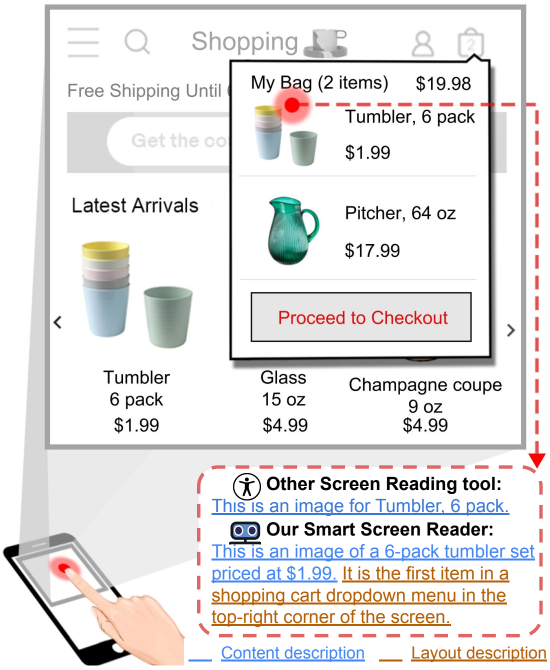
Read Anywhere Pointed: Layout-aware GUI Screen Reading with Tree-of-Lens Grounding

Yue Fan[1], Lei Ding[1], Ching-Chen Kuo[2], Shan Jiang[2], Yang Zhao[2], Xinze Guan[2], Jie Yang[3], Yi Zhang[1], and Xin Eric Wang[1]

[1]University of California, Santa Cruz
[2]eBay Inc.
[3]Cybever

Abstract

Graphical User Interfaces (GUIs) are central to our interaction with digital devices and growing efforts have been made to build models for various GUI understanding tasks. However, these efforts largely overlook an important GUI-referring task: screen reading based on user-indicated points, which we name the Screen Point-and-Read (ScreenPR) task. Currently, this task is predominantly handled by rigid accessible screen reading tools, in great need of new models driven by advancements in Multimodal Large Language Models (MLLMs). In this paper, we propose a Tree-of-Lens (ToL) agent, utilizing a novel ToL grounding mechanism, to address the ScreenPR task. Based on the input point coordinate and the corresponding GUI screenshot, our ToL agent constructs a Hierarchical Layout Tree. Based on the tree, our ToL agent not only comprehends the content of the indicated area but also articulates the layout and spatial relationships between elements. Such layout information is crucial for accurately interpreting information on the screen, distinguishing our ToL agent from other screen reading tools. We also thoroughly evaluate the ToL agent against other baselines on a newly proposed ScreenPR benchmark, which includes GUIs from mobile, web, and operating systems. Last but not least, we test the ToL agent on mobile GUI navigation tasks, demonstrating its utility in identifying incorrect actions along the path of agent execution trajectories. Code and data: https://screen-point-and-read.github.io.



This is an image for Tumbler, 6 pack.

This is an image of a 6-pack tumbler set priced at $1.99. It is the first item in a shopping cart dropdown menu in the top-right corner of the screen.

Figure 1: Our ToL agent describes the region on the screenshot indicated by a user's point. Distinguished from other screen reading tools, our ToL agent can output layout-aware descriptions for points anywhere on the screen.

1 Introduction

Graphical User Interfaces (GUIs), dominate our digital interactions with visually rich screenshots featuring colors, icons, texts, and spatial layouts. Recognizing that screenshots are more accessible and intuitive for depicting visual cues, there are growing efforts to build AI agents to interpret GUIs visually (Shaw et al., 2023; Deng et al., 2023; Cheng et al., 2024; Hong et al., 2023; You et al., 2024). Existing works, however, overlook the point-based screen reading task, where the input is the coordinate of a user's indicated point on the screen with the corresponding screenshot, and the output is a descriptive interpretation of the screen region pointed. This task is critical for accessible technology, providing valuable assistance to users with visual impairments and we refer to such a task as the Screen Point-and-Read (ScreenPR) task.

Aiming to solve the ScreenPR task, we introduce the Tree-of-Lens (ToL) agent. Taking advantage of the generalizability of the advanced Multimodal

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 9503–9522
November 12-16, 2024 ©2024 Association for Computational Linguistics

Large Language Models (MLLMs), the ToL agent is built for GUI screenshots across domains with a point anywhere on the screen as input. The output from the ToL agent is natural language descriptions for both the content of the indicated region and the related information about the screen layout, as shown in Figure 1. Especially, we find that it is critical to articulate the layout between elements in the screenshot for users to gain a comprehensive understanding of the interface and to avoid ambiguity. For example, with only content description, one cannot distinguish the two identical “Tumbler pack" shown in Figure 1.

Our ToL agent employs a ToL grounding mechanism, constructing a Hierarchical Layout Tree for each input screenshot to depict its underlying structure. The nodes in the tree represent regions of varying scales. We build this tree using an object detection model trained on our newly collected Android Screen Hierarchical Layout (ASHL) dataset, which contains 50k bounding boxes of hierarchical screen regions from Android screenshots. This model enables an automatic extraction of local and global regions, forming the Hierarchical Layout Tree. After constructing the tree, we extract the target path based on the region of interest, producing lenses as visual prompts for the MLLM, simulating a human-like, gradually refined focus.

To rigorously evaluate our ToL agent against other baselines, we introduce the Screen Point-and-Read (ScreenPR) benchmark with the ScreenPR task that requires the model to generate descriptions based on user-indicated points. This benchmark consists of 650 screenshots from web, mobile, and operating system GUIs, manually annotated with 1,500 target points and regions. With the ScreenPR benchmark, we show that our ToL agent achieves the best performance compared with the baselines of other general and GUI-specialized MLLMs, with over 15% and 30% improvements respectively in terms of the accuracy of content and layout descriptions compared with vanilla text-prompting GPT-4o.

In addition, we also test our ToL agent with trajectories from a mobile GUI navigation agent. By applying the ToL agent to describe the point of each action taken in the trajectory, we demonstrate its utility in identifying incorrect actions along the execution path. This capability is pivotal for refining the development of mobile agents, as it provides clear, actionable feedback on navigation decisions. This application not only validates the ToL agent's effectiveness but also opens up possibilities for strengthening the performance of mobile agents with sophisticated, layout-aware GUI understanding capabilities.

In conclusion, the contributions of our paper are as follows:

- We develop the Tree-of-Lens (ToL) grounding method and build the ToL agent that generates descriptions that effectively combine both content and layout information of the pointed screen regions.
- We propose the Hierarchical Layout Tree to represent the underlying hierarchical structure of the screenshot. We collect the Android Screen Hierarchical Layout dataset and train a model to build the Hierarchical Layout Tree.
- We introduce the Screen Point-and-Read benchmark, which specifically challenges the model to produce accurate descriptions that include both content and layout details.
- We rigorously test the ToL agent on the Screen Point-and-Read benchmark and we demonstrate its ability to accurately identify incorrect actions in mobile navigation trajectories.

