Article

Not peer-reviewed version

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

Jiwen Zhang , Yaqi Yu , Minghui Liao * , Wentao Li , Jihao Wu , Zhongyu Wei *

Posted Date: 30 August 2024

doi: 10.20944/preprints202408.2137.v1

Keywords: GUI Agents; Screen Stream Understanding

 doi:10.20944/preprints202408.2137.v1

Article

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

Jiwen Zhang [1,2,†], Yaqi Yu [2,†], Minghui Liao [2,*], Wentao Li [2], Jihao Wu [2] and Zhongyu Wei [1,*]

[1] Fudan University; jiwenzhang21@m.fudan.edu.cn
[2] Huawei Inc.; yuyaqi5@huawei.com
* Correspondence: liaominghui1@huawei.com (M.L.); zywei@fudan.edu.cn (Z.W.)
† These authors contributed equally to this work.

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 visual GUI agent specially designed to processing screen streams encountered during GUI navigation. UI-Hawk incorporates a history-aware visual encoder and an efficient resampler to handle the screen sequences. To acquire a better understanding of screen streams, we define four fundamental tasks—*UI grounding*, *UI referring*, *screen question answering*, and *screen summarization*. We develop an automated data curation method to generate the corresponding training data for UI-Hawk. Along with the efforts above, we have also 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 not only beneficial but also essential for GUI navigation.

Keywords: GUI Agents; screen stream understanding

1. Introduction

Smartphones have become integral to daily life, elevating the importance of autonomously operating graphical user interfaces (GUI). The task of instruction following on GUI, formalized as GUI navigation, offers substantial potential for automating complex tasks, reducing human workload, and improving 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 [1] as the zero-shot task executor [2–4] or directly tuning MLLMs on the downstream GUI tasks [5,6]. These agents put the basis of their decision-making primarily on current visual observations. Although action history is included to substitute the global context [5], plain-text based history such as "`click [x1, y1, x2, y2], then scroll up`" struggles to capture the nuanced details of clicked UI element [7], thereby hindering the progress. The rich semantics embedded within the stream of screens during navigation are necessary for GUI agents to accurately control the 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. The stream of screens demonstrating that it has searched for "hiking trial" and opened a related article supports the agent to mark the task as "done".

 doi:10.20944/preprints202408.2137.v1

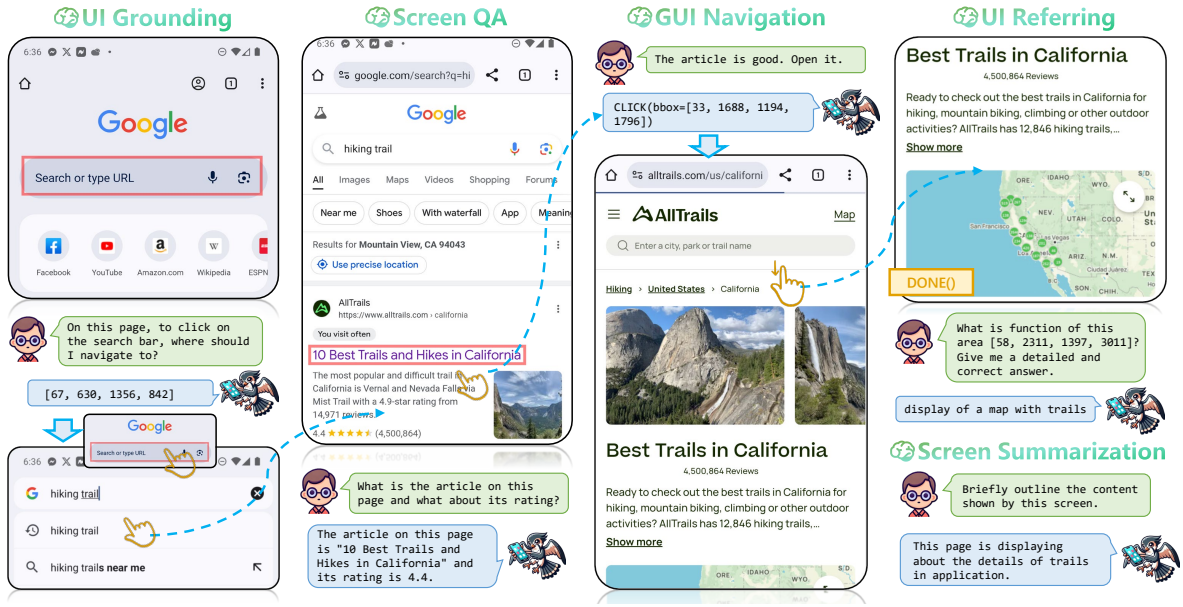


Figure 1. Example of a GUI navigation episode together with the screen understanding tasks supported by UI-Hawk. The user instruction is *"I want to use Chrome to discover a new hiking trail."* The bounding boxes predicted by UI-Hawk are represented by red rectangles. The navigation actions are denoted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

The development of screen stream understanding encounters two major challenges: (1) Efficient representation of screen sequences, especially for MLLMs with limited context window [8,9] is challenging. (2) As illustrated in Figure 1, the instructions associated with 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. To harness the screen sequences, UI-Hawk incorporates a history-aware visual encoder, which explicitly models the temporal dependencies of images. Moreover, we utilize an efficient resampler to compress the visual tokens, enabling UI-Hawk to handle multiple steps of history screens. This meticulous architecture design mitigates the information gap from plain-text based action history, empowering UI-Hawk to effectively perceive the fine-grained details involved in the entire navigation process. To train UI-Hawk, we decompose screen stream understanding into several fundamental tasks, including *UI grounding*, *UI referring*, *screen question answering* and *screen summarization*. Given the scarcity of large-scale UI data, we devise an automated data curation method to extract the task-relevant data from diverse screenshots of Chinese and English mobile devices. Specifically, the method is facilitated by a small yet effective element detection model trained by us. Based on these detections, we employ GPT-4V [1] as the labeller to generate question-answer pairs and descriptions, facilitating both pre-training and fine-tuning of models.

