodal llm-based
567 agents. *arXiv preprint arXiv:2406.10819*, 2024a.
- 568 Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman
569 Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large
570 language model as a unified interface for vision-language multi-task learning. *arXiv preprint
571 arXiv:2310.09478*, 2023.
- 572 Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu,
573 Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language models to versatile
574 gui agents. *arXiv preprint arXiv:2406.11317*, 2024b.
- 575
576 Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu.
577 SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd
578 Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp.
579 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL
580 <https://aclanthology.org/2024.acl-long.505>.
- 581 Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng,
582 Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An
583 open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL [https:
584 //lmsys.org/blog/2023-03-30-vicuna/](https://lmsys.org/blog/2023-03-30-vicuna/).
- 585 Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschan, Daniel Afegan, Yang Li, Jeffrey
586 Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design
587 applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software
588 and Technology*, pp. 845–854, New York, NY, USA, 2017. Association for Computing Machinery.
589 ISBN 9781450349819. doi: 10.1145/3126594.3126651. URL [https://doi.org/10.1145/
590 3126594.3126651](https://doi.org/10.1145/3126594.3126651).
- 591 Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and
592 Yu Su. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on
593 Neural Information Processing Systems*, 2023. URL [https://openreview.net/forum?
id=kiYqbO3wqw](https://openreview.net/forum?id=kiYqbO3wqw).

- 594 Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su.
595 Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing*
596 *Systems*, 36, 2024.
- 597
598 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of
599 deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and
600 Tamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the*
601 *Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and*
602 *Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational
603 Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- 604 Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas
605 Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit,
606 and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale.
607 In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=YicbFdNTTy>.
- 608
609 Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane
610 Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models.
611 In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=efFmBWioSc>.
- 612
613 Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchun Wu, Weichen
614 Zhang, Peiyi Wang, Xiangwu Guo, Hengxu Wang, Luowei Zhou, and Mike Zheng Shou. Assistgui:
615 Task-oriented pc graphical user interface automation. In *Proceedings of the IEEE/CVF Conference*
616 *on Computer Vision and Pattern Recognition (CVPR)*, pp. 13289–13298, June 2024.
- 617
618 Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery,
619 Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding html with large language
620 models. *arXiv preprint arXiv:2210.03945*, 2022.
- 621
622 Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and
623 Aleksandra Faust. A real-world webagent with planning, long context understanding, and program
624 synthesis. *arXiv preprint arXiv:2307.12856*, 2023.
- 625
626 Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan
627 Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. *arXiv*
preprint arXiv:2312.08914, 2023.
- 628
629 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,
630 and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint*
arXiv:2106.09685, 2021.
- 631
632 Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning
633 and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer*
634 *vision and pattern recognition*, pp. 6700–6709, 2019.
- 635
636 Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair
637 Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. A data-driven
638 approach for learning to control computers. In *International Conference on Machine Learning*, pp.
639 9466–9482. PMLR, 2022.
- 640
641 Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot,
642 Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al.
Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- 643
644 Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks.
645 *Advances in Neural Information Processing Systems*, 36, 2024.
- 646
647 Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete
Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings*
of the IEEE/CVF International Conference on Computer Vision, pp. 4015–4026, 2023a.

- 648 Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete
649 Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings*
650 *of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023b.
- 651
- 652 Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham
653 Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating
654 multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.
- 655
- 656 Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning
657 segmentation via large language model. In *Proceedings of the IEEE/CVF Conference on Computer*
658 *Vision and Pattern Recognition*, pp. 9579–9589, 2024.
- 659
- 660 Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos,
661 Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot
662 parsing as pretraining for visual language understanding. In *International Conference on Machine*
Learning, pp. 18893–18912. PMLR, 2023.
- 663
- 664 Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image
665 pre-training with frozen image encoders and large language models. In *International conference*
666 *on machine learning*, pp. 19730–19742. PMLR, 2023.
- 667
- 668 Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning:
669 Generating natural language description for mobile user interface elements. In Bonnie Webber,
670 Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical*
671 *Methods in Natural Language Processing (EMNLP)*, pp. 5495–5510, Online, November 2020.
Association for Computational Linguistics.
- 672
- 673 Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer
674 backbones for object detection. In *European conference on computer vision*, pp. 280–296. Springer,
2022.
- 675
- 676 Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement
677 learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*,
678 2018.
- 679
- 680 Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in*
681 *neural information processing systems*, 36, 2024.
- 682
- 683 I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- 684
- 685 Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang,
686 Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui
687 navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024a.
- 688
- 689 Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based
690 gui agent. *arXiv preprint arXiv:2408.00203*, 2024b.
- 691
- 692 Chuwei Luo, Yufan Shen, Zhaoqing Zhu, Qi Zheng, Zhi Yu, and Cong Yao. Layoutllm: Layout
693 instruction tuning with large language models for document understanding. In *Proceedings of the*
694 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15630–15640, 2024.
- 695
- 696 OpenAI. Gpt-4 technical report, 2023a.
- 697
- 698 OpenAI. Gpt-4v(ision) system card, 2023b. URL [https://openai.com/index/
699 gpt-4v-system-card/](https://openai.com/index/gpt-4v-system-card/).
- 700
- 701 Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the
wild: A large-scale dataset for android device control. *arXiv preprint arXiv:2307.10088*, 2023.
- 702
- 703 Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. An-
droidinthewild: A large-scale dataset for android device control. *Advances in Neural Information*
Processing Systems, 36, 2024.