2 Related Works

MLLMs for GUI Understanding Recent advancements in Multimodal Large Language Models (MLLMs) have led to a rising series of highly capable generalist models (Liu et al., 2024b; Chen et al., 2023; Ye et al., 2023; Dai et al., 2023; Bai et al., 2023; Liu et al., 2024a). Spurred by the strong general capability of MLLMs, recent works have been focused on utilizing and improving MLLMs for GUI tasks. Some focus mainly on a specific GUI task such as GUI navigation (Wang et al., 2024; Yang et al., 2023b; Zheng et al., 2024), GUI referring expression comprehension (Cheng et al., 2024), while some aim at a set of GUI grounding and referring tasks simultaneously, such as CogAgent (Hong et al., 2023), which is Multi-task Finetuned on CogVLM (Wang et al., 2023), and Ferret-UI (You et al., 2024), ensuing Ferret (You et al., 2023; Zhang et al., 2024), tailored for GUI domain. In our work, we focus on the Screen Point-and-Read (ScreenPR) task, a referring task in which inputs are point locations on the screen with the corresponding screenshots, and develop a specific

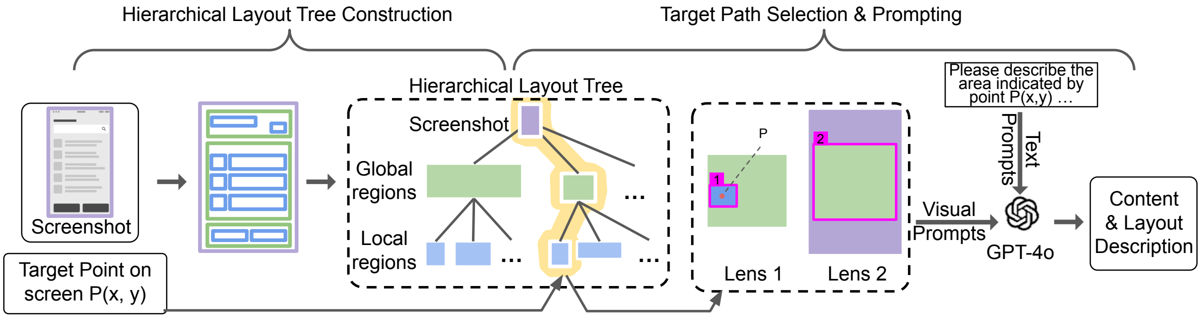

Hierarchical Layout Tree Construction
Target Path Selection & Prompting
Screenshot
Target Point on screen P(x, y)
Hierarchical Layout Tree
Screenshot
Global regions
Local regions
P
1
2
Lens 1
Lens 2
Please describe the area indicated by point P(x,y) ...
Text Prompts
Visual Prompts
GPT-4o
Content & Layout Description

Figure 2: Pipeline of the Tree-of-Lens agent. The Hierarchical Layout Tree is first constructed based on detected global and local regions from the input screenshot. Then, a set of hierarchical lenses with various field widths is generated from the selected target path in the tree and sent as visual prompts to GPT-4o to generate the content and layout descriptions.

benchmark to evaluate the model's performance in such tasks. Also, inspired by big challenges that MLLMs are facing in understanding images with complex layout information (Fan et al., 2024), we propose the Tree-of-Lens (ToL) agent that generates layout-aware description, which distinguishes from all other GUI agents with GUI referring ability.

Chain-of-Thoughts for Vision The Chain of Thoughts (CoT) (Wei et al., 2022) methodology in natural language processing (NLP) has spurred various inspiring ideas in this direction. (Creswell et al., 2022) applied it to interpretable logical reasoning, (Wang et al., 2022) focused on enhancing the "exploring-selecting" mechanism in reasoning path, and (Yao et al., 2024; Besta et al., 2024) brought in more sophisticated patterns for reasoning. To embrace CoT in the vision area, Xi et al. (2023) uses Chain-of-Look prompting for surgical scene understanding to learn rich semantics from surgical endoscopic videos. In comparison, Chen et al. (2024) designated an iterative step-by-step reasoning manner for VQA problem, letting MLLMs "See-Think-Confirm" to generate a better rationale for the decided answer. Recently, Shao et al. (2024) proposed a multi-turn processing pipeline that dynamically focuses on visual inputs and provides interpretable thoughts. Moving along with the direction, we aim to facilitate MLLMs on the GUI referring task by generating a chain of lenses with varying field widths, ranging from fine-grained to coarse views. The chain of lenses demonstrates the hierarchical layout of the GUI, simulating a human-like sequence of attention, improving MLLMs on producing robust and accurate descriptions with the content and layout information regarding the target point input in the accessible screen reading task.

3 Tree-of-Lens Agent

3.1 Task Definition

We aim to develop an agent to interpret designated regions on GUIs indicated by points, which we refer to as the Screen Point-and-Read (ScreenPR) task. This agent needs to process input consisting of a screenshot S_i and a point coordinate P_i on the screen, where the point represents a region of the user's interest. Two target outputs are language descriptions $\hat{D}_i^c$ for the content of the targeted region and $\hat{D}_i^l$ for the layout information of the screen related to the targeted region.

3.2 Method Overview

Targeting the ScreenPR task, we propose the Tree-of-Lens (ToL) agent equipped with the ToL grounding mechanism, as shown in Figure 2. Since our agent relies on the pure visual modality of the GUI and leverages the strong visual referring capabilities of MLLMs, such as GPT-4o, it can effectively read anywhere pointed on screens from any domain, including but not limited to web, mobile, and operating systems.

The three distinct design goals distinguish our LoT agent from previous models. First, our LoT agent outputs rich, natural language descriptions rather than fixed template descriptions from accessible screen reading tools such as VoiceOver[1]. Second, it can handle any point location on the screen as input by dynamically detecting the pointed region, which might be as small as an icon or a broader area, such as a full-screen window. Last but not least, the descriptions generated by our

[1] https://support.apple.com/guide/voiceover

agent encompass both the content and layout perspectives, detailing the targeted region's function and position within the GUI's layout.

3.3 Hierarchical Layout Tree Construction

Hierarchical Layout Tree The layout of the GUI screenshot usually features an inherent hierarchy decided by GUI source codes. We propose the Hierarchical Layout Tree to represent this hierarchical structure. This tree structure corresponds to each screenshot, with each node representing a square region within the GUI screenshot. Instead of using varied depths of tree structures from the underlying source codes of GUIs, for simplicity, the proposed Hierarchical Layout Tree has a fixed 3-layer depth for every GUI screenshot, with nodes in each layer representing regions of varied hierarchies. The top layer node represents the whole screenshot. The middle layer nodes represent global regions, which encompass larger, comprehensive areas that include multiple related elements, such as panels or groups of controls, forming significant parts of the user interface. The leaf nodes represent local regions, referring to smaller, more specific areas, typically denoting individual interactive elements like buttons, text fields, and icons.

Model and tree construction pipeline In order to extract the Hierarchical Layout Tree that depicts the hierarchical layout of screenshots without relying on such structured text modality, we train a GUI region detection model to detect the local and global regions for each GUI screenshot, then construct the Hierarchical Layout Trees accordingly. The GUI region detection model is fine-tuned on the DINO detection model with ResNet-50 backbone (Zhang et al., 2022; Chen et al., 2019). Once the regions are detected, we construct the Hierarchical Layout Tree, where each predicted global region serves as a node connected to one or more leaves representing the predicted local regions. The connection between global and local regions is determined by the highest IoU between the local and global regions, where for a predicted local region $\hat{R}_i$, its parent global region is $\hat{R}_j, j = \mathit{ArgMax}_j \mathit{IoU}(R_i, R_j)$.

Android Screen Hierarchical Layout dataset To train a detection model for the Hierarchical Layout Tree, we build a dataset named the Android Screen Hierarchical Layout (ASHL) dataset, featuring screenshots and bounding boxes labeled as either global or local regions. There are around 52k bounding boxes from 2,180 screenshots, spanning all 18 different Android applications.

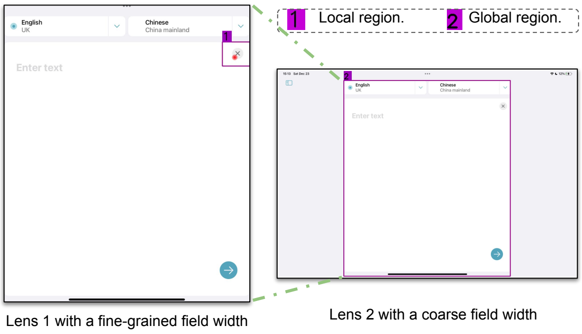


Figure 3: Example of the lenses generated from the Hierarchical Layout Tree based on a point coordinate. Lens 2 can be seen as a zooming-out from Lens 1. ToL agent's corresponding output is in Appendix A.