Considering the significance of these fundamental capabilities [10,11], we introduce ***FunUI***, a comprehensive benchmark to quantitatively evaluate the fundamental 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 downstream GUI navigation tasks demonstrates that UI-Hawk establishes a new standard for screen understanding. Our further ablation experiments demonstrate 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 action prediction accuracy by 7.7% and 6.7%, respectively.

 doi:10.20944/preprints202408.2137.v1

Our main contributions are summarized as follows:

- We introduce a unified GUI agent UI-Hawk, combining a history-aware visual encoder and an efficient resampler with LLM to effectively process stream of screens.
- We meticulously identify four fundamental tasks for screen stream understanding, and validates the effectiveness of proposed tasks towards episodic GUI navigation.
- We rigorously construct a bilingual and comprehensive screen understanding benchmark *FunUI*, encompassing 32k samples with over 120 types of UI elements.
- Experimental results demonstrate that possessing the screen stream understanding capability is the key to enhancing the performance of GUI navigation.

2. Related Works

Automatic execution of user instructions on mobile devices has been a trend. Early works [12–14] concentrated on synthetic web or mobile screens. Later, datasets are collected on real webs and apps [15–18] and further scaled-up to facilitate the training [19,20]. Recent progress in this area are dominated by proprietary MLLM-based agents [2], relying on visual prompting [21,22], complex context modeling [4,7] or self-refine [23] 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 [21,22], even with the view hierarchy or other annotations as additional inputs [3,24]. To deal with this problem, this work constructs a universal GUI agent UI-Hawk by customizing MLLMs on multiple fundamental tasks aimed for better screen understanding.

Since screen understanding is significant [25–27], several works utilize MLLMs as the foundation and fine-tunes the model on partial aspects of screen-related tasks [6,10,28,29]. However, these models only takes text-based action 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.

3. Methodology

To enable screen stream understanding, UI-Hawk introduces two key enhancements: (1) an optimized model architecture for efficient screen perception, detailed in Section 3.1, and (2) training data encompassing a wide range of screen understanding tasks, as outlined in Section 3.2.

3.1. 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 key 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 foundational MLLMs, TextHawk [30] stands out as the closest to fulfilling these needs. Building on TextHawk, UI-Hawk integrates a lightweight visual encoder, specifically the ViT from SigLIP-SO [31], along with a resampler and a large language model enhanced with LoRA modules and a detection head. UI-Hawk excels in perceiving fine-grained details across various screen sizes through a shape-adaptive cropping strategy that processes images of any resolution. Additionally, it features a carefully designed resampler to efficiently compress visual tokens and a detection head for direct modeling of bounding boxes. The model architecture is depicted in Figure 2.

