UI-Hawk: Unleashing the Screen Stream Understanding for Mobile GUI Agents

Jiwen Zhang[1,2*], **Ya-Qi Yu**[2*],
Minghui Liao[2†], **Wentao Li**[2], **Jihao Wu**[2], **Zhongyu Wei**[1†]
[1]Fudan University [2]Huawei Technologies Co., Ltd.
jiwenzhang21@m.fudan.edu.cn yuyaqi5@huawei.com

Abstract

Graphical User Interface (GUI) agents are expected to precisely operate on the screens of digital devices. Existing GUI agents merely depend on current visual observations and plain-text action history, ignoring the significance of history screens. To mitigate this issue, we propose ***UI-Hawk***, a multi-modal GUI agent specially designed to process screen streams encountered during GUI navigation. UI-Hawk incorporates a history-aware visual encoder to handle the screen sequences. To acquire a better understanding of screen streams, we select four fundamental tasks—*UI grounding*, *UI referring*, *screen question answering*, and *screen summarization*. We further propose a curriculum learning strategy to subsequently guide the model from fundamental tasks to advanced screen-stream comprehension. Along with the efforts above, we have created a benchmark ***FunUI*** to quantitatively evaluate the fundamental screen understanding ability of MLLMs. Extensive experiments on FunUI and GUI navigation benchmarks consistently validate that screen stream understanding is essential for GUI tasks. Our code and data are now available at https://github.com/IMNearth/UIHawk.

1 Introduction

Smartphones have become integral to daily life, raising the importance of autonomously operating graphical user interfaces (GUI). The task of following instructions on the GUI, formalized as GUI navigation, offers substantial potential to automate complex tasks, reduce human workload, and improve user experiences across various applications.

Recent advances in multimodal large language models (MLLMs) have greatly accelerated the development of GUI navigation agents, by either prompting GPT-4V (OpenAI, 2023) as the zero-shot task executor (Yang et al., 2023; Wang et al., 2024; Zhang et al., 2024a) or directly tuning MLLMs on the downstream GUI tasks (Zhan and Zhang, 2023; Hong et al., 2024).

These agents base their decision making primarily on current visual observations. Although textual action history is included to substitute the global context (Zhan and Zhang, 2023), plain text based action history such as "`click [x1, y1, x2, y2], then scroll up`" struggles to capture the nuanced details of clicked UI element, thereby hindering the progress (Zhang et al., 2024b). The rich semantics embedded within the screens is necessary for GUI agents to accurately control mobile devices. As shown in Figure 1, precisely grounding the search bar facilitates the prediction of a click action, followed by selecting "hiking trail" as the search option. Agents could read out the action semantics by grounding and referring to the corresponding screen. The screen stream demonstrating that it has searched for "hiking trial" and opened a related article supports the agent to mark the task as "done". This underscores the importance of understanding screen streams during GUI navigation.

The development of screen stream understanding encounters two major challenges: (1) Efficient representation of screen sequences, especially for MLLMs with limited context window (Bai et al., 2023; Yang et al., 2024) is challenging. (2) As illustrated in Figure 1, the instructions associated with the screen streams could "*refer*" to different elements, requiring the agents to "*ground*" its understanding in the correct regions. Additionally, user instructions could pose complex questions about the screen, necessitating the agent to analyze, "*answer*" and "*summarize*". Building a sophisticated model endowed with these capabilities is difficult.

In this paper, we introduce UI-Hawk, a MLLM-based GUI agent equipped with screen stream understanding capabilities. Firstly, we enable UI-Hawk to harness screen sequences by incorporat-

*Equal contribution.
†Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 18217–18236

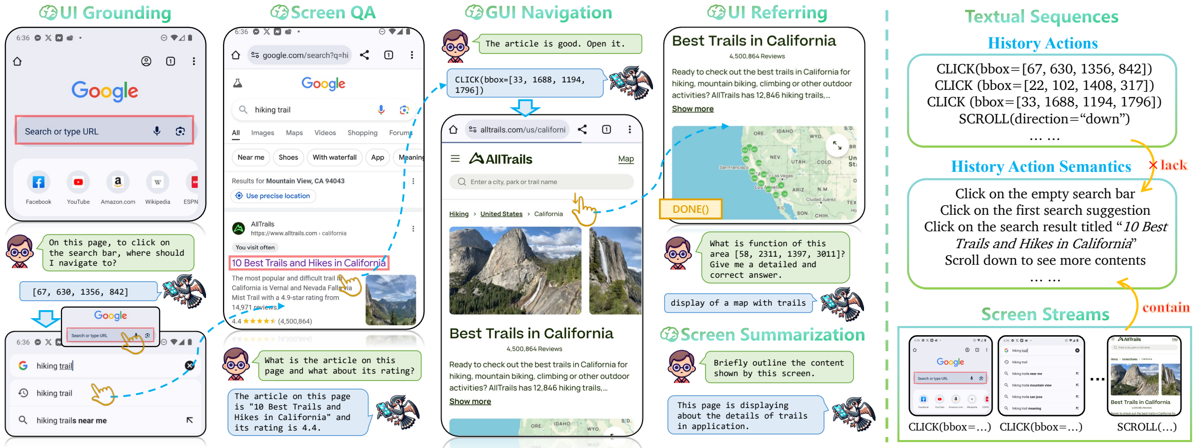


Figure 1: **Example of a GUI navigation episode together with the UI understanding tasks supported by UI-Hawk.** User instruction is *"I want to use Chrome to discover a new hiking trail."* Bounding boxes predicted by UI-Hawk are represented by red rectangles. Navigation actions are denoted by yellow hands and yellow rectangles.

ing a history-aware visual encoder, which explicitly models the temporal dependencies of images. Then, to mitigate the challenge of obtaining efficient visual representations, we borrow the resampler from TextHawk (Yu et al., 2024) with 16x compression ratio to process the visual tokens, enabling UI-Hawk to handle multiple steps of history screens. This specific architecture empowers UI-Hawk to effectively perceive the fine-grained details involved in the entire navigation process. Lastly, to substantially acquire the screen stream understanding capabilities, we adopt a curriculum-like training paradigm. We initially train UI-Hawk on several single-step screen understanding tasks, including *UI grounding*, *UI referring*, *screen question answering* and *screen summarization*, and then transfer the model as an agent on episodic navigation tasks to facilitate screen stream understanding.

Considering the significance of these fundamental capabilities (Cheng et al., 2024; Fan et al., 2024), we introduce ***FunUI***, a comprehensive benchmark to quantitatively evaluate the single-step understanding of screens. *FunUI* contains 2150 Chinese screenshots and 9347 English screenshots, covering 32k annotated samples with a variety of icons, texts and widgets. We assure the diversity of the *FunUI* dataset by collecting nine categories of questions. Evaluation results on *FunUI* benchmark and episodic GUI navigation tasks demonstrate that UI-Hawk establishes a new standard for screen understanding. Our further ablation experiments prove that, equipped with advanced screen stream understanding capabilities, UI-Hawk achieves new state-of-the-art performance on both English and Chinese GUI navigation tasks, improving the prediction accuracy by 7.7% and 6.7%, respectively.

Our contributions are summarized as follows.

- We introduce a GUI agent, UI-Hawk, that can effectively process stream of screens via a history-aware visual encoder.
- We focus on four fundamental tasks for screen understanding, and validate the usefulness of these tasks towards episodic navigation through our curriculum learning strategy.
- We rigorously construct a comprehensive screen understanding benchmark *FunUI*, encompassing 32k samples with over 120 types of UI elements.
- Experiments demonstrate that possessing the screen stream understanding capability is the key to enhancing the performance of GUI navigation.

2 Methodology

To enable screen stream understanding, UI-Hawk introduces two key characteristics: (1) an optimized model architecture for efficient screen perception, detailed in Section 2.1, and (2) the curriculum learning paradigm encompassing a wide range of screen tasks, as outlined in Section 2.2.