The dataset collection process is fully automated based on the MagicWand simulator (Ding et al.), where we first sample screenshots along with their corresponding source codes based on pre-recorded actions of humans navigating different Android applications. Then, we extract GUI regions R_i from the source codes with pre-defined coordinates to become nodes and leaves in a tree structure. The parent-child relationships $\mathit{Parent\text{-}Child}(R_i, R_j) \mathit{establish} = 1$ are also inherent directly from the code. Next, we refine the tree by pruning branches corresponding to non-visible regions and merging connected nodes with an Intersection Over Union (IoU) greater than 0.9 to a single node, representing the region R_{i+j}, where

$$\begin{aligned} & R_{i+j} = \mathit{Max}(R_i, R_j), \\ \textbf{if} \quad & \mathit{IoU}(R_i, R_j) > 0.9 \\ & \& \ \mathit{Parent\text{-}Child}(R_i, R_j) == 1. \end{aligned} \tag{1}$$

Notably, we find that the merged connected nodes R_{i+j} often indicate regions having united semantic contents due to their multiple code attributes. Therefore, we further label those merged nodes with more than one leaf as global regions and the corresponding leaves as local regions:

$$\text{Global Regions} = \{R_{i+j} | \mathit{len}(\mathit{Leaf}(R_{i+j})) > 1\} \tag{2}$$

$$\text{Local Regions} = \{\mathit{Leaf}(R_{i+j}) | \mathit{len}(\mathit{Leaf}(R_{i+j})) > 1\}, \tag{3}$$

where $\mathit{Leaf}(R_i)$ outputs all the leaves in the tree tracing down from the node of region R_i.

3.4 Target Path Selection and Multi-lense Prompting

Based on the constructed Hierarchical Layout Tree, we next select a target path from the tree and generate a series of image prompts with annotations based on the input screenshot S_i and point P_i. The image series represents different "lenses" with varying field widths and becomes the visual prompt for the MLLM, GPT-4o, which we refer to as Multi-lense prompting.

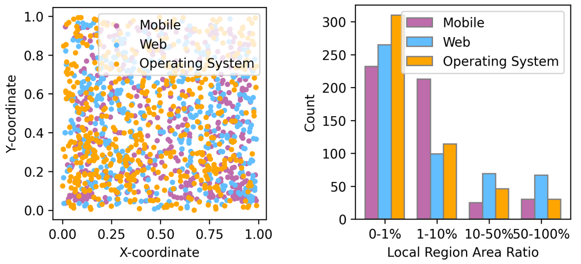
The process first identifies the smallest local region that contains the input point. Then the connected parent global region is extracted from the Hierarchical Layout Tree[2]. We next curate two lenses in a sequence that visually articulates this hierarchical relationship from a fine-grained to a coarse view: the first lens shows only the global region from the screenshot while marking the local region with a colored bounding box labeled ‘1’, the target point with a semi-transparent red dot; the second lens includes the complete input screenshot and marks the global region with another bounding box labeled ‘2’. Figure 3 shows an example.

Then a Multi-lens Prompting method is adopted, where the curated set of lenses is sent to GPT-4o with a text prompt which includes the text coordinate of the target point. Some specific instructions are also included in the prompt, asking, "Explain where box 1 is in relation to box 2 and where box 2 is located on the overall screen." In this way, we provide clear visual cues that facilitate the GPT-4o to generate detailed descriptions that reflect the GUI’s content and layout. For more details about the prompt used, please refer to Appendix D.

4 Screen Point-and-Read Benchmark

4.1 Overview

We introduce the Screen Point-and-Read (ScreenPR) benchmark to rigorously evaluate our ToL agent on the ScreenPR task, where descriptions are required based on user-indicated points on screenshots. As shown in Table 1, this benchmark covers a diverse domain of GUIs with a total 650 screenshots. Each screenshot is accompanied by manually verified target points P_i, their corresponding local regions R_i, and content descriptions. Figure 4i illustrates the normalized coordinates of target points P_i in the ScreenPR benchmark, demonstrating comprehensive coverage of every location on the screen. The regions included have various content, and their area ranges from 1% to over 50% of the screen, as shown in Figure 4ii.



(i) (ii)

Figure 4: (i) shows the normalized locations of target points P_i in the input screenshot S_i. (ii) shows distributions of the area of local regions R_i in our ScreenPR benchmark.

Domain	Screenshots	Target Points	Local Regions
Web	199	500	500
Mobile	201	500	500
Operating System	250	500	500
Total	650	1,500	1,500

Table 1: Key Statistics of the ScreenPR Benchmark

4.2 Data Collection

We first gather mobile and web screenshots from Screenspot (Cheng et al., 2024) and operating system screenshots from the data explorer videos[3] of OSWorld (Xie et al., 2024). Then we randomly generate 4 or 5 points as the candidates for target points P_i within each screenshot. Next, we hire 3 students as annotators to draw bounding boxes for the local regions R_i corresponding to the points. To maximize the variety of the data, candidate target points that represent duplicated local regions will be removed. Finally, we assign the remaining points within each screenshot as either target points P_i or reference points $P_{i'}$, where they are paired such that every pair of $(P_i, P_{i'})$ all have non-overlapping corresponding local regions ($P_{i'}$ is used for our proposed *cycle consistency evaluation* detailed in Section 4.3). Additionally, we collect human-verified descriptions for all local regions R_i corresponding to P_i. These descriptions are captions generated using GPT-4o with the direct input of $Crop(S_i, R_i)$, and we manually examined and corrected mistakes. After the annotation, we

[2] In some rare cases, the input point is outside all local regions, we treat the global regions as local regions in this process and the parent of the global region is the full screenshot.

[3] https://os-world.github.io/explorer.html

ensure the data quality by first using scripts to spot errors, such as ensuring that local regions R_i are included P_i inside, and then manually examine the data.

4.3 Evaluation

The ScreenPR benchmark evaluates both the accuracy of the generated content description $\hat{D}^c$ and the generated layout description $\hat{D}^l$. Since there is no ground truth for the generated descriptions, we involve human evaluations and also propose automated cycle consistency evaluations inspired by the two-agent evaluation in Padmakumar et al. (2022); Fan et al. (2023).

Human evaluation We conduct surveys with human judges to assess the quality of the generated descriptions. Each survey presents a screenshot with an indicated point, accompanied by descriptions from different screen readers. The judges are asked to rate these descriptions based on three criteria: how well they describe the content, how well they articulate the layout, and their overall preference. Each question has four options, ranging from "very well" to "not at all," which we numerically mapped to accuracy scores of 100%, 66%, 33%, and 0%, respectively.

Cycle consistency evaluation In order to evaluate screen reader models automatically, we propose the cycle consistency evaluation. The outputs $(\hat{D}^c, \hat{D}^l)$ from the screen reader are each used as the input for an auxiliary model tasked with specific validation tasks, with the alternative model's performance indicating the quality of the screen reader's descriptions.

Specifically, to evaluate content accuracy, GPT-4V (Achiam et al., 2023) serves as the auxiliary model, performing a multi-choice selection task. This alternative model is asked to select one from the multi-choice candidates based on the predicted descriptions $\hat{D}^c$. The candidates are four screenshots of different GUI regions cropped based on human annotation, with only one targeted region

$$\text{Multi-choice question} = Q \tag{4}$$

$$\begin{aligned}&\text{Multi-choice candidates}(S_i, P_i, \hat{D^c}_i)\\ &\quad = \{\mathit{Crop}(S_{k_j}, R_{k_j}) | j \in [0, 4],\\ &\qquad\qquad k_0 = i, k_1, k_2, k_3 \in N\},\end{aligned} \tag{5}$$

where Q is the fixed question and $\mathit{Crop}(S_i, R_i)$ means the screenshot S_i cropped at the region R_i.