 doi:10.20944/preprints202408.2137.v1

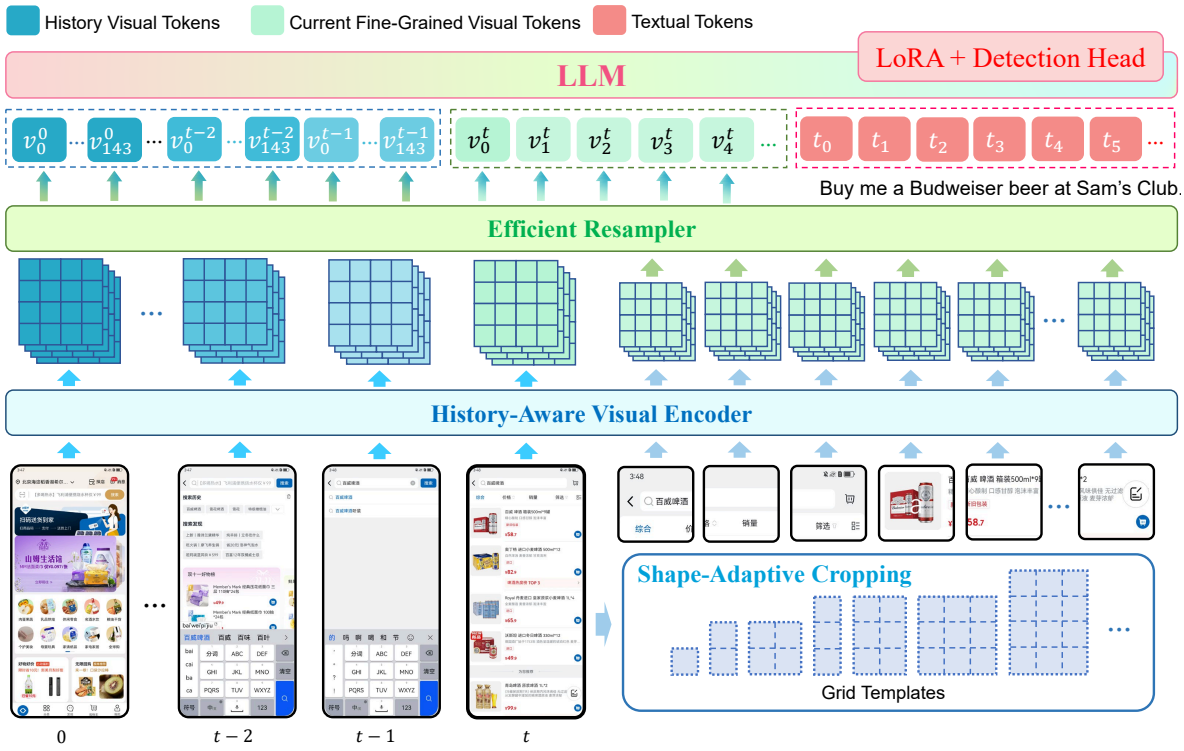


Figure 2. Model architecture of UI-Hawk. The text tokenizer, layer normalization, and skip connections are omitted for simplicity. During the pre-training stage, the visual encoder is trained together with the LLM to obtain the fine-grained perception capabilities. During the fine-tuning stage, the visual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

Importantly, UI-Hawk places a strong emphasis on modeling visual history. Historical images often contain valuable details pertinent to ongoing tasks, while most GUI agents rely solely on text-based history, like chain-of-actions [5] or chain-of-action-thoughts [7]. UI-Hawk addresses this gap by incorporating images of historical screens as model inputs. Notably, we apply a scalable positional encoding and add textual indicators (e.g., "`History Screenshot x`") for each historical screen to explicitly represent the screen streams. Previous models faced challenges with efficiently modeling visual history, as encoding each page required thousands of visual tokens. To overcome this, UI-Hawk introduces two key improvements to reduce context length. First, history images are downscaled to a quarter of their original size. Second, UI-Hawk employs a much larger visual token compression ratio of 16, which is four times larger than in previous models [8,32]. As a result, a typical historical screenshot is divided into 8 sub-images along with a global thumbnail, using only 144 visual tokens.

3.2. Model Training

We develop UI-Hawk through continual pre-training on TextHawk [30]. During pre-training, we unfreeze the ViT by LoRA [33] and train model on the screen annotation dataset we collected in Section 4.1.0.2 together with other document-related datasets used by TextHawk (i.e. DocGemini) for one epoch. The pre-training improves both the OCR and the screen infographics understanding ability of models, 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, in which UI-Hawk is firstly trained to comprehend the screens, and then transformed into a GUI agent by training on sequential navigation tasks. Specifically, in stage one, UI-Hawk includes a broad range of screen-related tasks as shown in Table 1 to obtain the basic screen understanding capabilities. In stage two, we utilize sequential tasks for GUI navigation as the training

 doi:10.20944/preprints202408.2137.v1

data, enabling UI-Hawk to learn to deal with screen streams based on user instructions and execution history. These tasks are bilingual and are detailed in Section 4. The entire fine-tuning takes 3 days on 32 Tesla V100. We kindly refer readers who are interested in our training details to Appendix B.

Table 1. Summary of the fune-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 instruction-episode pairs.

Task	# Samples		Data Source	
	ZH	EN	ZH	EN
UI Grounding	580k	16k	Ours	[25]
UI Referring	600k	109k	Ours	[34]
Screen QA	1200k	288k	Ours	[35]
Screen Sum.	50k	78k	Ours	[36]
GUI Navigation	55k	87k	Ours	[18]

4. Dataset and Task Formulation

In this section, we demonstrate the process of generating tasks and dataset for model training and evaluation. In Section 4.1, we detail the screen data collection. While in Section 4.2, we explain how we formulate the fundamental screen-related tasks. In Section 4.3, we demonstrate the sequential navigation tasks used to train model as a GUI agent.

4.1. Data Collection

Mobile Screens

To obtain screen understanding ability, it is essential to assemble a diverse range of mobile screens. For Chinese screens, following [37], we use an automated traversal tool to crawl screens from more then 420 apps, sorted by download counts in the app market. Screens collected by automated traversal often have a high degree of repetition [38]. We employ a pixel-wise filtering algorithm combined with screen structure to eliminate duplicate images (see Appendix A for more details). 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 RICO dataset [13], which serves as the image foundation for several screen-related tasks [25,34–36]. We keep the original data split from each task to further construct the training data and evaluation benchmark (detailed in Section 5). 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 [39,40]. 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 [41] based UI detection model. Our model is responsible for detecting basic UI elements covering ICON (133 types, extended from [42]), TEXT, IMAGE, INPUT_FIELD and KEYBOARD. Similar to previous works [11,40], we group basic elements into associated items, namely high-level widgets. Since screen annotations enable textual representation of screens [39], we refer the task of generating such annotations solely based on the input image as *screen annotation* task, which serves as a pre-training task to enable the understanding of screen infographics.