2.1 Model Architecture

Given that mobile device screenshots typically have high and variable resolutions, a highly efficient and fine-grained perception capability is crucial for developing effective mobile GUI agents. We begin by identifying several essential requirements: the ability to handle multiple images of any resolution simultaneously, efficient compression of visual tokens, accurate OCR functionality, and precise referring and grounding capabilities. Among existing

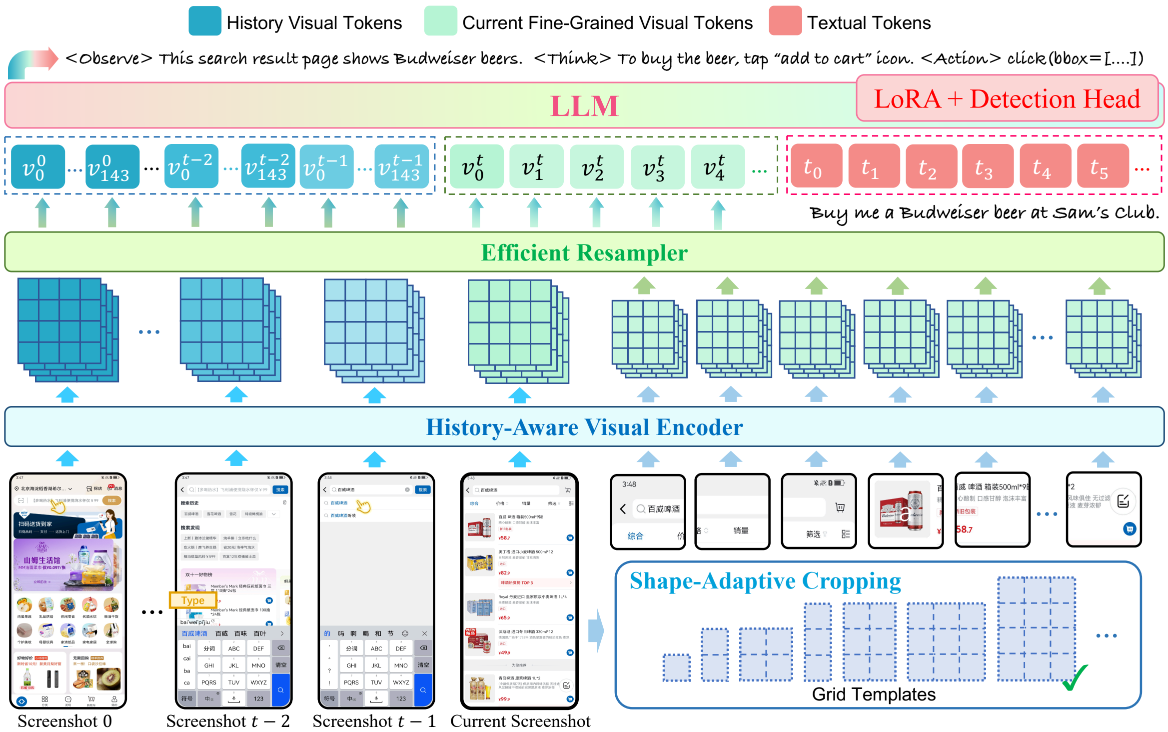


Figure 2: **Model architecture of UI-Hawk.** The text tokenizer, layer normalization, and skip connections are omitted for simplicity. During pre-training, the visual encoder is trained together with the LLM to obtain fine-grained perception capabilities. During fine-tuning, the visual encoder is frozen and the LLM is tuned with the resampler.

foundational MLLMs, TextHawk (Yu et al., 2024) stands out as the closest to fulfilling these needs. Specifically, UI-Hawk inherits several key features from TextHawk: (1) a shape-adaptive cropping strategy that processes images of any resolution, to perceive fine-grained details across various screen sizes; (2) a carefully designed resampler with 16× compression ratio, to efficiently encode visual tokens; (3) a detection head for direct modeling of bounding boxes, to explicitly improve the grounding abilities. We replace the original language backbone InternLM1.0-7B (Team, 2023) of TextHawk with Qwen2-7B (Yang et al., 2024), and employ SigLIP-SO (Zhai et al., 2023) as the visual encoder. The model architecture is depicted in Figure 2.

Different from TextHawk, UI-Hawk places a strong emphasis on modeling historical screens, as visual history often contain valuable details pertinent to ongoing tasks. Despite proprietary MLLM-based agents (Yan et al., 2023; Zheng et al., 2024; Zhang et al., 2024a; He et al., 2024) being capable of processing multiple screenshots, such capability is lacked for open-sourced MLLM-based agents, most of which rely solely on text-based history, like chain-of-actions (Zhan and Zhang, 2023) or chain-of-action-thoughts (Zhang et al., 2024b). To address this gap, UI-Hawk incorporates images of observed screens as model inputs, and explicitly add special tokens (e.g., "<History Screenshot>") for each historical screen to explicitly represent the screen streams. Unlike TextHawk, we devise a curriculum-like tuning strategy to understand screen streams, where UI-Hawk starts with learning across multiple images with single-step screen-related tasks and then extends to serialized GUI navigation tasks, as detailed in the following Section 2.2. Moreover, previous models faced challenges with efficiently modeling visual history, as encoding each page required thousands of visual tokens (Bai et al., 2023; Ye et al., 2023). To overcome this, UI-Hawk downscales history images to a quarter of their original size. As UI-Hawk employs a much larger visual token compression ratio of 16, a typical historical screenshot is divided into 8 sub-images along with a global thumbnail, using only 144 visual tokens.

2.2 Model Training

We train UI-Hawk from scratch by utilizing the pre-training mixtures from TextHawk (Yu et al., 2024). As TextHawk has not encountered mobile screen images during pre-training, we supplement the pre-training mixtures with the screen annotation dataset collected in Section 3.1. We unfreeze the

Task	# Samples		Data Source	
	ZH	EN	ZH	EN
UI Grounding	580k	16k	Ours	(Bai et al., 2021)
UI Referring	600k	109k	Ours	(Li et al., 2020)
Screen QA	1200k	288k	Ours	(Hsiao et al., 2022)
Screen Sum.	50k	78k	Ours	(Wang et al., 2021)
GUI Navigation	55k	87k	Ours	(Lu et al., 2024)

Table 1: **Summary of the fine-tuning data of UI-Hawk.** "Screen Sum." is short for screen summarization task. For GUI navigation tasks, we measure the number of samples by counting the time steps in each episode.

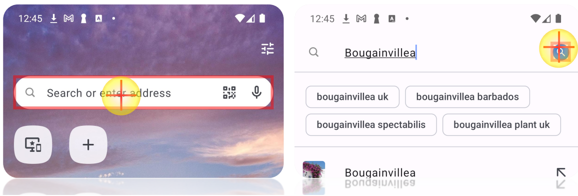


Figure 3: **Examples from GUI-Odyssey dataset.** *Left*: The region of clicked element (red bounding box) is larger than area where click action is considered correct (shadowed orange circle). *Right*: The region of clicked element is smaller than the area of correct click actions.

ViT by LoRA (Hu et al., 2021) and train UI-Hawk for one epoch. This is also one-step further than TextHawk which froze the ViT during pre-training. Our pre-training improves both the OCR and the screen infographics understanding ability of UI-Hawk, taking 7 days on 128 Tesla V100.

Nonetheless, the pre-trained model still lacks the understanding of semantics on the screen carried by UI elements. For example, ICON_HEART is an icon with heart shape, but can represent different meaning of "liking" or "adding to favorite" on different screens. Consequently, a two-stage fine-tuning scheme is adopted. Table 1 summarizes the training data. In stage one, UI-Hawk includes a broad range of screen-related single-step tasks to obtain the basic screen understanding capabilities. The training sequence contains multiple images, with format "`[img1] question answer [img2] question answer ...`". The question-answer pairs are sampled from different single-step tasks to enable flexibly switch between screen streams. In stage two, we utilize sequential GUI navigation tasks as the training data, enabling UI-Hawk to learn to deal with screen streams based on user instructions and execution history. The input sequence contains the history screens, history actions, current screen and user instructions. UI-Hawk is required to output the correct action API (see Appendix C.2). These GUI navigation tasks are bilingual and are detailed in Section 3.3. The entire fine-tuning takes 3 days on 32 Tesla V100. We kindly refer readers to Appendix B for details.

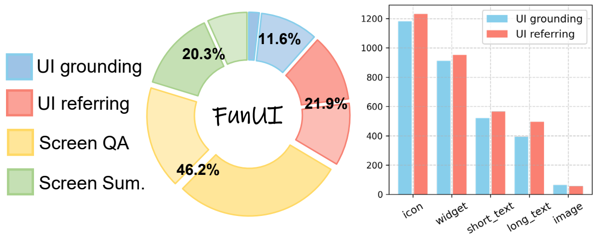
3 Dataset and Task Formulation

In this section, we demonstrate the process of generating tasks and dataset for model training and evaluation. In Section 3.1, we detail the screen data collection. While in Section 3.2, we explain how we formulate the screen-related tasks. In Section 3.3, we demonstrate the sequential navigation tasks used to train model as a GUI agent.