- 702 Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese.
703 Generalized intersection over union: A metric and a loss for bounding box regression. In *Pro-*
704 *ceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 658–666,
705 2019.
- 706 Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi
707 Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow
708 instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36:
709 34354–34370, 2023.
- 710 Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi
711 Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow
712 instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36,
713 2024.
- 714 Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits:
715 An open-domain platform for web-based agents. In Doina Precup and Yee Whye Teh (eds.),
716 *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings*
717 *of Machine Learning Research*, pp. 3135–3144. PMLR, 06–11 Aug 2017.
- 718 Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Abhanshu Sharma,
719 James Stout, et al. Towards better semantic understanding of mobile interfaces. *arXiv preprint*
720 *arXiv:2210.02663*, 2022.
- 721 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
722 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and
723 efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.
- 724 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay
725 Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation
726 and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.
- 727 Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Au-
728 tomatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium*
729 *on User Interface Software and Technology*, pp. 498–510, 2021.
- 730 Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao
731 Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv*
732 *preprint arXiv:2401.16158*, 2024.
- 733 Li Wang, Zechen Bai, Yonghua Zhang, and Hongtao Lu. Show, recall, and tell: Image captioning
734 with recall mechanism. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34,
735 pp. 12176–12183, 2020.
- 736 Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing
737 Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal
738 agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*,
739 2024.
- 740 An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu
741 Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for
742 zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.
- 743 Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent:
744 Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023.
- 745 Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei
746 Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *arXiv*
747 *preprint arXiv:2404.05719*, 2024.
- 748 Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu
749 Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*,
750 2024.

Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 3132–3149, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.186. URL <https://aclanthology.org/2024.findings-acl.186>.

Longtao Zheng, Rundong Wang, and Bo An. Synapse: Leveraging few-shot exemplars for human-level computer control. *arXiv preprint arXiv:2306.07863*, 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. *arXiv preprint arXiv:2307.13854*, 2023.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

APPENDIX

A MORE INFORMATION FOR THE COLLECTED DATA

In this section, we provide additional details about our collected data.

Topics and Domains. We list the topics and some sampled domains in our collected data, as is shown in Table 10.

Topic	Related Domains
News and Journalism	www.bbc.com , www.cnn.com , ...
Technology and Science	www.wired.com , www.sciencedaily.com , ...
Sports	www.espn.com , www.cbssports.com , ...
Entertainment	www.imdb.com , www.netflix.com , ...
Lifestyle and Wellness	www.wellandgood.com , www.goop.com , ...
Shopping and E-commerce	www.amazon.com , www.ebay.com , ...
Travel	www.airbnb.com , www.booking.com , ...
Food and Cooking	tasty.co , www.foodnetwork.com , ...
DIY and Home Improvement	www.diynetwork.com , www.houzz.com , ...
Educational and Learning	www.coursera.org , www.ted.com , ...
Others	www.github.com , www.capcut.com , ...

Table 10: Examples topics and some corresponding domains in our collected data.

Data Example Visualization. In Figure 4, we present several samples from our collected dataset. These visualizations include bounding boxes highlighting the corresponding elements.

Examples of Augmented Query Templates. In Table 11, we showcase a portion of the augmented query templates used to generate queries for GUI grounding training. These templates include both common templates and element-specific templates. During training, the query is generated using either the common templates or the element-specific templates.