4.2. Fundamental Tasks

Previous work [7] have found that the process of GUI screen streams can be divided into several minor steps, including describing the screen, referring to the target UI elements and generating the

 doi:10.20944/preprints202408.2137.v1

corresponding action coordinates. Hence, we discover four fundamental capacities that are crucial for screen stream understanding, including:

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 [34]. 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

UI Grounding task measures the regional localization capability. The model is required to accurately locate UI elements based on the instructions. For English screens, Referring Expression [25] 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. We categorize the questions into nine major types, detailed in Appendix A.2. For English screens, we employ the Google ScreenQA datset [35]. For Chinese screens, we prompt GPT-4V 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 [36] dataset is applied to maintain a fair comparison with previous SOTA models [40]. For Chinese, we employ GPT-4V to concisely describe the screen within three sentences. Around 50k screens are annotated by GPT-4V.

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 correct all samples 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.

4.3. GUI Navigation Tasks

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

GUI-Odyssey+

GUI-Odyssey [18] 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

 doi:10.20944/preprints202408.2137.v1

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+.

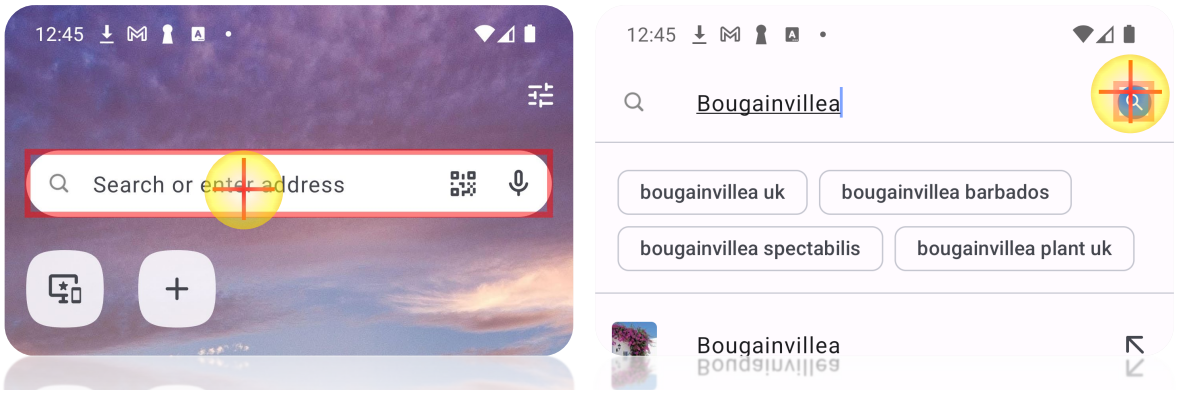


Figure 3. Examples from GUI-Odyssey dataset. *Left*: The region of clicked element (red bounding box) is larger than area with which click operations are considered correct (shadowed orange circle). *Right*: The region of clicked element is smaller than the area of correct click operations.

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.

5. *FunUI* Benchmark

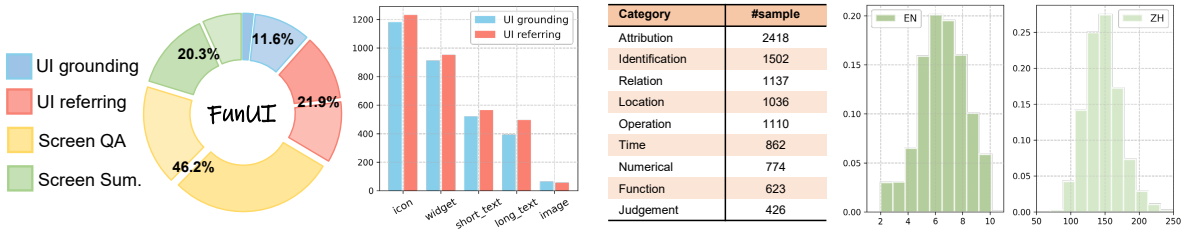
Screen understanding is the basis of screen stream understanding. Currently, the evaluation of the screen understanding capabilities of MLLMs remains a open question. Main challenges comes from ambiguous definition of what is the fundamental aspects of the screen understanding, and the lack of an exhaustive dataset that could cover both the various types of screen elements and the screen semantics.

To remedy the blank in this area, we introduce *FunUI*, a bilingual evaluation benchmark encompassing four fundamental screen understanding tasks. Concretely, *FunUI* distinguishes with previous benchmarks [10,28,35] on the following aspects: **(a) 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 screen understandings. **(b) Comprehensive**: *FunUI* includes different evaluation dimensions of screen understanding, including *UI grounding* and *UI referring* tasks to access the regional location and identification abilities of models, together with *screen question answering* and *screen summarization* tasks that require more integrated analysis of screen contents. **(c) 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, attributions, arithmetics and so on. These questions present a greater challenge for models compared to the OCR-related tasks commonly used in benchmarks like GUICourse.

To ensure reliable evaluation under real scenarios, *FunUI* is carefully crafted: (1) For English screens, we meticulously select the union of test images from previous dataset [25,34–36] so that models trained for English screens could be consistently compared with previous SOTA methods, i.e.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 August 2024 doi:10.20944/preprints202408.2137.v1

Ferret-UI [40]. (2) For Chinese screens, we recruit 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. The basic statistics of *FunUI* are illustrated in Figure 4.



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

(a) Task distribution (b) Types of UI elements (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 question answering pairs. Note that these categories are not mutually excluded. (d) The annotated answer lengths of screen summarization task.