For layout description accuracy, we design another multi-choice selection task for the auxiliary model, GPT-4 (Achiam et al., 2023). This multi-choice selection task requires the auxiliary model to select the positional relationship of two regions indicated by P_i and $P_{i'}$ based on corresponding generated layout descriptions $\hat{D^l}_i$ and $\hat{D^l}_{i'}$. $P_{i'}$, the reference points, collected in the same way as target points P_i (detailed in Section 4.2) are manually annotated to represent a region $R_{i'}$ that does not overlap with the local region of the target point R_i on the same screen S_i, especially for the evaluation of the layout description accuracy. The multi-choice selection task presented to the auxiliary model is

$$\text{Multi-choice question} = Q_{\text{gen}}(\hat{D^l}_i, \hat{D^l}_{i'}) \tag{6}$$

$$\text{Multi-choice candidates} = C, \tag{7}$$

where Q_{gen} is a question template that needs to be filled with the layout descriptions generated, and C are some fixed choices provided such as 'upper,' 'lower,' 'left,' etc. Based on the ground truth location of R_i and $R_{i'}$, we calculate the accuracy of the choices made by the auxiliary model, which indicates the screen reader's proficiency in describing the layout.

5 Experiments

Baselines We adopt several baselines, including generalist MLLMs: the proprietary GPT-4o (OpenAI, 2024) and the open-source LlaVA-NeXT (Liu et al., 2024a). We also test the CogAgent (Hong et al., 2023), a MLLM specifically tuned for GUI tasks, and a visual prompting method Scaffolding (Lei et al., 2024) that is specially developed to enhance the spacial reasoning ability of MLLMs. The inputs for baselines are the text prompt with the input point coordinate P_i and the screenshot S_i. We keep prompts for baselines in the same style as the ToL agent, except that we have to simplify it for CogAgent as we find it necessary for CogAgent to follow the instructions. Detailed prompt design is shown in Appendix D.

Training We divide the ASHL dataset into 90 % training and 10% evaluation sets for the GUI region detection model training process. Training is performed on four NVIDIA A6000 GPUs with a batch size of 8, spanning 90 epochs. The optimized model demonstrated excellent performance,

	Avg. Len.	Human evaluation (%)			Cycle consistency evaluation (%)		Language Similarity	
		Content	Layout	Overall	Content Acc	Layout Acc	BERTScore	ROUGE
LlaVA-NeXT	51.95	21.69	9.59	21.19	25.43	7.20	85.18	16.52
CogAgent	30.69	22.18	13.33	20.62	25.86	8.00	82.31	15.49
GPT-4o	34.44	32.73	28.12	30.88	35.10	21.87	85.76	15.08
GPT-4o+Scaffolding	28.51	35.21	29.54	33.28	47.82	25.60	86.25	17.25
ToL Agent	38.37	**41.82**	**33.42**	**41.63**	**74.96**	**39.67**	**86.73**	**19.47**

Table 2: Main results. We evaluate our ToL agent against three other baselines with the human evaluation and cycle consistency evaluation of our ScreenPR benchmark. Additionally, we compare the generated content descriptions with human-verified content descriptions using language similarity scores. The results show that the ToL agent achieves the best performance.

	Content Acc	Layout Acc
GPT-4o	35.10	21.87
ToL Agent (GPT-4o)	**74.96**	**39.67**
Qwen-VL-Plus	28.29	12.50
ToL Agent (Qwen-VL-Plus)	**41.93**	**14.47**
Gemini Pro Vision	33.20	12.27
ToL Agent (Gemini Pro Vision)	**45.33**	**18.53**

Table 3: Results of the ToL Agent built upon various MLLMs. The results of our ToL agent based on various MLLMs are consistently better than the MLLMs alone, providing strong evidence that our approach is effective and generalizable. All the MLLMs are accessed through their APIs around Jun 2024.

	Content Acc	Layout Acc
ToL Agent	75.0	39.7
+ w/o multi-lens visual prompt	71.2 (-3.8)	37.7 (-2.0)
+ w/o target point mark (≈ SoM)	65.6 (-9.4)	37.6 (-2.1)
+ w/o local & global regions marks (degrade to vanilla GPT-4o)	35.1 (-29.9)	21.9 (-18.6)

Table 4: Ablation results. We gradually refine the ablation to exclude various processes in our ToL grounding mechanism for the ToL agent on the ScreenPR benchmark. The results show the effectiveness of various components.

achieving an Average Precision (AP) of 94.1% and an Average Recall (AR) of 95.9%.

5.1 Main Results

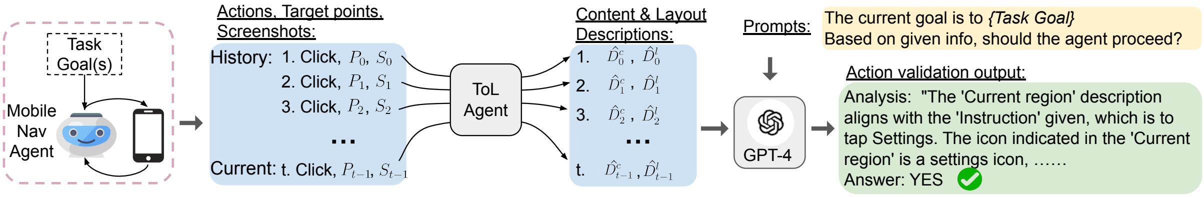
We evaluate our ToL agent against three baselines on the ScreenPR benchmark as shown in Table 2. The results are consistent across both human evaluation and automatic cycle consistency evaluation, showing that our ToL agent achieves the best performance in terms of content and layout description accuracy. Notably, our ToL agent demonstrates over a 15% improvement compared to the GPT-4o model. Furthermore, the overall scores from the human evaluation indicate that descriptions generated by our ToL agent are the most favorable. Interestingly, although the results show that the CogAgent performs slightly worse than LlaVA-NeXT in terms of the content and layout description accuracy, the overall scores from human judges slightly prefer LlaVA-NeXT. We believe this preference is due to the LlaVA-NeXT model's tendency to produce longer descriptions with more information about the screen, creating a more comprehensive feel.

Additionally, we leverage language similarity scores, including BERTScore and ROUGE-L, to compare the generated content descriptions with human-verified descriptions. We observe that the overall trend from the language similarity evaluation results matches the content evaluation results from human and cycle consistency evaluations. However, the differences in language similarity scores between models are relatively indistinguishable. We believe this is because the captions are more comprehensive, sometimes redundant, whereas the screen reading output is more focused on delivering the key features of the content to help users understand it.

To further evaluate the generalizability of our ToL agent pipeline, we conduct experiments using the cycle consistency evaluation on our ToL agent with two other different MLLMs rather than the default GPT-4o: the Qwen-VL-Plus model (Bai et al., 2023) and the Gemini Pro Vision model (Team et al., 2023). We access both of these models via their APIs. As shown in Table 3, the results show that our ToL agent built upon various MLLMs consistently outperforms the standalone MLLMs, offering compelling evidence that our approach is effective and generalizable.

5.2 Ablation Study

Effectiveness analysis To assess the effectiveness of the ToL design, we conduct an ablation


Task Goal(s)
Mobile Nav Agent
Actions, Target points, Screenshots:
History: 1. Click, P_0, S_0
2. Click, P_1, S_1
3. Click, P_2, S_2
...
Current: t. Click, P_{t-1}, S_{t-1}
ToL Agent
Content & Layout Descriptions:
1. $\hat{D}^c_0$, $\hat{D}^l_0$
2. $\hat{D}^c_1$, $\hat{D}^l_1$
3. $\hat{D}^c_2$, $\hat{D}^l_2$
...
t. $\hat{D}^c_{t-1}$, $\hat{D}^l_{t-1}$
GPT-4
Prompts:
Action validation output:

The current goal is to *{Task Goal}*
Based on given info, should the agent proceed?

Analysis: "The 'Current region' description aligns with the 'Instruction' given, which is to tap Settings. The icon indicated in the 'Current region' is a settings icon, ...
Answer: YES ✓

Figure 5: Pipeline of employing our ToL agent in verifying the actions from a mobile navigation agent.