3.1 Data Collection

Mobile Screens It is essential to assemble a diverse range of mobile screens to obtain screen understanding ability. For Chinese screens, following (Wu et al., 2023), we use an automated traversal tool to crawl screens from more then 420 apps, sorted by download counts in the app market. We filter the duplicated screens, with methods detailed in Appendix A.1. As a result, we gather 115k unique Chinese images in total (113k for training and 2k remains for evaluation). For English screens, we use the widely adopted RICO dataset (Deka et al., 2017), which serves as the image foundation for several screen-related tasks (Li et al., 2020; Bai et al., 2021; Hsiao et al., 2022; Wang et al., 2021). In total, there are 72k images (63k for training and 9k for evaluation).

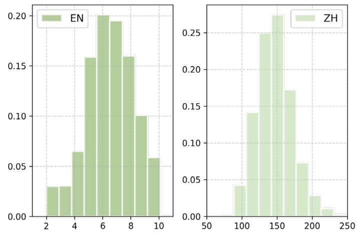
Screen Annotations Detecting UI elements on the screen is crucial for data construction (Baechler et al., 2024; You et al., 2024). We find existing UI detection models have some deficiencies (See Appendix A.1 for more details). Therefore, we manually collected 270k UI element detection annotations for both Chinese and English mobile screens and train an RT-DETR (Zhao et al., 2024) based UI detection model. Our model is responsible for detecting basic UI elements covering ICON (133 types, extended from (Sunkara et al., 2022)), TEXT, IMAGE, INPUT_FIELD and KEYBOARD. Similar to previous works (You et al., 2024; Fan et al., 2024), we group basic elements into associated items, namely high-level widgets. Since screen annotations enable textual representation of screens (Baechler et al., 2024), we refer the task of generating such annotations solely based on the input image as *screen annotation* task, used as a pre-training task mentioned in Section 2.2. An example of screen annotation is shown in Figure 6(c).



(a) Task distribution (b) Types of UI elements

Category	#sample
Attribution	2418
Identification	1502
Relation	1137
Location	1036
Operation	1110
Time	862
Numerical	774
Function	623
Judgement	426

(c) Diverse Screen QA pairs



(d) Lengths of Screen Summarization

Figure 4: **Statistics of FunUI Benchmark.** (a) Distributions of four fundamental tasks. The deep and shallow color represents for English and Chinese, respectively. (b) Various UI types included. (c) Diverse categories of screen QA pairs. Note that these categories are not mutually excluded. (d) The annotated summarization lengths.

3.2 Fundamental Tasks

The process of GUI screen streams can be divided into several minor steps (Zhang et al., 2024b), including describing the screen, referring to the target UI elements and generating the corresponding action coordinates. Hence, we identify from previous works (Bai et al., 2021; Li et al., 2020; Hsiao et al., 2022; Wang et al., 2021) that four fundamental one-step tasks are crucial for screen stream understanding, as shown in Figure 1. Specifically, these fundamental tasks includes:

- **UI Referring**: This task requires the model to describe the UI element based on its position on the screen, emphasizing the understanding of the functionality and semantics of UI elements.
- **UI Grounding**: UI Grounding task measures the regional localization capability. The model is required to accurately locate UI elements based on the instructions (i.e. output bounding boxes).
- **Screen Question-Answering**: For this task, the model has to answer questions related to element relationships. We categorize the questions into nine major types, detailed in Appendix A.2.
- **Screen Summarization**: This task involves summarizing the main contents or functions of the screen with several sentences.

In short, we collect samples for these fundamental tasks via two methods: For English screens, which already have widely recognized datasets, we simply employ the corresponding dataset for each task to keep consistent with previous models (You et al., 2024; Baechler et al., 2024). For Chinese screens, we utilize the screen annotation generated with our UI detection model to prompt GPT-4V and generate corresponding question-answer pairs. Figure 5 summarizes the data collection pipeline. After generation, we conduct manual check to ensure the correctness of these samples. The used prompts, more visualized examples for each task and other details can be found in Appendix A.2.

3.3 GUI Navigation Tasks

To fairly evaluate the screen stream processing ability, two GUI navigation datasets are selected for English and Chinese mobile screens, respectively.

GUI-Odyssey+ GUI-Odyssey (Lu et al., 2024) is a comprehensive dataset for evaluating GUI navigation agents on cross-app tasks, comprising of 7,735 navigation episodes from six categories of apps. Within GUI-Odyssey, the click events are recorded by coordinates (x, y). As shown in Figure 3, such representation hinders the precise evaluation of click actions. To tackle with the problem, we augment the click event annotations via the bounding boxes of the corresponding UI elements recognized by our UI detection model. The augmented dataset is called GUI-Odyssey+.

GUI-Zouwu There is a lack of GUI navigation episodes collected for Chinese mobile phones, whose screen layout is vastly different from English mobile devices. Therefore, we manually collected 3232 episodes, resulting in the first large-scale Chinese GUI dataset, GUI-Zouwu. GUI-Zouwu spans 137 apps from 6 daily scenarios, including trip (34.2%), shopping (18.3%), medical (15.5%), social (15.0%), locallife (9.6%) and message (7.3%). For a detailed collection process of the data, please refer to Appendix A.3. In consistent with GUI-Odyssey+, the click events in GUI-Zouwu are annotated by the bounding box of UI elements.

4 *FunUI* Benchmark

The evaluation of the UI understanding capabilities of MLLMs remains a open question. The main challenge lies in the lack of consensus on what constitutes the fundamental aspects of UI understanding, as well as the absence of a comprehensive benchmark that jointly captures the diverse types

Model	Tool	Information	Shopping	Media	Social	Multi-Apps	Overall	ClickAcc
GPT-4V*	10.6	9.8	11.2	7.6	5.0	11.2	9.2	3.4
GPT-4O*	37.2	29.9	28.9	28.5	27.2	25.1	29.5	13.2
CogAgent	12.9	10.0	14.2	10.5	9.0	8.4	10.3	7.5
SeeClick	6.8	6.4	5.8	7.2	8.1	5.5	6.5	6.5
OdysseyAgent	81.5	63.6	62.2	72.5	72.5	68.8	70.8	43.8
UI-Hawk	**88.2**	**70.9**	**66.8**	**82.4**	**81.4**	**80.1**	**79.4**	**76.3**

Table 2: **Sequential navigation performance on GUI-Odyssey+ dataset.** We report the overall action matching score on six categories of navigation tasks, including tool, information, shopping, media, social and multi-apps, and the overall action matching score. "ClickAcc" stands for the accuracy of click actions, reflecting the grounding ability of models. *Due to the budget limit, we randomly sampled 500 instances for each task category for evaluation.

of screen elements and tasks. Prior works typically introduced the evaluation tasks independently to assess narrow abilities such as grounding (Bai et al., 2021) or captioning (Li et al., 2020).

To address this gap, we introduce *FunUI*, a bilingual evaluation benchmark that unifies and standardizes four fundamental UI understanding tasks that are previously studied in isolation: *UI grounding* and *UI referring* tasks are designed to access the regional location and identification abilities of models, whereas *screen question answering* and *screen summarization* tasks require more integrated analysis of the screen contents. We posit that these four capabilities form the essential building blocks for robust GUI navigation agents. Concretely, *FunUI* distinguishes with previous benchmarks (Hsiao et al., 2022; Li and Li, 2022; Cheng et al., 2024) on three key aspects:

- **Bilingual**: *FunUI* comprises of 2150 Chinese screens and 9347 English screens from Android devices, annotated with 14k and 18k samples, respectively. To the best of our knowledge, this is the first benchmark that enables the assessment of Chinese UI understandings.
- **Comprehensive**: Instead of concentrating on a single aspect, *FunUI* includes different evaluation dimensions of UI understanding, ranging from fine-grained UI grounding and UI referring, to complicated screen question answering and screen summarization.
- **Diverse**: *FunUI* covers various types of question answering pairs, including grounding and referring questions about 120+ icons and widgets, and complex questions with related to elements relations and arithmetics. This is more challenging for models to answer than text-related tasks used in GUICourse (Chen et al., 2024b).