6. Experiments

In this section, we first introduce the general set-up of the experiments, followed by a comprehensive assessment of screen understanding capabilities on representative MLLMs. Then, we conduct ablation studies to verify the effectiveness of screen stream modeling towards mobile GUI navigation.

6.1. Experimental Setup

6.1.1. Baselines

We adopt different types of MLLMs as the baselines: (1) the proprietary GPT-4V [1], (2) the open-source models like Qwen-VL-Chat [8] and InternVL2-8B [43], (3) models specifically designed for GUI tasks, including MLLMs for screen understanding like Spotlight [28], Ferret-UI [40] and SeeClick [10], and MLLMs targeted for GUI navigation like CogAgent [6] and OdysseyAgent [18]. Since currently all UI-specific models are trained under English contexts, we only compare UI-Hawk with two generalist MLLMs, GPT-4V and Qwen-VL-Chat, on Chinese screens.

6.1.2. Evaluation Metrics

For fundamental tasks, we use the accuracy computed at IoU=0.5 for UI grounding, SQuAD-F1 score for screen question answering following [35], 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 [5,18,19]. An action is considered correct if both the action type and the action details (i.e. scroll direction, typed text, clicked position and pressed button) match the gold ones. As we have mentioned in Section 4.3, a correct click action means the model predicted location (either a point or the center of the bounding box) falls in the ground truth bounding box [44,45]. The evaluation details can be found in Appendix C.

6.2. 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

 doi:10.20944/preprints202408.2137.v1

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 screen-related information in general data, proving the significance of constructing such training samples to acquire the domain-specific knowledge for GUI tasks. Overall, Table 3 suggests that UI-Hawk is a bilingual model with advanced screen understanding capabilities.

GUI Navigation

Following the evaluation methods use by CogAgent [6], SeeClick [10] and GUI-Odyssey [18], we assess the performance of UI-Hawk under in-domain settings. Results are summarized 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. Further ablation studies confirm that, the improvements in UI-Hawk are largely attributed to its advanced screen stream understanding ability.

Table 2. Sequential navigation performance on GUI-Odyssey+ dataset. We report the averaged 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 operations, which directly reflects the grounding ability of models. "FT?" means whether the model is fine-tuned on the train split of GUI-Odyssey+ dataset. * Due to the budget limit, we randomly sampled 500 instances for each category of navigation tasks for evaluation.

Model	FT?	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
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**

 doi:10.20944/preprints202408.2137.v1

Table 3. Performance of UI understanding on *FunUI* benchmark. *GRD*: grounding, *REF*: referring, *SQA*: screen question answering, *SUM*: screen summarization. "FT?" means whether the model is trained on UI-related tasks. *Due to the budget limit, we randomly sampled 500 samples for each fundamental task for evaluation. †Performance of close-source models from the original paper.

(a) Results on English screens.

Model	FT?	GRD	REF	SQA	SUM
		Acc	CIDEr	SQuAD-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

(b) Results on Chinese screens.

Model	FT?	GRD	REF	SQA	SUM
		Acc	CIDEr	SQuAD-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**

6.3. Ablation Studies

The Impact of Fundamental Screen Understanding

To investigate the influence of basic screen understanding towards the final navigation performance, we randomly sample 64k data for each task to conduct the stage one fine-tuning. As shown in Table 4, each fundamental task contributes to the improvement of navigation performance, within which UI grounding task influences the prediction of click operations most. Model trained on the averagely mixed data (line 8) has outstanding performance on GUI-Odyssey+ but 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.

Table 4. Ablation study on the effect of fundamental UI tasks and different history representations. Following [18], the default history length is set as 4 across all experiments, no matter represented by texts or image sequences. For English grounding tasks, we repeat the original 16k training samples to 32k for a fair comparison.

	Grounding		Referring		ScreenQA		Screen Sum.		Visual	GUI-Odyssey+		GUI-Zouwu	
	EN	CN	EN	CN	EN	CN	EN	CN	History	Overall	ClickAcc	Overall	ClickAcc
(1)									×	71.7	66.9	41.2	49.3
(2)									✓	75.7	71.9	44.8	56.5
(3)	32	32							✓	77.8	74.5	46.1	59.2
(4)			32	32					✓	77.7	73.9	46.0	58.4
(5)					32	32			✓	77.6	73.6	46.5	58.4
(6)							32	32	✓	77.3	73.1	45.6	57.7
(7)	32	32	32	32	32	32	32	32	×	72.7	68.1	43.3	55.6
(8)	32	32	32	32	32	32	32	32	✓	78.3	75.1	45.9	58.4
(9)	*Full Data*								✓	**79.4**	**76.3**	**47.9**	**61.4**

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 August 2024 doi:10.20944/preprints202408.2137.v1

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 4 indicate that visual history information has an essential impact on GUI navigation. And 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.

7. Conclusion

In this paper, we introduced UI-Hawk, the first 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 under Chinese and English scenarios, 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.