	Content Acc	Layout Acc
Human	92.8	70.3
ToL Agent	72.5 (-20.3)	41.3 (-29.0)
ToL w/ GT local regions	85.5 (-7.3)	44.3 (-26.0)
ToL w/ GT global regions	88.2 (-4.6)	46.6 (-23.7)
ToL w/ GT global local regions	89.5 (-3.3)	48.2 (-22.1)

Table 5: Bottleneck analysis. Replacing local and global regions with GT annotations mainly improves content description accuracy, indicating rooms of improvements for both the GUI region generation process and the MLLM leveraged.

study where we progressively refine the ablation to exclude key design components and evaluate these ablated versions on our ScreenPR benchmark using the cycle consistency evaluation, as shown in Table 4. First, we modify our ToL agent to exclude the multi-lens visual prompting. This change necessitates annotating bounding boxes for the local region, global region, and the target point on the complete screenshot as a single visual prompt. Then we further remove the red dot mark from the visual prompt, resulting in a design similar to the Set-of-Mark (SoM) prompting method (Yang et al., 2023a). As a result, these changes lead in performance degradation in both content and layout description accuracy, indicating the effectiveness of our visual prompting strategies. Finally, we ablate the input visual prompt to remove marks of local and global regions that we extract from our Hierarchical Layout Tree. This drastic modification results in a significant decrease in performance, underscoring the importance of our Hierarchical Layout Tree in the of the ToL design.

Bottleneck analysis To examine the bottlenecks in our ToL agent, we replace the local and global regions generated from the Hierarchical Layout Tree with human annotations, considered as the ground truth (GT). We also assess human performance by having human annotators write both content and layout descriptions manually. Due to the high cost of human efforts, we conduct experiments on a 20% random sample of the ScreenPR benchmark. As shown in Table 5, substituting either GT local regions or GT global regions significantly close the gap toward human performance in the ToL agent's accuracy of content descriptions. However, for layout descriptions, the gap remains, indicating that the MLLM we leverage is the bottleneck in this aspect. This finding aligns with the conclusion from (Fan et al., 2024), which highlights that current MLLMs face significant challenges in understanding the layout of multipanel images.

5.3 Broader Application: Verification of Mobile Navigation Agent Actions

In this section, we demonstrate how our ToL agent helps identify incorrect actions for a mobile navigation agent, MagicWonder, on the MagicWand platform (Ding et al.). Driven by a specific task goal, such as "connect to Wifi in settings", MagicWonder can generate actions to interact with a live mobile screen and save the detailed information for each step along its execution trajectory. More details are provided in Appendix E.1. To incorporate our ToL agent for verification of MagicWonder's output actions, we design a simple pipeline. As shown in Figure 5, the layout and content descriptions are generated by our ToL agent for each action-point-screenshot triplet and sent to the GPT-4 model. The GPT-4 model, based on the task goal, instructions, and descriptions from the ToL agent, decides whether the last action from the MagicWonder is correct.

Data sampling We select 52 mobile agent trajectories with specific task goals from instruction-guided executions on the MagicWand platform, with more details in Appendix E.2. Each trajectory consists of multiple execution steps triggered by distinct actions, resulting in a total action number of 209. We hire two students to annotate incorrect actions in the chosen trajectories to get the ground truth labels.

	Predicted as		TP/FP Rate
	Correct	Incorrect	
Correct Action	100.2±2.0	8.8 ±2.0	92% ±0.02
Incorrect Action	61.9 ±2.9	36.1 ±2.9	38% ±0.03

Table 6: Results of leveraging the ToL agent to verify actions from a mobile navigation agent. "Predicted as" column shows the number of predicted actions. True positive and false positive rate are shown on the right most column calculated from the data on the left

Verification Method	**F1 score**	**Repetition Detection Rate**
Confidence Score Filter	65%±0.00	0%±0.00
GPT-4o	61%±0.01	40%±0.10
ToL Agent	**74%±0.01**	**44%±0.04**

Table 7: Performance of different verification methods in verifying actions of mobile navigation agent. The result shows that using our ToL agent to verify the actions of the mobile navigation agent achieves the best F1 score and performs the best in detecting the notable issue of the "execution loop", i.e. repetition detection.

Result We perform 10 end-to-end evaluations and present the mean value and standard deviation in Table 6. Despite minor fluctuations, while 92% correct actions can be stably recognized, about 38% incorrect actions can be identified. This reveals that by incorporating the ToL agent, the performance of the mobile navigation agent is significantly and robustly improved. Our ToL agent can also identify particularly tricky challenges faced by mobile navigation agents during execution. One notable issue is the "execution loop", where agents repeatedly take actions on the same or similar regions without making progress. Our results show that for all 20 incorrect actions that fall into this category, 44% ± 0.04 of them can be successfully detected. To further validate the effectiveness of our method, we introduce two baselines: Confidence Score Filter and GPT-4o Verifier. The Confidence Score Filter considers actions as positive when the relevant confidence score is above 0.7 and the GPT-4o Verifier directly sends the actions with corresponding screenshots as prompts to GPT-4o to identify the correctness of actions. We run the end-to-end evaluation equivalently and the results in Table 7 show that our method of using the ToL agent to verify the output from the mobile navigation agent outperforms the other baselines.

6 Conclusion

In this work, we propose a novel Tree-of-Lens (ToL) grounding method and build a ToL agent for the Screen Point-and-Read (ScreenPR) task, an important GUI referring task that is critical for accessible technology. Our ToL agent outputs layout-aware descriptions for pointed regions on screenshots. With our newly proposed Screen Point-and-Read (ScreenPR) benchmark, we show that our ToL agent outperforms all other baselines in terms of content and layout description accuracy, demonstrating great potential in assisting visually impaired users when using digital devices. Last but not least, in our experiment, we also apply our ToL agent to identify incorrect actions performed by a mobile navigation agent, yielding promising results.

7 Limitations

In this work, we propose a novel Tree-of-Lens (ToL) agent for the Screen Point-and-Read (ScreenPR) task. By leveraging GPT4-o to generate content and layout descriptions, we are able to set up the pipeline quickly and validate the effectiveness of ToL grounding method in a short time period. However, the accumulated server delay in requesting GPT4-o services has occupied more than 80% of end-to-end evaluation, not to mention the high expense accompanying, which is a non-trivial issue for real-world applications. Thus, a more efficient and locally hosted model has to be given out in place of GPT-4o. Moreover, since our ToL agent relies on a certain MLLM to generate descriptions, there are potential risks that the generated descriptions from the MLLM could contain harmful contents. Therefore more post-processing strategy might be needed. Last but not least, another critical challenge is balancing application efficiency with improved semantic representation, thereby enabling on-device support and further expanding ToL's application areas. Our current work is limited for research purpose and more application-wise exploration is pending.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023. Minigpt-v2: Large language model as a unified interface for vision-language multi-task learning. *arXiv:2310.09478*.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. 2019. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

Zhenfang Chen, Qinhong Zhou, Yikang Shen, Yining Hong, Zhiqing Sun, Dan Gutfreund, and Chuang Gan. 2024. Visual chain-of-thought prompting for knowledge-based visual reasoning. In *AAAI Conference on Artificial Intelligence*.

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*.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Lei Ding, Yi Zhang, and Jeshwanth Bheemanpally. Enhancing mobile" how-to" queries with automated search results verification and reranking. In *The Second Workshop on Generative Information Retrieval*.

Yue Fan, Jing Gu, Kaizhi Zheng, and Xin Wang. 2023. R2H: Building multimodal navigation helpers that respond to help requests. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14803–14819, Singapore. Association for Computational Linguistics.

Yue Fan, Jing Gu, Kaiwen Zhou, Qianqi Yan, Shan Jiang, Ching-Chen Kuo, Xinze Guan, and Xin Eric Wang. 2024. Muffin or chihuahua? challenging large vision-language models with multipanel vqa. *arXiv preprint arXiv:2401.15847*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2023. Cogagent: A visual language model for gui agents. *arXiv preprint arXiv:2312.08914*.