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863

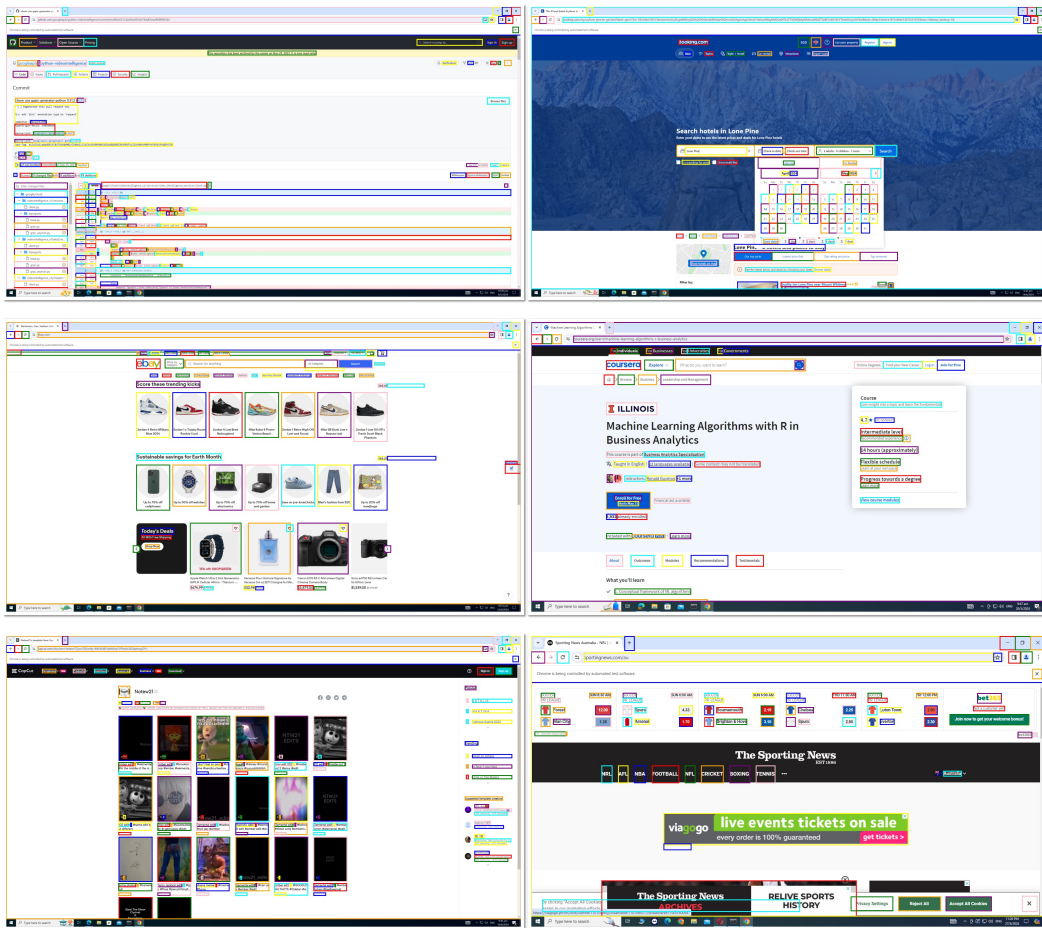


Figure 4: Data examples from our collected data.

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

Augmented Query Examples.	
Examples for common queries	<ul style="list-style-type: none"> ○ "Look for {content}", ○ "Identify the element {content}", ○ "Locate the {class name} called {content}", ○ "Scroll to the {class name} with {content}", ○ "Highlight the {class name} that contains {content}", ○ "Focus on the {class name} that displays {content}", ○ "Locate the {class name} that is part of {parent info} with the name {content}" ...
Examples for specific element (button)	<ul style="list-style-type: none"> ○ "Click the {content} button", ○ "Press the button labelled {content}", ○ "Activate the {content} button", ○ "Find and click the {content} on the page", ○ "Where is the {content} button located?", ○ "Can you locate the button that says {content}?", ○ "Click the {content} button in the {parent info}" ...
Examples for specific element (DataItem)	<ul style="list-style-type: none"> ○ "Tick the {content} checkbox", ○ "Check the {content} option", ○ "Mark the checkbox for {content}", ○ "Select the {content} checkbox from the list", ○ "Ensure the {content} checkbox is checked", ○ "Activate the checkbox labeled {content}", ○ "Verify that the {content} checkbox is selected" ...
Examples for specific element (checkbox)	<ul style="list-style-type: none"> ○ "Identify the data item labeled {content}", ○ "Locate the {content} data field on the page", ○ "Find the {content} item in the dataset", ○ "Access the information for {content}", ○ "Highlight the data item {content}", ○ "Explore the {content} item for more info" ...
Examples for specific element (hyperlink)	<ul style="list-style-type: none"> ○ "Use the {content} link to access the desired page", ○ "Identify the link called {content} and click on it", ○ "Proceed by using the hyperlink {content}", ○ "Check the {content} link for more details", ○ "Open the web page linked by {content}", ○ "Access resources through the {content} hyperlink", ○ "Click the {content} URL to navigate", ○ "Follow the online pointer marked as {content}" ...
Examples for specific element (static text)	<ul style="list-style-type: none"> ○ "Highlight the statement {content}", ○ "Observe the static text {content} on this page", ○ "Search for the label {content} in the content", ○ "Inspect the static field titled {content}", ○ "See the displayed static text {content}", ○ "Look for the static description {content}", ○ "Spot the phrase {content} in the view", ○ "Find the static wording {content}" ...

Table 11: Augmented query examples during GUI grounding training. Shown are examples of some of the augmented templates.