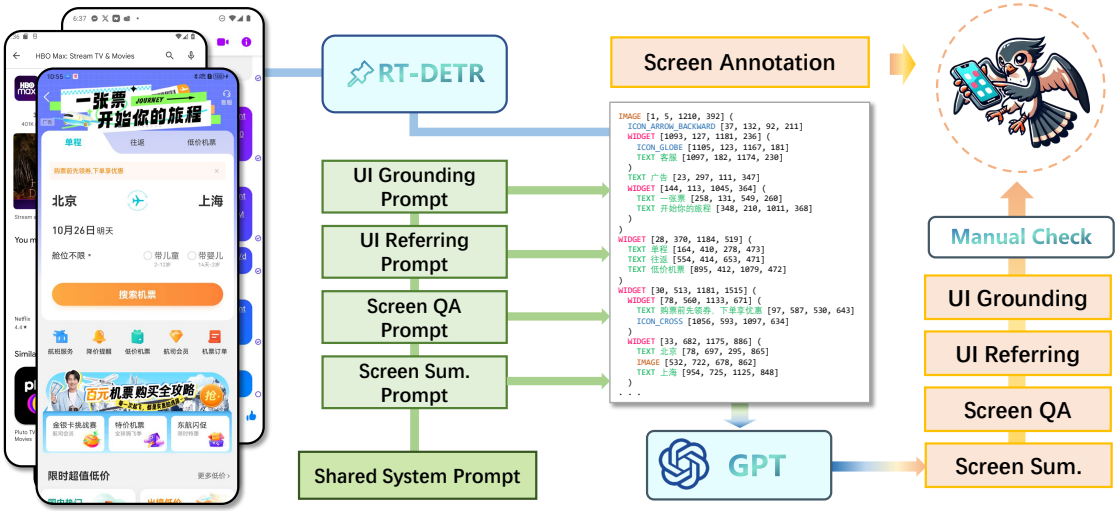


Figure 5. Overall data collection pipeline.

Appendix 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.

Appendix A.1. Screen Annotation

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

 doi:10.20944/preprints202408.2137.v1

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 [11,42] have severe issues including inaccurate bounding boxes and missed detections. As shown in Figure A1(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, achieving an average recall of 95% for various UI elements. An illustration example is shown in Figure A1(b). Following [39], we construct the textual representation of a screen by considering the containment relationships between the elements. See Figure A1(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.

Figure A1. Comparison of different detection models. (a) The detection outcomes from IconNet [42]. (b) The detection outcomes from RT-DETR model trained by us. (c) The corresponding screen annotation example.

Appendix 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. Figure 5 summarizes the data construction pipeline. The prompt we used are summarized in Figure A2. Specifically, for Screen QA task, 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 "What is the rating of the xxx?" and "Who is the author of the book yyy?".

 doi:10.20944/preprints202408.2137.v1

- 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 located at 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 on this page?"
- Judgement Questions: These questions involve making yes/no or true/false determinations. For example, “Is it possible to upgrade to VIP on this page? "

Regarding the screen summarization, since GPT-4V tends to generate overly long and detailed explanations, we instruct it in the prompt to summarize the screen content within three sentences. 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 (see Appendix C)

Appendix 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.

Table A1. Details of the evaluation for Chinese screen summarization. We require the GPT-4O to evaluate from four perspectives: content(0-10), structure(0-10), fluency(0-10) and authenticity(0-10). The final GPT-Score is 10× the average score.

	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

Appendix 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 from our collected screens. As we employ SigLIP-SO [31], each sub-image is sized at 224×224. The language backbone is Qwen2-7B [9]. We employ an effective batch size approaching 1500 and we train UI-Hawk for about

 doi:10.20944/preprints202408.2137.v1

one epoch on a data mixture of screen annotation data and other document-related data used in [30]. 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 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.

Table A2. Impact of IoU thresholds on *grounding* accuracy. Obviously, a low IoU threshold exaggerates the model's performance, especially for those with inaccurate predictions.

(en)	GPT4-V	SeeClick	UI-Hawk
IoU=0.1	27.4	62.1	85.5
IoU=0.5	2.3	29.6	63.9
$\|\Delta\|$	25.13	32.57	21.59

Appendix C. Evaluation Details

Appendix C.1. Fundamental Tasks

UI Grounding

Previous work often employ a relatively low IoU threshold, such as IoU = 0.1 [39], when evaluating grounding tasks. However, in the field of object detection, setting the IoU threshold at 0.5 is more widely used [46]. This stricter standard prevents the exaggeration of the performance (see Table A2).

UI Referring

For referring, we apply the CIDEr metric [47] as [34] has done, as this task is relatively straightforward and the responses are typically one sentence long.

Screen QA

Following [35], 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 [36]. 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 A1.

 doi:10.20944/preprints202408.2137.v1

Appendix C.2. GUI Navigation

Action Space

Following [7], we unify the action space into 5 kinds of actions: CLICK, SCROLL, TYPE, PRESS and DONE, with functionality as:

- **`CLICK(bbox=[x1, y1, x2, y2])`**: click (including long press) the on-screen element within the box.
- **`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 going to the home screen, “press back" means going to the previous screen, “press recent" means going to the previous app.
- **`DONE(status="complete|impossible")`**: stop and judge whether the task has been completed.

Metrics

We utilize the action matching score [5,19] 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. 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.

 doi:10.20944/preprints202408.2137.v1

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": "..."}, ...] }

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 A2. Data collection prompt. Note that we use the Chinese version of above prompts to generate Chinese data.