Xuanyu Lei, Zonghan Yang, Xinrui Chen, Peng Li, and Yang Liu. 2024. Scaffolding coordinates to promote vision-language coordination in large multi-modal models. *arXiv preprint arXiv:2402.12058*.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024a. Llava-next: Improved reasoning, ocr, and world knowledge.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. *Advances in neural information processing systems*, 36.

OpenAI. 2024. Gpt-4o.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. 2022. Teach: Task-driven embodied agents that chat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2017–2025.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. 2024. Visual cot: Unleashing chain-of-thought reasoning in multi-modal language models. *arXiv preprint arXiv:2403.16999*.

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

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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. *arXiv preprint arXiv:2401.16158*.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. 2023. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Nan Xi, Jingjing Meng, and Junsong Yuan. 2023. Chain-of-look prompting for verb-centric surgical triplet recognition in endoscopic videos. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 5007–5016.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023a. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. 2023. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*.

Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. 2023. Ferret: Refer and ground anything anywhere at any granularity. In *The Twelfth International Conference on Learning Representations*.

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*.

Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. 2022. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *The Eleventh International Conference on Learning Representations*.

Haotian Zhang, Haoxuan You, Philipp Dufter, Bowen Zhang, Chen Chen, Hong-You Chen, Tsu-Jui Fu, William Yang Wang, Shih-Fu Chang, Zhe Gan, et al. 2024. Ferret-v2: An improved baseline for referring and grounding with large language models. *arXiv preprint arXiv:2404.07973*.

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 Example outputs of the ToL Agent

For each GUI domain, we show one example of the output of our ToL agent along with the lenses generated. We do not deliberately cherry-pick the best case but show examples with both strong and weak performances. An example for the operation system domain is shown in Figure 6. An example for the mobile domain is shown in Figure 7. An example for the web domain is shown in Figure 8.

B Examples of the Cycle Consistency Evaluation

We select one result of our cycle consistency evaluation for the content description and show it in Figure 9. The left four images come from the cropped areas from different GUI screens, with only one of which matches the description in the prompt on the right side. As shown, GPT-4o can successfully choose the correct option in the "Answer" field and give its reasoning in the "Analysis" field.

For cycle consistency evaluation regarding the layout description, the question contains the generated layout descriptions from models. We let GPT-4o choose one option from predefined 9 relative positions together with its analysis, referring to the layout evaluation output in Figure 11. The ground truth is provided with script based on the human annotated local regions for the target point and reference point. The ground truth is also checked by human.

Prompt used by LlaVA-NEXT and GPT-4o For the baseline models, LlaVA-NEXT and GPT-4o share the same prompting template as shown in Figure 13.

Prompt used by CogAgent In order to drive CogAgent to follow instructions, a simplified prompting template is applied, as shown in Figure 14.

C Cross-evaluation Experiments with Cycle Consistency Evaluation

Our cycle consistency evaluation, relying on GPT-4V as the auxiliary model, shows that our ToL agent (leveraging GPT-4o by default) outperforms the other baselines including the vanilla GPT-4o. Since the evaluator, the baseline model, and our ToL agent all rely on GPT models, we believe the results regarding models relying on GPT models are less influenced by the potential bias as mentioned by the review. Furthermore, to strengthen our result, we first include the Gemini Pro Vision model as a baseline model and introduce a new version of our ToL agent relying on the Gemini Pro Vision model. Then, we extend our main results with the cross-evaluation setting where the models relying on GPT-4o will not be evaluated with GPT-4V as the auxiliary model, and the models relying on Gemini Pro Vision will not be evaluated with Gemini Pro Vision as the auxiliary model. The results are shown in Table 8, where under this cross-evaluation setting, our ToL agent consistently demonstrates clear superiority over the vanilla MLLM baseline.

D Text Prompt templates

We show the templates of text prompts we used for different MLLMs. We substitute the $\{\{x\}\}$,$\{\{y\}\}$ with the normalized coordinate of the input point.

Prompt used by ToL agent For our ToL agent, we use the text prompt template shown in Figure 12 for GPT-4o.

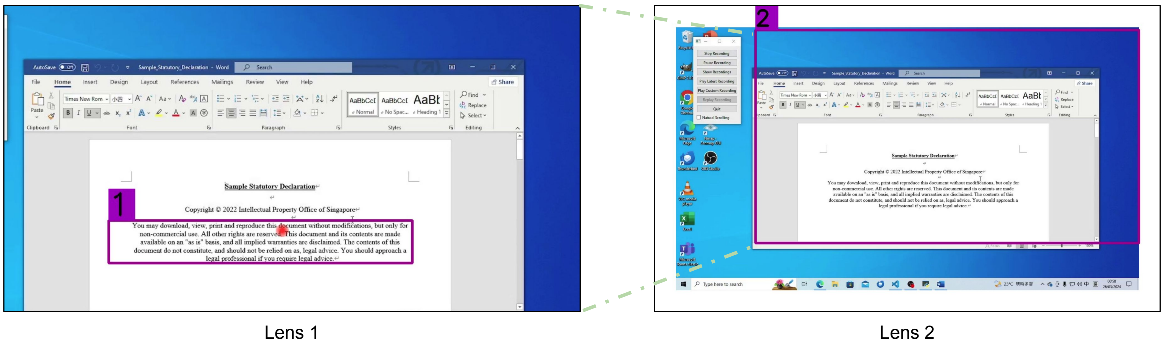
E Mobile Navigation Action Verification

E.1 Example

Figure 16 illustrates one action verification case in our experiment: MagicWonder is asked to accomplish "pause historical track in Youtube App". Len 1 and Len 2 show the agent was attempting to click the "Setting" icon by giving the instruction "Tap Settings" at the second step along the execution trajectory. The first step in action verification is to get the response from the ToL Agent that describes Len 1 and Len 2 correctly with more detailed application contexts. Using this information, GPT-4o could correctly judge whether this action on the screen was in accordance with the given instruction.

E.2 Sample Strategy of Mobile Navigation Action Execution Trajectory

MagicWand simulator can launch mobile agents to take action on devices under the guidance of a series of instructions. The corresponding runtime GUI context, like GUI screenshots, actionable regions, instructions for each step, and action details, are saved into a self-descriptive JSON format. We deliberately choose the instructions from the WeWeb dataset (Ding et al.) having few overlapping with our ASHL dataset. After execution on MagicWand, we chose those trajectories with task



Sample Statutory Declaration

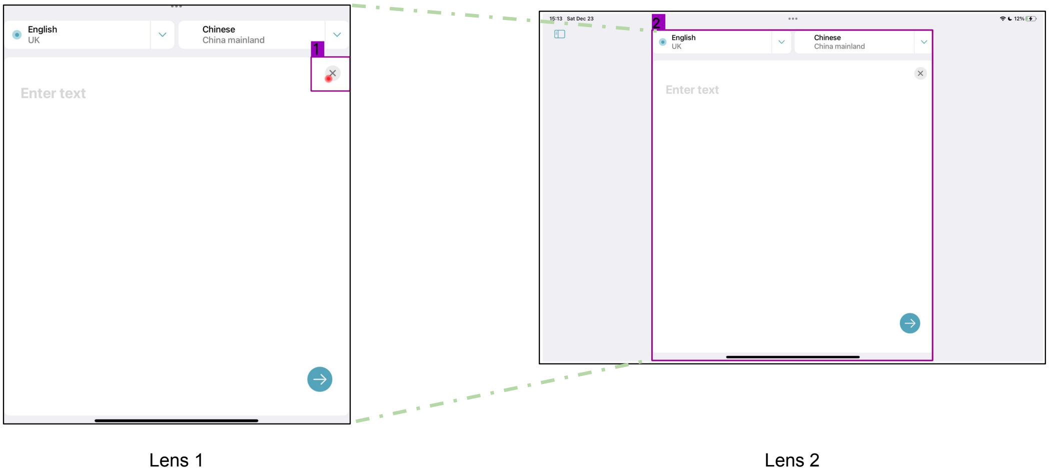
Copyright © 2022 Intellectual Property Office of Singapore

You may download, view, print and reproduce this document without modifications, but only for non-commercial use. All other rights are reserved. This document and its contents are made available on an "as is" basis, and all implied warranties are disclaimed. The contents of this document do not constitute, and should not be relied on as, legal advice. You should approach a legal professional if you require legal advice.