To ensure reliable evaluation under real scenarios, *FunUI* is carefully crafted: (1) For English screens, we meticulously select the union of test images from authoritative dataset, i.e. WidgetCaption (Li et al., 2020), RefExp (Bai et al., 2021), ScreenQA (Hsiao et al., 2022) and Screen2Words (Wang et al., 2021), so that models trained for English screens could be consistently compared with previous SOTA methods, i.e. Ferret-UI (You et al., 2024). (2) For Chinese screens, we recruited experienced annotators to label the questions along with the bounding boxes of related UI elements, enforcing the samples to be novel and excluded in existing resources. Details about the construction of *FunUI* benchmark can be found in Appendix A.4. The basic statistics of *FunUI* are illustrated in Figure 4.

Consequently, by systematically organizing these tasks as fundamental UI understanding capabilities, we further demonstrate their individual contributions and synergistic impact for downstream GUI navigation (see Table 5 in Section 5.3).

5 Experiments

5.1 Experimental Setup

Baselines We adopt different types of MLLMs as the baselines: (1) the proprietary GPT-4V (OpenAI, 2023) and GPT-4O (OpenAI, 2024) (2) the open-source models like Qwen-VL-Chat (Bai et al., 2023) and InternVL2-8B (Chen et al., 2024c), (3) models specifically designed for GUI tasks, including MLLMs for screen understanding like Spotlight (Li and Li, 2022), Ferret-UI (You et al., 2024) and SeeClick (Cheng et al., 2024), and MLLMs targeted for GUI navigation like CogAgent (Hong et al., 2024) and OdysseyAgent (Lu et al., 2024). Since currently all UI-specific models are trained under English contexts, we compare UI-Hawk with three generalist MLLMs that could understand Chinese (GPT-4V, Qwen-VL-Chat and InternVL2-8B).

Model	GRD	REF	SQA	SUM
	Acc	CIDEr	F1	CIDEr
GPT-4V*	2.3	23.5	74.7	34.8
Spotlight†	–	141.8	–	106.7
Ferret-UI†	–	140.3	–	**115.6**
SeeClick	29.6	–	28.3	102.3
UI-Hawk	**63.9**	**144.3**	**85.9**	106.5

(a)Results on English screens.

Model	GRD	REF	SQA	SUM
	Acc	CIDEr	F1	GPT
GPT-4V*	2.0	5.5	52.4	60.8
Qwen-VL	2.2	1.3	45.7	47.3
InternVL2	–	14.5	**60.4**	77.9
UI-Hawk−	63.9	62.3	50.9	78.7
UI-Hawk	**67.6**	**66.2**	53.6	**79.5**

(b)Results on Chinese screens.

Table 3: **Performance of UI understanding on *FunUI* benchmark.** *GRD*: grounding, *REF*: referring, *SQA*: screen question answering, *SUM*: screen summarization. *Due to the budget limit, we randomly sampled 500 samples for each task for evaluation. †Performance of the close-source model from its original paper.

Model	UI-PT	UI-SFT	GRD	NAV
TextHawk	×	×	18.0	71.6
TextHawk+UI	×	✓	54.7	75.9
UI-Hawk-Naive	✓	×	33.8	75.7
UI-Hawk	✓	✓	**63.9**	**79.4**

Table 4: **Ablation study on the effect of UI-related training phrases.** As TextHawk is pre-trained purely on document-related tasks, we label × on the "UI-PT" column to distinguish it with our pre-training that involves screen annotation data. The accuracy of English UI grounding task (GRD) and the overall action matching score on GUI-Odyssey+ dataset (NAV) are reported.

Evaluation Metrics For fundamental tasks, we use the accuracy computed at IoU=0.5 for UI grounding, SQuAD-F1 score (Hsiao et al., 2022) for screen question answering, and CIDEr for UI referring. With regard to screen summarization, we utilize CIDEr for English evaluation and GPT-4O as the judger for Chinese evaluation, since the annotated Chinese screen summarizations are longer and more complicated. For GUI navigation, we employ the widely used action matching score as the metric (Zhan and Zhang, 2023; Rawles et al., 2024; Lu et al., 2024). Details are in Appendix C.

5.2 Main Results

Screen Understanding Table 3 demonstrates the performance of UI-Hawk compared with previous state-of-the-art models on various screen understanding tasks. On English screens, compared to Spotlight and Ferret-UI, UI-Hawk possesses superior results in UI referring and screen question-answering. Compared with SeeClick, UI-Hawk exhibits better performance on grounding, even though SeeClick uses 320k English screenshots for training. Although UI-Hawk slightly falls short on screen summarization, the results are still competitive. Since there is a lack of Chinese UI-specific models, we compare UI-Hawk with GPT-4V and Qwen-VL. We additionally include a minor version of UI-Hawk, UI-Hawk-Minus, which is fine-tuned on a total of 128k Chinese samples, where each fundamental task accounts for 32k samples. As shown in Table 3(b), even UI-Hawk-Minus surpasses Qwen-VL and InternVL2 on grounding and referring by a large margin, and it achieves on-par performance with GPT4V in screen question answering. This underscores the scarcity of UI-related information in general data, proving the significance of constructing such training samples to acquire the domain-specific knowledge. Overall, Table 3 suggests that UI-Hawk is a bilingual model with advanced screen understanding capabilities.

GUI Navigation We follow the evaluation methods used by CogAgent (Hong et al., 2024), SeeClick (Cheng et al., 2024) and GUI-Odyssey (Lu et al., 2024) to assess the performance of UI-Hawk under in-domain settings. As shown in Table 2, SeeClick performs poorly, as it only predicts the "CLICK" actions and does not generalize well to GUI-Odyssey+. UI-Hawk significantly outperforms all other models, achieving a 9% absolute increase in overall action matching score and a 32.5% absolute increase in the prediction accuracy of click operations compared to the most capable OdysseyAgent. The results validate that UI-Hawk represents a state-of-the-art GUI agent.

5.3 Ablation Studies

The Impact of Training Strategy To further illustrate the validity of the collected data and associated training strategy, highlighting the differences between UI-Hawk and TextHawk (Yu et al., 2024), we conducted ablation experiments on training phases, where "UI-PT" represents the pre-training that involves the curated screen annotation dataset and "UI-SFT" refers to the stage one screen understanding fine-tuning. Since the ability for se-

Line	Grounding		Referring		ScreenQA		Screen Sum.		History	GUI-Odyssey+		GUI-Zouwu	
	EN	CN	EN	CN	EN	CN	EN	CN		Overall	ClickAcc	Overall	ClickAcc
(1)									T	71.7	66.9	41.2	49.3
(2)									V	75.7	71.9	44.8	56.5
(3)	✓	✓							V	77.8	74.5	46.1	59.2
(4)			✓	✓					V	77.7	73.9	46.0	58.4
(5)					✓	✓			V	77.6	73.6	46.5	58.4
(6)							✓	✓	V	77.3	73.1	45.6	57.7
(7)	✓	✓	✓	✓	✓	✓	✓	✓	T	72.7	68.1	43.3	55.6
(8)	✓	✓	✓	✓	✓	✓	✓	✓	V	78.3	75.1	45.9	58.4
(9)	*Full Data*								V	**79.4**	**76.3**	**47.9**	**61.4**

Table 5: **Ablation study on the effect of fundamental UI tasks and different history representations.** "T" and "V" represents textual and visual history, respectively. Following (Lu et al., 2024), the default history length is set as 4 across all experiments. A "✓" indicates we sample 32k examples for that language from the corresponding task, resulting in a total of 64k samples across both languages as the training data. For English grounding tasks, we repeat the original 16k training samples to 32k for a fair comparison.

quential GUI navigation must be acquired through training on corresponding data, all these ablation models underwent the stage-two fine-tuning, which we omit in the table for brevity.

We compare UI-Hawk with the original TextHawk model, TextHawk+UI model that is continually trained with our two-stage fine-tuning scheme, and UI-Hawk-Naive that only undergoes our UI-related pre-training phrase. Results on Table 4 demonstrate that: (1) screen annotation task is beneficial for structural understanding of screens; (2) our proposed four fundamental screen understanding tasks are crucial for the enhancement of both grounding and navigation capabilities.