References

1. OpenAI. 2023. GPT-4V.

 doi:10.20944/preprints202408.2137.v1

2. Yang, Z.; Liu, J.; Han, Y.; Chen, X.; Huang, Z.; Fu, B.; Yu, G. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771* **2023**.
3. Wang, J.; Xu, H.; Ye, J.; Yan, M.; Shen, W.; Zhang, J.; Huang, F.; Sang, J. Mobile-Agent: Autonomous Multi-Modal Mobile Device Agent with Visual Perception, 2024, [arXiv:cs.CL/2401.16158].
4. Zhang, C.; Li, L.; He, S.; Zhang, X.; Qiao, B.; Qin, S.; Ma, M.; Kang, Y.; Lin, Q.; Rajmohan, S.; others. Ufo: A ui-focused agent for windows os interaction. *arXiv preprint arXiv:2402.07939* **2024**.
5. Zhan, Z.; Zhang, A. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436* **2023**.
6. Hong, W.; Wang, W.; Lv, Q.; Xu, J.; Yu, W.; Ji, J.; Wang, Y.; Wang, Z.; Dong, Y.; Ding, M.; others. Cogagent: A visual language model for gui agents. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 14281–14290.
7. Zhang, J.; Wu, J.; Teng, Y.; Liao, M.; Xu, N.; Xiao, X.; Wei, Z.; Tang, D. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713* **2024**.
8. Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; Zhou, J. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966* **2023**.
9. Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; Dong, G.; Wei, H.; Lin, H.; Tang, J.; Wang, J.; Yang, J.; Tu, J.; Zhang, J.; Ma, J.; Yang, J.; Xu, J.; Zhou, J.; Bai, J.; He, J.; Lin, J.; Dang, K.; Lu, K.; Chen, K.; Yang, K.; Li, M.; Xue, M.; Ni, N.; Zhang, P.; Wang, P.; Peng, R.; Men, R.; Gao, R.; Lin, R.; Wang, S.; Bai, S.; Tan, S.; Zhu, T.; Li, T.; Liu, T.; Ge, W.; Deng, X.; Zhou, X.; Ren, X.; Zhang, X.; Wei, X.; Ren, X.; Liu, X.; Fan, Y.; Yao, Y.; Zhang, Y.; Wan, Y.; Chu, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; Guo, Z.; Fan, Z. Qwen2 Technical Report, 2024, [arXiv:cs.CL/2407.10671].
10. Cheng, K.; Sun, Q.; Chu, Y.; Xu, F.; Li, Y.; Zhang, J.; Wu, Z. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935* **2024**.
11. Fan, Y.; Ding, L.; Kuo, C.C.; Jiang, S.; Zhao, Y.; Guan, X.; Yang, J.; Zhang, Y.; Wang, X.E. Read Anywhere Pointed: Layout-aware GUI Screen Reading with Tree-of-Lens Grounding, 2024, [arXiv:cs.CL/2406.19263].
12. Shi, T.; Karpathy, A.; Fan, L.; Hernandez, J.; Liang, P. World of bits: An open-domain platform for web-based agents. International Conference on Machine Learning. PMLR, 2017, pp. 3135–3144.
13. Deka, B.; Huang, Z.; Franzen, C.; Hibschman, J.; Afergan, D.; Li, Y.; Nichols, J.; Kumar, R. Rico: A mobile app dataset for building data-driven design applications. Proceedings of the 30th annual ACM symposium on user interface software and technology, 2017, pp. 845–854.
14. Liu, E.Z.; Guu, K.; Pasupat, P.; Shi, T.; Liang, P. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. International Conference on Learning Representations, 2018.
15. Burns, A.; Arsan, D.; Agrawal, S.; Kumar, R.; Saenko, K.; Plummer, B.A. Mobile app tasks with iterative feedback (motif): Addressing task feasibility in interactive visual environments. *arXiv preprint arXiv:2104.08560* **2021**.
16. Sun, L.; Chen, X.; Chen, L.; Dai, T.; Zhu, Z.; Yu, K. META-GUI: Towards Multi-modal Conversational Agents on Mobile GUI. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, 2022, pp. 6699–6712.
17. Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; Su, Y. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems* **2024**, *36*.
18. Lu, Q.; Shao, W.; Liu, Z.; Meng, F.; Li, B.; Chen, B.; Huang, S.; Zhang, K.; Qiao, Y.; Luo, P. GUI Odyssey: A Comprehensive Dataset for Cross-App GUI Navigation on Mobile Devices. *arXiv preprint arXiv:2406.08451* **2024**.
19. Rawles, C.; Li, A.; Rodriguez, D.; Riva, O.; Lillicrap, T. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems* **2024**, *36*.
20. Chen, D.; Huang, Y.; Wu, S.; Tang, J.; Chen, L.; Bai, Y.; He, Z.; Wang, C.; Zhou, H.; Li, Y.; others. GUI-WORLD: A Dataset for GUI-oriented Multimodal LLM-based Agents. *arXiv preprint arXiv:2406.10819* **2024**.
21. Yan, A.; Yang, Z.; Zhu, W.; Lin, K.; Li, L.; Wang, J.; Yang, J.; Zhong, Y.; McAuley, J.; Gao, J.; others. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562* **2023**.
22. Zheng, B.; Gou, B.; Kil, J.; Sun, H.; Su, Y. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614* **2024**.