ToL Agent output:

This is a copyright statement and usage disclaimer from the Intellectual Property Office of Singapore. It is located near the center-bottom area of the document shown in the word processor on the screen, which is situated within the larger desktop view containing various icons and applications.

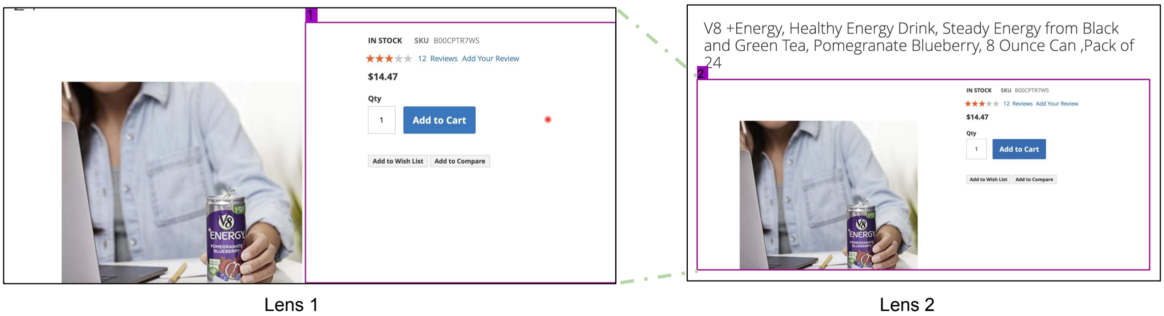
Figure 6: Example of the generated description for an operation system screenshot from our ToL agent with the two lenses generated. The global region generated is not perfect as the bounding box is a little bigger than the application window on the top right part.



ToL Agent output:

This is a close button (X icon). It is located in the upper right corner of the language selection area, which itself is positioned centrally on the screen.

Figure 7: Example of the generated description for a mobile screenshot from our ToL agent with the two lenses generated.


1
IN STOCK SKU B00CPTR7WS
12 Reviews Add Your Review
$14.47
Qty
1
Add to Cart
Add to Wish List
Add to Compare
V8 +Energy, Healthy Energy Drink, Steady Energy from Black and Green Tea, Pomegranate Blueberry, 8 Ounce Can ,Pack of 24
2
Lens 1
Lens 2

ToL Agent output:

This is the section where users can view the status, SKU, price, quantity selector, add to cart button, and options to add to wish list or compare products for a V8 energy drink.
This section is located on the right side within the product information area. The product information area itself is positioned on the right half of the complete screen.

Figure 8: Example of the generated description for a web screenshot from our ToL agent with the two lenses generated. The description output is not perfect regarding the layout information as the "product information area" indicated by the global region is not only "on the right half of the complete screen" but the middle and lower areas of the complete screen.

	Auxiliary Model: GPT-4V			Auxiliary Model: Gemini Pro Vision	
	Content Acc	**Layout Acc**		**Content Acc**	**Layout Acc**
Gemini Pro Vision	33.20	12.27	GPT-4o	43.52	22.33
ToL Agent (Gemini Pro Vision)	**45.33**	**18.53**	ToL Agent (GPT-4o)	**58.44**	**33.93**

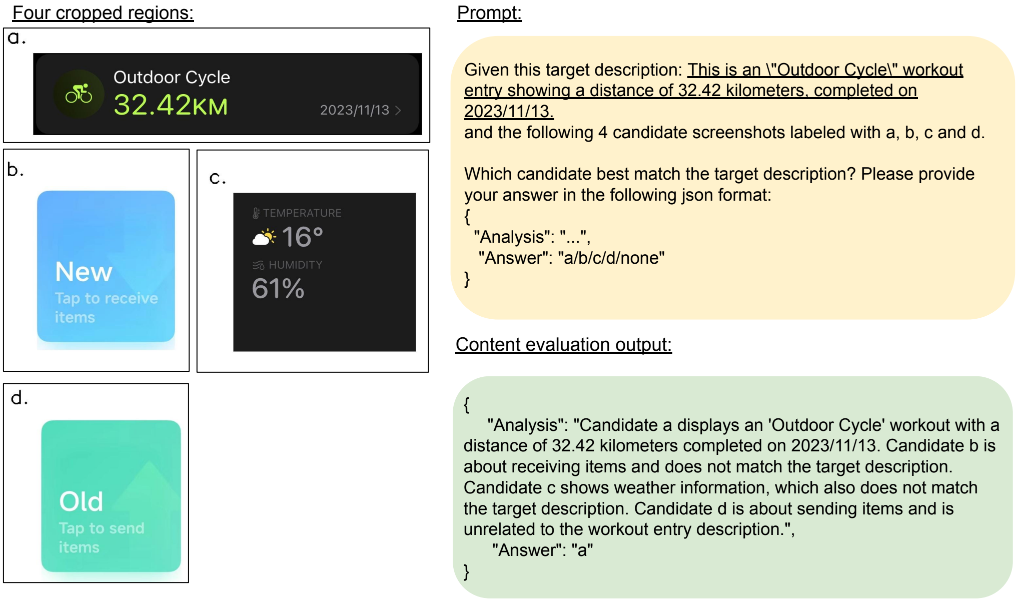
Table 8: Cycle consistency evaluation of the ToL agent and vanilla MLLMs under the cross-evaluation settings. The performance of the ToL agent and the baseline MLLM are evaluated with the auxiliary model being a different MLLM than the one under test. The ToL agent consistently outperforms the baselines.

completion rates less than 50% to evaluate whether ToL information was accurate enough. The selected trajectories cover 11 Android applications from 8 application domains, where the chosen applications include SYSTEM, Entertainment, Education, Food & Drinks, Communication, Shopping, News, Maps & Navigation. The chosen Android apps include Messenger, Google Maps, YouTube, Settings, Quizlet, FlipBoard, McDonalds, eBay, Google Chat, Here WeGo, and DoorDash.

E.3 Action verification Pipeline Details

As shown in Figure 5, a set of navigation actions, target points, and corresponding screenshots, are first recorded during the interaction between the mobile navigation agent and the mobile screen. Then, our ToL agent first processes each pair of target points and screenshots (P_i, S_i), including those from the history path of navigation trajectory and the currently planned one $(i \in [0, t-1])$, where the $P_t - 1$ is the target point that the navigation agent plans to interact with for the next step. As a result, the layout and content descriptions $\hat{D}_i^c$, $\hat{D}_i^l$ are generated. Next, we concatenate all the descriptions and send them to the GPT-4 to determine whether agent actions follow the given task goal.

Target path selection For each mobile screenshot saved in the execution trajectory, the DINO detection model extracts all global regions with confidential scoring larger than 0.15 and local regions with confidential scoring larger than 0.05, consistent with our baseline evaluation. For click action, ToL first finds the smallest local and global regions that contain the click point. Usually, DINO's output aligns with regions that have clear pixel boundaries. To improve the accuracy of local regions with smaller sizes, we extend DINO's output regions by 50 pixels. If no match is found, we further extend the boundaries horizontally, following general GUI design principles. Different from click action, input action involves a region with a clear-cut boundary. Accordingly, ToL selects the smallest local region with an IoU value greater than 0.4 with the input region and the corresponding global region with an IoU value greater than 0.1 with the input region. With the chosen local and global regions, ToL follows Chain-of-Lens prompting mentioned in section 3.4 to get region description.



Given this target description: This is an \"Outdoor Cycle\" workout entry showing a distance of 32.42 kilometers, completed on 2023/11/13.
and the following 4 candidate screenshots labeled with a, b, c and d.