The Impact of Fundamental Screen Understanding To further investigate the influence of each fundamental task towards the final navigation performance, we randomly sample 64k data for each task to conduct the stage one fine-tuning. As shown in Table 5, each fundamental task contributes to the improvement of navigation performance, within which UI grounding task influences the prediction of click operations most. The model trained on the averagely mixed data (line 8) has outstanding performance on GUI-Odyssey+ but is marginally inferior to the model trained solely on screen question answering (line 5) on GUI-Zouwu. We attribute this to the Chinese screen summarization task, as in line 6 its positive influence is minimal. We finally build UI-Hawk with all collected samples as the training data, which excels in both English and Chinese GUI navigation tasks. These results validate the significance of enhancing screen understanding in the development of autonomous GUI agents.

The Impact of Screen Streaming History modeling of sequential decision-making tasks has long been a problem, especially for MLLMs with limited context windows. To gain a deep insight on the effect of screen streaming encountered during navigation, we further conduct an ablation study on using plain text-based historical actions only, or using screen sequences of historical screenshots together with historical actions. The results presented in Table 5 indicate that visual history information has an essential impact on GUI navigation. Such impact is much more significant than the impact brought by fundamental screen abilities, demonstrating that screen stream understanding is not only beneficial but also essential for GUI navigation.

6 Related Works

Automatic execution of user instructions on mobile devices has been a trend. Early works (Shi et al., 2017; Deka et al., 2017; Liu et al., 2018) concentrated on synthetic web or mobile screens. Later, datasets are collected on real webs and apps (Burns et al., 2021; Sun et al., 2022; Deng et al., 2024; Lu et al., 2024) and further scaled-up to facilitate the training (Rawles et al., 2024; Chen et al., 2024a). Recent progress in this area are dominated by proprietary MLLM-based agents (Yang et al., 2023; He et al., 2024), relying on visual prompting (Yan et al., 2023; Zheng et al., 2024), complex context modeling (Zhang et al., 2024b,a) or self-refine (Kim et al., 2024) capabilities of language models to generalize on user interfaces. The lack of screen-related training make such agents struggles with grounding to correct UI elements (Yan et al., 2023; Zheng et al., 2024), even with the view hierarchy or other annotations as additional inputs (Wen et al., 2023; Wang et al., 2024). To deal with this problem, this work constructs a universal GUI agent UI-Hawk by customizing a open-sourced MLLM on multiple fundamental screen-related tasks aimed for better screen understanding.

Since screen understanding is significant (Bai et al., 2021; Zhang et al., 2021; Venkatesh et al., 2022), several works utilize open-sourced MLLMs as the foundation and fine-tunes the model on partial aspects of screen-related tasks (Li and Li, 2022; Jiang et al., 2023; Hong et al., 2024; Cheng et al., 2024). However, these models only takes text-based history, overlooking the information carried by historical screen streams. To bridge the gap, we propose to integrate the screen stream processing capability into GUI agents. Through a history-aware visual encoder and an efficient resampler, UI-Hawk achieves state-of-the-art performance on GUI navigation by using screen streams as input.

7 Conclusion

In this paper, we introduced UI-Hawk, a GUI agent focused on screen stream understanding. Leveraging the efficient architecture to tackle with screen streams, UI-Hawk excels in four fundamental screen understanding tasks, including *UI grounding*, *UI referring*, *screen question answering*, and *summarization*. For a comprehensive assessment, we established the bilingual FunUI benchmark to evaluate the screen comprehension of MLLMs. Extensive experiments demonstrates that UI-Hawk sets new state-of-the-art performance on GUI navigation tasks, highlighting the importance of robust screen understanding for autonomous GUI agents.

Limitations

As layout styles evolve, general knowledge of UI elements remains transferable. Since there exists numerous widly-adoped high-quality annotations on RICO datasets, we utilize these data to construct the basic UI tasks for English screen. Our ablation study (see Table 5) shows that such transferable UI knowledge improve the performance on GUI-Odyssey+ dataset, whose screens are newly collected. However, to build a practically reliable GUI agent in real life, it is still essential to have updated screens and apps as training data. We leave it for future work to collect more training samples on up-to-date English apps, and explore the long-term effect to understand whether UI-Hawk could adapt to evolving GUI designs.

Acknowledgments

The research is supported by National Natural Science Foundation of China (Grant No. 62176058) and National Key RD Program of China (Grant No. 2023YFF1204800). The project's computational resources are supported by CFFF platform of Fudan University.

References

Amos Azaria, Rina Azoulay, and Shulamit Reches. 2024. Chatgpt is a remarkable tool—for experts. *Data Intelligence*, 6(1):240–296.

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615*.

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. 2021. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2021. Mobile app tasks with iterative feedback (motif): Addressing task feasibility in interactive visual environments. *arXiv preprint arXiv:2104.08560*.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, et al. 2024a. Gui-world: A dataset for gui-oriented multimodal llm-based agents. *arXiv preprint arXiv:2406.10819*.

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

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. 2024c. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024.

Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854.

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

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338.

Yue Fan, Lei Ding, Ching-Chen Kuo, Shan Jiang, Yang Zhao, Xinze Guan, Jie Yang, Yi Zhang, and Xin Eric Wang. 2024. Read anywhere pointed: Layout-aware gui screen reading with tree-of-lens grounding. *Preprint*, arXiv:2406.19263.

Shirin Feiz, Jason Wu, Xiaoyi Zhang, Amanda Swearngin, Titus Barik, and Jeffrey Nichols. 2022. Understanding screen relationships from screenshots of smartphone applications. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*, pages 447–458.

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

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

Yu-Chung Hsiao, Fedir Zubach, Maria Wang, et al. 2022. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yue Jiang, Eldon Schoop, Amanda Swearngin, and Jeffrey Nichols. 2023. Iluvui: Instruction-tuned language-vision modeling of uis from machine conversations. *arXiv preprint arXiv:2310.04869*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36.

Gang Li and Yang Li. 2022. Spotlight: Mobile ui understanding using vision-language models with a focus. *arXiv preprint arXiv:2209.14927*.

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. 2022. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. 2020. 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)*, pages 5495–5510.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*.

OpenAI. 2023. Gpt-4v.

OpenAI. 2024. Gpt-4o.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36.

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

Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022. Meta-gui: Towards multi-modal conversational agents on mobile gui. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6699–6712.

Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Abhanshu Sharma, James Stout, et al. 2022. Towards better semantic understanding of mobile interfaces. *arXiv preprint arXiv:2210.02663*.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Sagar Gubbi Venkatesh, Partha Talukdar, and Srini Narayanan. 2022. Ugif: Ui grounded instruction following. *arXiv preprint arXiv:2211.07615*.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 498–510.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *Preprint*, arXiv:2401.16158.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2023. Empowering llm to use smartphone for intelligent task automation. *arXiv preprint arXiv:2308.15272*.

Jason Wu, Rebecca Krosnick, Eldon Schoop, Amanda Swearngin, Jeffrey P Bigham, and Jeffrey Nichols. 2023. Never-ending learning of user interfaces. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13.

An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. 2023. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

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

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Guohai Xu, Chenliang Li, Junfeng Tian, Qi Qian, Ji Zhang, et al. 2023. Ureader: Universal ocr-free visually-situated language understanding with multimodal large language model. *arXiv preprint arXiv:2310.05126*.

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

Ya-Qi Yu, Minghui Liao, Jihao Wu, Yongxin Liao, Xiaoyu Zheng, and Wei Zeng. 2024. Texthawk: Exploring efficient fine-grained perception of multimodal large language models. *Preprint*, arXiv:2404.09204.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986.

Zhuosheng Zhan and Aston Zhang. 2023. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*.

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

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024b. Android in the zoo: Chain-of-action-thought for GUI agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031, Miami, Florida, USA. Association for Computational Linguistics.

Xiaoyi Zhang, Lilian De Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, et al. 2021. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Zhizheng Zhang, Wenxuan Xie, Xiaoyi Zhang, and Yan Lu. 2023. Reinforced ui instruction grounding: Towards a generic ui task automation api. *arXiv preprint arXiv:2310.04716*.

Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. 2024. Detrs beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*.

A Data Collection

Here we provide the details about our data collection process. In Section A.1, we demonstrate how we collect screen annotations and convert it into a pre-training task. In Section A.2, we provide the details about the prompting of GPT-4V to generate samples. In Section A.3, we illustrate the design of GUI-Zouwu and the data collection process.

A.1 Screen Annotation

Screens collected by automated traversal often have a high degree of repetition (Feiz et al., 2022). We employ a pixel-wise filtering algorithm combined with screen structure to eliminate duplicate images.