 doi:10.20944/preprints202408.2137.v1

23. Kim, G.; Baldi, P.; McAleer, S. Language models can solve computer tasks. *Advances in Neural Information Processing Systems* **2024**, *36*.
24. Wen, H.; Li, Y.; Liu, G.; Zhao, S.; Yu, T.; Li, T.J.J.; Jiang, S.; Liu, Y.; Zhang, Y.; Liu, Y. Empowering llm to use smartphone for intelligent task automation. *arXiv preprint arXiv:2308.15272* **2023**.
25. Bai, C.; Zang, X.; Xu, Y.; Sunkara, S.; Rastogi, A.; Chen, J.; others. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731* **2021**.
26. Zhang, X.; De Greef, L.; Swearngin, A.; White, S.; Murray, K.; Yu, L.; Shan, Q.; Nichols, J.; Wu, J.; Fleizach, C.; others. Screen recognition: Creating accessibility metadata for mobile applications from pixels. Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 2021, pp. 1–15.
27. Venkatesh, S.G.; Talukdar, P.; Narayanan, S. Ugif: Ui grounded instruction following. *arXiv preprint arXiv:2211.07615* **2022**.
28. Li, G.; Li, Y. Spotlight: Mobile ui understanding using vision-language models with a focus. *arXiv preprint arXiv:2209.14927* **2022**.
29. Jiang, Y.; Schoop, E.; Swearngin, A.; Nichols, J. ILuvUI: Instruction-tuned LangUage-Vision modeling of UIs from Machine Conversations. *arXiv preprint arXiv:2310.04869* **2023**.
30. Yu, Y.Q.; Liao, M.; Wu, J.; Liao, Y.; Zheng, X.; Zeng, W. TextHawk: Exploring Efficient Fine-Grained Perception of Multimodal Large Language Models, 2024, [arXiv:cs.CV/2404.09204].
31. Zhai, X.; Mustafa, B.; Kolesnikov, A.; Beyer, L. Sigmoid loss for language image pre-training. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 11975–11986.
32. Ye, J.; Hu, A.; Xu, H.; Ye, Q.; Yan, M.; Xu, G.; Li, C.; Tian, J.; Qian, Q.; Zhang, J.; others. Ureader: Universal ocr-free visually-situated language understanding with multimodal large language model. *arXiv preprint arXiv:2310.05126* **2023**.
33. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* **2021**.
34. Li, Y.; Li, G.; He, L.; Zheng, J.; Li, H.; Guan, Z. Widget Captioning: Generating Natural Language Description for Mobile User Interface Elements. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 5495–5510.
35. Hsiao, Y.C.; Zubach, F.; Wang, M.; others. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199* **2022**.
36. Wang, B.; Li, G.; Zhou, X.; Chen, Z.; Grossman, T.; Li, Y. Screen2words: Automatic mobile UI summarization with multimodal learning. The 34th Annual ACM Symposium on User Interface Software and Technology, 2021, pp. 498–510.
37. Wu, J.; Krosnick, R.; Schoop, E.; Swearngin, A.; Bigham, J.P.; Nichols, J. Never-ending learning of user interfaces. Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2023, pp. 1–13.
38. Feiz, S.; Wu, J.; Zhang, X.; Swearngin, A.; Barik, T.; Nichols, J. Understanding screen relationships from screenshots of smartphone applications. Proceedings of the 27th International Conference on Intelligent User Interfaces, 2022, pp. 447–458.
39. Baechler, G.; Sunkara, S.; Wang, M.; Zubach, F.; Mansoor, H.; Etter, V.; Cărbune, V.; Lin, J.; Chen, J.; Sharma, A. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615* **2024**.
40. You, K.; Zhang, H.; Schoop, E.; Weers, F.; Swearngin, A.; Nichols, J.; Yang, Y.; Gan, Z. Ferret-UI: Grounded Mobile UI Understanding with Multimodal LLMs. *arXiv preprint arXiv:2404.05719* **2024**.
41. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. Detrs beat yolos on real-time object detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 16965–16974.
42. Sunkara, S.; Wang, M.; Liu, L.; Baechler, G.; Hsiao, Y.C.; Sharma, A.; Stout, J.; others. Towards better semantic understanding of mobile interfaces. *arXiv preprint arXiv:2210.02663* **2022**.
43. Chen, Z.; Wang, W.; Tian, H.; Ye, S.; Gao, Z.; Cui, E.; Tong, W.; Hu, K.; Luo, J.; Ma, Z.; others. How Far Are We to GPT-4V? Closing the Gap to Commercial Multimodal Models with Open-Source Suites. *arXiv preprint arXiv:2404.16821* **2024**.

 doi:10.20944/preprints202408.2137.v1

44. Li, L.H.; Zhang, P.; Zhang, H.; Yang, J.; Li, C.; Zhong, Y.; Wang, L.; Yuan, L.; Zhang, L.; Hwang, J.N.; others. Grounded language-image pre-training. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10965–10975.
45. Zhang, Z.; Xie, W.; Zhang, X.; Lu, Y. Reinforced ui instruction grounding: Towards a generic ui task automation api. *arXiv preprint arXiv:2310.04716* **2023**.
46. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision* **2010**, *88*, 303–338.
47. Vedantam, R.; Lawrence Zitnick, C.; Parikh, D. Cider: Consensus-based image description evaluation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 4566–4575.