Screen Filtering We observe that the screenshots collected through automated traversal often contain many duplicates, as clickable elements do not always lead to a page transition. Therefore, we utilize a two-step filtering algorithm to remove the duplicates. We first perform a pixel-wise check of the images, with the goal of filtering out identical screens to reduce the computational load on subsequent algorithms. Then, we use our trained RT-DETR model to detect the UI elements, thereby extracting the structural information of the screen. We define a screenshot as duplicated if its structure remains unchanged while unimportant content, such as carousel images or advertisements, varies. Therefore, we mask regions in the screen that are labeled as “IMAGE” and then perform the pixel-wise comparison of the masked images. About 15% of the screens are filtered.

Screen Annotation We find existing UI detection models have some deficiencies. Recent open-sourced UI element detection models (Sunkara et al., 2022; Fan et al., 2024) have severe issues including inaccurate bounding boxes and missed detections. As shown in Figure 6(a), IconNet detects bounding boxes that are smaller than the actual elements, and it misses detecting the app icons for Photos and YouTube. Moreover, these models are trained on English data, hence perform poorly on Chinese mobile screens. Therefore, we build our own RT-DETR model by manually collecting 270k bounding boxes from both Chinese and English screens. These annotations were manually annotated by human annotators, with each annotation individually created and subsequently reviewed by our data quality team, achieving an average recall of 95% for various UI elements. An illustration example is shown in Figure 6(b). Following (Baechler et al., 2024), we construct the textual representation of a screen by considering the containment relationships between the elements. See Figure 6(c) for an example. We define the *screen annotation* task, which requires the model to generate the structured textual representation of screens by taking the image solely as the input.

A.2 Fundamental Tasks

The ability to understand screen streams is built upon the understanding of individual screens. Therefore, we designed four fundamental tasks to help the model comprehend screen contents.

UI Referring This task requires the model to describe the UI element based on its position on the screen, emphasizing the understanding of the functionality and semantics of UI elements. For English screens, we utilize the open-sourced dataset Widget Caption (Li et al., 2020). For Chinese screens, we distinguish the data by the UI types, where question-answer pairs related to ‘TEXT’ elements (i.e. OCR) are generated by templates and others are generated by prompting GPT-4V. We finally construct 600k referring samples.

UI Grounding This task measures the regional localization capability. The model is required to accurately locate UI elements based on the instructions. For English screens, Referring Expression (Bai et al., 2021) dataset is used. For Chinese screens, since grounding is the reverse process of referring, we utilize GPT-4-Turbo to rewrite the referring question-and-answer pairs as the grounding data, resulting in 580k Chinese grounding samples.

Screen question answering For this task, the model has to answer questions related to element relationships. Specifically, we categorize the questions into nine major types, which are:

- Identification Questions: These involve queries about what something is, such as “What is the doctor’s name?" when presented with a doctor’s information, but not directly telling you that this person is a doctor.
- Attribution Questions: Such questions involve associated attributes of screen elements, such as

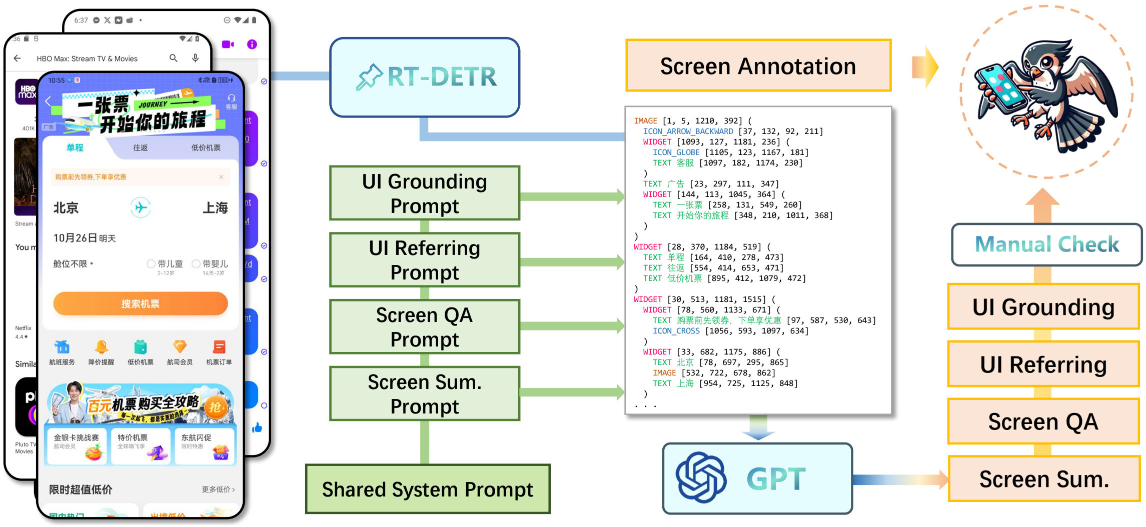


Figure 5: **Overall data collection pipeline.**

“What is the rating of the xxx?" and “Who is the author of the book yyy?".

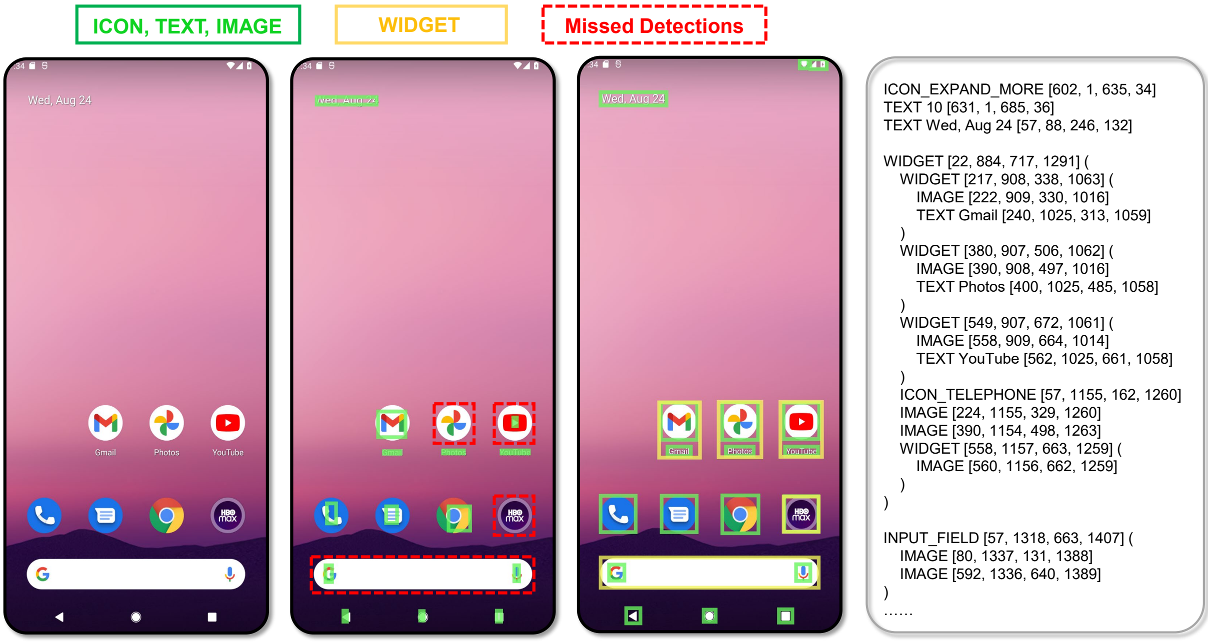
- Relationship questions: These include comparisons between two or more screen elements, such as “Which one has a higher price, A or B?" or “Which shopping market is the farthest away?"
- Localization questions: These questions provide a detailed description of a specific screen element and then ask about its location on the screen, such as “Where is the 2019 MacBook Air product located on the screen?"
- Operation Questions: These questions involve operations on the screen, such as “How to open the shopping cart?"
- Temporal Questions: Any questions related to time or date fall into this category, such as “What is the current time on the screen?" or “What are the departure and arrival dates of the flight?"
- Numerical Questions: These contain any questions related to numbers or calculations, such as “How many items are there in the cart?" or “What is the lowest price of the science fictions?"
- Judgement Questions: These questions involve making yes/no or true/false determinations. For example, “Is it possible to upgrade to VIP? "

For English screens, we employ the Google ScreenQA datset (Hsiao et al., 2022). For Chinese screens, we prompt GPT-4V as a good expert (Azaria et al., 2024) by faked in-context samples to generate question-answer pairs corresponding to these major categories. In total, we obtain 1200k Chinese samples.

Screen Summarization This task involves summarizing the main contents or functions of the screen. Specifically, for English screens, existing Screen2words (Wang et al., 2021) dataset is applied to maintain a fair comparison with previous SOTA models (You et al., 2024). For Chinese, we employ GPT-4V to concisely describe the screen within three sentences. Around 50k Chinese screen descriptions are annotated by GPT-4V.

As shown in Figure 4(d), the generated Chinese summarizations are longer than English ones, making them less suitable for evaluation using CIDEr. Therefore, we utilize GPT-4O as the judger and score the response from four different perspectives.

Figure 5 summarizes the data collection pipeline. The prompts we used are summarized in Figure 8 and Figure 9. Note that due to the poor recognition ability of GPT-4V on Chinese characters, we include detected screen annotations as additional input to reduce the hallucinations during data generation. Apart from the referring text data generated by templates, we conduct manual verification for all sample pairs. We recruit around 50 annotators, and allocate data for each annotator on a per-image basis. Annotators are required to verify the model-generated answers and correct the errors for each assigned image. Once the human annotation is completed, our data quality team conducts acceptance checks. Specifically, in each round, 10% of the images are sampled for inspection. If the accuracy of the sampled data exceeds 95%, it passes; otherwise, the data undergoes a second round of annotation. The average number of annotation-verification rounds per image is 2.6.



Raw Screen **(a) IconNet** **(b) Our RT-DETR** **(c) Screen annotation**

Figure 6: **Comparison of different detection models.** (a) The detection outcomes from IconNet (Sunkara et al., 2022). (b) The detection outcomes from RT-DETR model trained by us. (c) Corresponding screen annotation.

A.3 GUI-Zouwu

To evaluate the influence of screen streaming in Chinese mobile devices, we construct GUI-Zouwu dataset. We first identify six major scenarios from daily life, involving trip, shopping, medical, social, locallife and message. We collect data from the top apps involved in each scenario. For each scenario, instead of using predefined task templates, we instruct annotators to first explore the app and then create tasks based on the functionalities the app can perform. This approach ensures the diversity of tasks. Once the tasks are defined, we ask the annotators to complete the tasks based on the given instructions. The data quality team then checks the accuracy of the collected sequences and the quality of the task instructions. Finally we obtain 3232 instruction-episode pairs, covering 137 apps, with an average of 20 navigation episodes per app and 15 apps per scenario.

A.4 FunUI Benchmark

As we have mentioned in Section 4, we build *FunUI* benchmark by two methods:

- For English screens, we carefully select the union of test images from authoritative dataset (Li et al., 2020; Bai et al., 2021; Hsiao et al., 2022; Wang et al., 2021). And we have removed some repeated images in the union set from the training split of each dataset to ensure that no data contamination. We did so for two reasons: (1) *consistency of evaluation*: Models trained for English screens could be consistently compared with previous SOTA methods, i.e. Ferret-UI (You et al., 2024). (2) *leveraging existing resources*: We observed that although the difficulty of these authoritative datasets are moderate, the performance of models were suboptimal (see Table 3), especially on UI grounding and screen question answering tasks. This indicates that the potential of these datasets has not been fully explored. Therefore, we believe that these datasets still hold significant value and are worth utilizing as evaluation data. This is also an environmentally friendly approach that reduces resource consumption.
- For Chinese screens, we recruited experienced annotators to label the questions along with the bounding boxes of related UI elements, enforcing the samples to be novel and excluded in existing

Task	EN	CN
UI Grounding	565	3124
UI Referring	3621	3369
Screen QA	9186	5525
Screen Sum.	4310	2150

Table 6: **#Samples of each fundamental task in FunUI benchmark.** "Screen QA" and "Screen Sum." represent screen question answering task and screen summarization task, respectively.

resources. Annotators are also required to exclude the questions and screens that might lead to privacy leakage. Specifically, we selected 10 annotators with the highest accuracy from the pool of workers who labeled the training data of four fundamental tasks (we have mentioned in Appendix A.2). We required annotators to follow the prompts in Figure 8 and Figure 9, while also taking into account human usage habits and the primary functionality of the current screen. As these data are used for evaluation, our data quality team reviewed all the samples and retained only the correct ones, resulting in 14k samples.

The detailed statistics for each evaluation task are presented in Figure 4 and Table 6. Visualized examples are shown in Figure 7.

B Training Details

During the whole training, we adopt AdamW optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0.05. All the training is conducted on Tesla V100 GPUs.

Pre-Training For pre-training, we utilize images of various sizes and aspect ratios. As we employ SigLIP-SO (Zhai et al., 2023), each sub-image is sized at 224×224. The language backbone is Qwen2-7B (Yang et al., 2024). We employ an effective batch size approaching 1500 and we train UI-Hawk for about one epoch on a data mixture of screen annotation data and other document-related data used in (Yu et al., 2024). The resampler, the LoRA for ViT and LLM, and the randomly initialized detection head are updated. The learning rate is linearly increased to 1.5×10^{-4} during the first 3% of steps, then gradually decays to 5×10^{-6} following a cosine schedule.

Supervised Fine-Tuning During fine-tuning, we integrate LoRA weights into the LLM and jointly train the entire model, excluding the visual encoder. At stage one, we set the context length as 2048 and fine-tune the model on four fundamental screen tasks for one epoch, using a batch size of 256. At stage two, we adapt the model to sequential tasks by increasing the context length to 4096. The batch size is set as 64. During each fine-tuning stage, the learning rate is linearly increased to $2e^{-5}$ at the beginning 3% of steps, then gradually reduced to 0 using cosine decay schedule.

C Evaluation Details

C.1 Fundamental Tasks

UI Grounding Previous work often employ a relatively low IoU threshold, such as IoU = 0.1 (Baechler et al., 2024), when evaluating grounding tasks. However, in the field of object detection, setting the IoU threshold at 0.5 is more widely used (Everingham et al., 2010). This stricter standard prevents the exaggeration of the performance (as shown in Table 8).

UI Referring For referring, we apply the CIDEr metric (Vedantam et al., 2015) as (Li et al., 2020) has done, as this task is relatively straightforward and the responses are typically one sentence long.

Screen QA Following (Hsiao et al., 2022), we utilize the SQuAD-F1 score as the evaluation metric. For a specific question, we compile all candidate answers, whether they are long or short, into a reference list. The model's response is then compared to this list to calculate the score.

Screen Summariztion As dipicted in Figure 4, there is a significant difference in the length distribution between Chinese and English summarizations, where English length measured by words and Chinese by characters. Hence, for English summarizations, we use the CIDEr metric as (Wang et al., 2021). For Chinese summarizations, we employ GPT-4O as the judger to score the responses from the following four aspects: (1) Content: Assesses how well the summary captures the main content and functionality of the screen; (2) Structure: Judges the accuracy in reflecting the layout and structure of the screen; (3) Fluency: Evaluates the naturalness and readability of the generated text; (4) Authenticity: Measures whether the summarizations is truthful and free from hallucinations. We instruct GPT-4O to assign a score between 0 and 10 for each aspect, and compute the final score as an average of these four scores multiplied by 10. The detailed scores can be found in Table 7.

C.2 GUI Navigation

Action Space Following (Zhang et al., 2024b), we unify the action space into 5 kinds of actions: CLICK, SCROLL, TYPE, PRESS and DONE:

- **`CLICK(bbox=[x1, y1, x2, y2])`**: click (including long press) the UI element whose exact bounding box is `[x1, y1, x2, y2]`.

	Content	Structure	Fluency	Authenticity	GPT-Score
GPT-4V*	5.52	5.83	7.64	5.34	60.8
Qwen-VL	4.13	4.32	6.44	4.02	47.3
InternVL2-8B	7.34	7.50	8.41	7.91	77.9
UI-Hawk-Minus	7.45	7.71	8.53	7.79	78.7
UI-Hawk	7.51	7.84	8.60	7.85	79.5

Table 7: **Details of the evaluation for Chinese screen summarization.** We evaluate from four perspectives: content(0-10), structure(0-10), fluency(0-10) and authenticity(0-10). The final GPT-Score is $10\times$ the average score.

(en)	GPT-4V	SeeClick	UI-Hawk
IoU=0.1	27.4	62.1	85.5
IoU=0.5	2.3	29.6	63.9
$\vert\Delta\vert$	25.13	32.57	21.59

Table 8: **Impact of IoU thresholds on *grounding* accuracy.** Obviously, a low IoU threshold exaggerates the model's performance, especially for those models with inaccurate bounding box predictions.

ViT	LLM	GRD-zh	GUI-Zouwu
SigLip-SO	InternLM1.0	63.8	45.3
SigLip-SO	Qwen2	67.6	47.9
SigLip-SO	InternLM2.5	67.9	47.4

Table 9: **Influence of difference language backbones.**

Unfreeze ViT	GRD-zh	GUI-Zouwu
×	62.5	42.0
✓	67.9	47.4

Table 10: **Comparison between freezing and unfreezing the ViT during pre-traning.**

- `SCROLL(direction="up|down|left|right")`: swipe the screen to a specified direction.
- `TYPE(text="...")`: type text with keyboard.
- `PRESS(button="home|back|recent")`: press the system level shortcut buttons provided by Android OS. "press home" means directly going to the home screen, "press back" means moving to the previous screen, "press recent" means jumping to the most recent app.
- `DONE(status="complete|impossible")`: stop and judge whether the task has been completed.

Metrics We utilize the action matching score (Rawles et al., 2024; Zhan and Zhang, 2023) to evaluate the action prediction accuracy. An action is considered correct if both the action type and the details (i.e. scroll direction, typed text, clicked position and pressed button) match the gold ones. Previous works take CLICK action as correct if the predicted click point fall within a 14% screen distance from the gold gestures, which is very inaccurate as shown in Figure 3. Therefore, as our datasets contains the bounding boxes of the elements, we define CLICK actions to be correct if the predicted click point or the center of the predicted bounding box falls within the ground truth bounding box (Li et al., 2022; Zhang et al., 2023). For SCROLL actions, we compare whether the predicted direction matches the ground truth. For TYPE actions, if the Average Normalized Levenshtein Similarity (ANLS) between the predicted text and the ground truth is lower than 0.5, we consider it correct. For PRESS actions, we compare the predicted button with the ground truth and consider it as correct if the two are exactly the same. For DONE actions, we consider the prediction correct as long as the action type is accurately predicted.

D Further Analysis

D.1 Choice of Model Structure

We have explored the influence of different architectures, together with the training settings in our preliminary experiments to support both Chinese and English screen understanding. As most MLLMs have native support for English, we put the emphasis on Chinese performance.

Language backbone We select three LLMs, including InternLM1.0 used by TextHawk, InternLM2.5 which is superior to InternLM1.0, and the most recent Qwen2 as the candidates. As shown in Table 9, Qwen2- and InternLM2.5-based agents show better grounding performance, with Qwen2-based agent excelling in sequential navigation tasks. Thus, UI-Hawk is built with Qwen2 backbone.

Unfreezing ViT during pre-training We follow TextHawk (Yu et al., 2024) to use SigLip-SO as the visual encoder. Although TextHawk froze the ViT during the entire training process, several works have illustrated that train ViT is beneficial for

Model	FT?	History	Overall	ClickAcc.
Qwen2-VL	✓	T	58.6	30.8
UI-Hawk	✓	T	73.9	69.3
OdyAgent	✓	V	70.8	43.8
TextHawk	✓	V	71.6	65.8
UI-Hawk	✓	V	79.4	76.3

Table 11: **Comparison between the textual (T) and visual (V) history across various agents fine-tuned on GUI-Odyssey+ dataset.**

obtaining advanced grounding capabilities (Chen et al., 2024c). Therefore, we conduct an ablation study. Table 10 validates that unfreezing ViT during pre-training leads to better grounding ability, and further improves the navigation performance.

D.2 Impact of Screen Streams across MLLMs

To further demonstrate the effectiveness of applying screen streams into history, we have added two LoRA fine-tuned baseline models, Qwen2-VL and TextHawk. As shown in Table 11, under textual history settings, UI-Hawk surpasses Qwen2-VL (they share the same Qwen2 language backbone), validating our architecture advantages. Moreover, even T-history based UI-Hawk is better than V-history based TextHawk, showing the significance of UI-related training process. At the result, V-history based UI-Hawk performs the best, validating the effectiveness of both model and training strategies.

Figure 7: **Qualitative examples of UI-Hawk in FunUI benchmark.** In most cases, UI-Hawk can perform the tasks well. While in some cases where the screen is obstructed or the question contains implicit app knowledge, UI-Hawk's answer would be inaccurate. As shown in the Chinese example of screen question answering task, UI-Hawk fails to identify the most recently added image but conduct OCR to answer the question.

Shared System Prompt

Your name is GUI-Expert, a user interface interaction assistant specifically designed for the Android operating system.
- As a virtual assistant, you can interact with users through the operating system's interface, assist them in resolving requests, and provide descriptions of the content displayed in screenshots.

Use Guidelines:
1. You are provided a screenshot of the current smartphone, along with a textual representation of the current screen.
2. The textual representation is called "Screen Annotation", which is composed of a series of detected UI elements.
3. Each UI element has a class, which is expressed in capital letter. The class is sometimes followed by a description, and then 4 numbers between 0 and 999 represent the bounding box of each element.

Your task is to respond to user requests by reviewing the screenshots of the mobile app interface.

UI Grounding Prompt

Given the screenshot and the screen annotation, I need you to generate referring question-answer pairs: Given a description of an element, provide the corresponding bounding box.

Based the provided referring question-answer pairs, you should convert the questions and answers: While maintaining the original question-answer relationship, place the description of the element into the question and respond with the element's bounding box in the answer. Your output must strictly adhere to the JSON format.

UI Referring Prompt

Given the screenshot and the screen annotation, I need you to generate referring question-answer pairs: Given the bounding box of an element, describe the corresponding element.

Requirements:
1. Question-answer pairs related to ICON: Users may ask questions about icons. Based on elements classified as ICON in the screen annotation, generate potential questions and corresponding answer pairs.
2. Question-answer pairs related to TEXT: The app interface contains a large amount of text. Based on elements classified as TEXT and containing Chinese characters in the screen annotation, generate potential user questions and provide the corresponding text from the screenshot.
3. Question-answer pairs related to WIDGET: The app interface consists of multiple basic elements that form various interactive controls. Users may ask questions about the meaning or functionality of these controls. Based on the higher-level elements identified as WIDGET in the screen annotation, generate potential question-answer pairs.

Response Format
{ "icon": [{"q": "...", "a": "..."}, ...], "text": [{"q": "...", "a": "..."}, ...], "widget": [{"q": "...", "a": "..."}, ...] }

Figure 8: **Data collection prompt for UI grounding and UI referring tasks.** Note that we use the Chinese version of above prompts to generate Chinese data.

Shared System Prompt

Your name is GUI-Expert, a user interface interaction assistant specifically designed for the Android operating system.
- As a virtual assistant, you can interact with users through the operating system's interface, assist them in resolving requests, and provide descriptions of the content displayed in screenshots.

Use Guidelines:
1. You are provided a screenshot of the current smartphone, along with a textual representation of the current screen.
2. The textual representation is called "Screen Annotation", which is composed of a series of detected UI elements.
3. Each UI element has a class, which is expressed in capital letter. The class is sometimes followed by a description, and then 4 numbers between 0 and 999 represent the bounding box of each element.

Your task is to respond to user requests by reviewing the screenshots of the mobile app interface.

Screen QA Prompt

Given the screenshot and the screen annotation, I need you to generate question-answer pairs about the screen contents. You should consider this task from the following aspects: {$screen_qa_types_and_examples}.
Please generate 10 potential questions and provide corresponding answers.

Response Format: [{"q": "...", "a": "..."}, {"q": "...", "a": "..."}, {"q": "...", "a": "..."}, ...]

Screen Summarization Prompt

Given the screenshot and the screen annotation, I need you to summarize the screen contents.
You should carefully observe the screenshot and summarize the contents. Ensure that your description is clear and concise. Answer within threee sentences. The screen summarization should include all important information on the screen and also focus on the screen layout, describing the content in a top-to-bottom, left-to-right order. Note that:
1. For apps with specific names, directly use the app name instead of referring to it generically as "the app."
2. Do not include inherent phone information, such as battery level, network signal, time, or on-screen keyboard.

Figure 9: **Data collection prompt for screen question-answering (QA) and screen summarization tasks.** Note that we use the Chinese version of above prompts to generate Chinese data.