Which candidate best match the target description? Please provide your answer in the following json format:

```
{
  "Analysis": "...",
   "Answer": "a/b/c/d/none"
}
```

```
{
    "Analysis": "Candidate a displays an 'Outdoor Cycle' workout with a distance of 32.42 kilometers completed on 2023/11/13. Candidate b is about receiving items and does not match the target description. Candidate c shows weather information, which also does not match the target description. Candidate d is about sending items and is unrelated to the workout entry description.",
      "Answer": "a"
}
```

Figure 9: One example of our cycle consistency evaluation for content description. A predicted description is shown on the right and four different cropped screen regions were sent as four separate images to GPT-4o. GPT-4o choose the matched region or answer with "unknown".

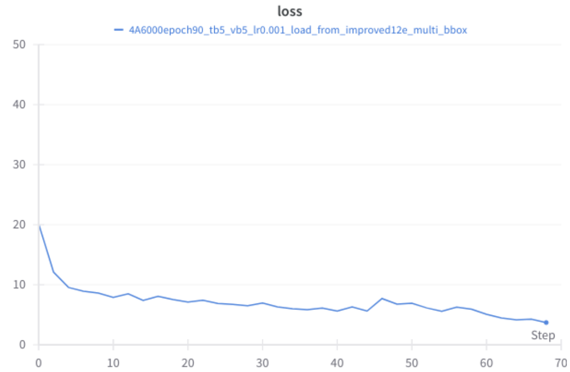


Figure 10: Training Loss of ResNet-50 backbone with epoch 90.

Action verification We prompt GPT-4o by using the current action name, instruction, and region description at the current step, together with historical action names and region descriptions, letting it judge whether the mobile agent can proceed. Only two general rules are mentioned in the prompt:

1. The agent should not proceed if the "Current region" is repeated too often in "Historical action and region description"

2. The agent may proceed if the "Current region" aligns with "Instruction"

GPT4o's response includes a simple yes or no and a detailed reason.

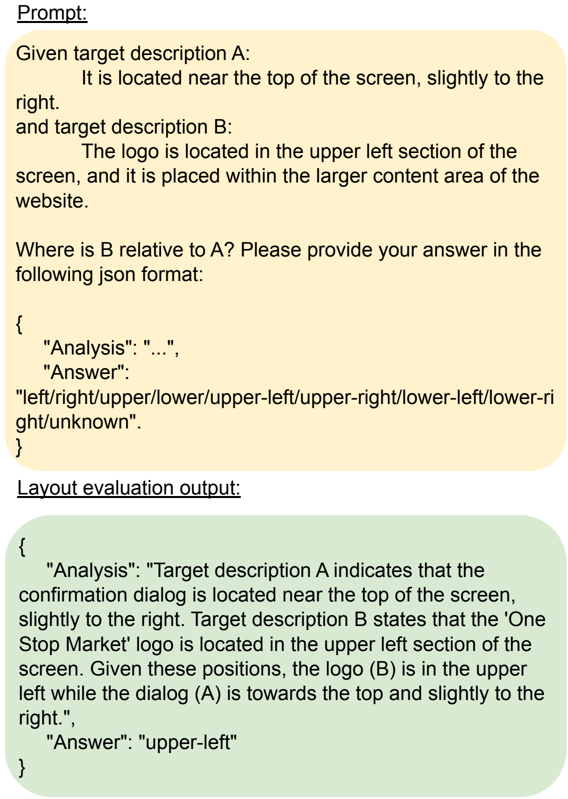
Prompt used by GPT-4o in Action Verification To evaluate the role of LoT in mobile agents, we only list historical action and region description, current region, and the instruction for the current execution step together with the format we expect. During runtime, the aforementioned information will be used to replace variables in the prompting template {{haction}}, {{instruction}}, {{region}}, {{action}} separately.

We briefly mention two high-level goals that the agent is trying to achieve and expect it to figure out details via ToL information. More fine-grained work could be introduced later to get even better performance of action validation, though not our main focus now.

F Human Evaluation

We leverage human evaluation to test the performance of different screen readers on our ScreenPR benchmark. We leverage the Amazon Mechanical Turk[4] platform to find annotators to complete the survey that we designed for our human evaluation. We hired 3 human judges for each data sample. The judges are asked to rate the generated descriptions based on three criteria: how well they describe the content, how well they articulate the layout, and their overall preference. Each question has four options, ranging from "very well", “fair”, “not well”, and "awful," which we numerically mapped

[4] https://www.mturk.com



Prompt:

Given target description A:
It is located near the top of the screen, slightly to the right.
and target description B:
The logo is located in the upper left section of the screen, and it is placed within the larger content area of the website.

Where is B relative to A? Please provide your answer in the following json format:

```
{
    "Analysis": "...",
    "Answer": "left/right/upper/lower/upper-left/upper-right/lower-left/lower-right/unknown".
}
```

Layout evaluation output:

```
{
    "Analysis": "Target description A indicates that the confirmation dialog is located near the top of the screen, slightly to the right. Target description B states that the 'One Stop Market' logo is located in the upper left section of the screen. Given these positions, the logo (B) is in the upper left while the dialog (A) is towards the top and slightly to the right.",
    "Answer": "upper-left"
}
```

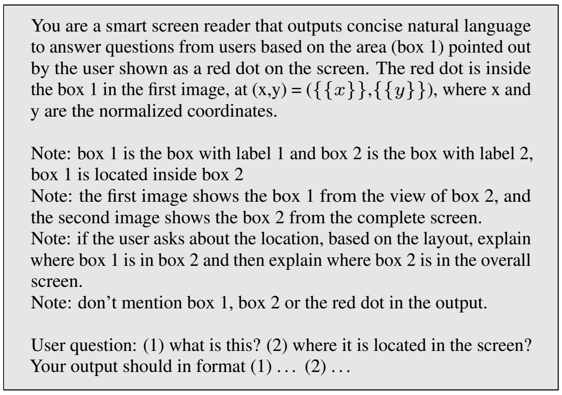
Figure 11: One example of our cycle consistency evaluation for layout descriptions. The layout descriptions for target point and reference point are provided in prompt and GPT-4o is asked to decide their spatial relationship by choosing one option from the given 9 options.

to accuracy scores of 100%, 66%, 33%, and 0%, respectively. We pay $0.15 for each survey and the screenshot of the survey is shown in Figure 18. Approximately 81% of the data samples receive at least two identical ratings from the human judges, and the standard deviation is 19.4%.

G GUI Region Detection Model

Training details Using MMDetection framework (Chen et al., 2019), we change the training into two-label classification according to our task setting. Considering our dataset volume, we load pre-trained weights yielding the best performance on COCO, instead of training from scratch. We train the model for 90 epochs with a MultiStepLR scheduler set by milestones per 30 epochs and γ = 0.1, the loss had further reduced to 2.5803 with small fluctuation, as shown in Figure 10.

We also examine the Average Precision metrics with different IoU and Area on the validation set: they show a clear convergence trend after epoch 70, and the results on the validation set are shown in Table 9.

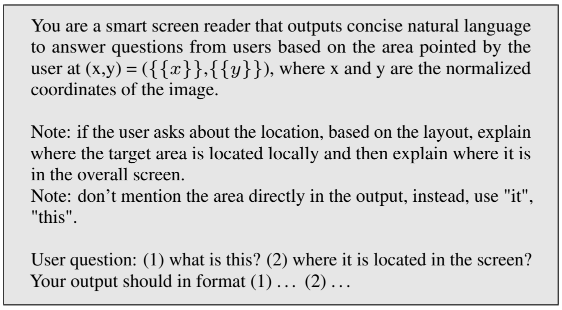


You are a smart screen reader that outputs concise natural language to answer questions from users based on the area (box 1) pointed out by the user shown as a red dot on the screen. The red dot is inside the box 1 in the first image, at (x,y) = ($\{\{x\}\}$,$\{\{y\}\}$), where x and y are the normalized coordinates.

Note: box 1 is the box with label 1 and box 2 is the box with label 2, box 1 is located inside box 2
Note: the first image shows the box 1 from the view of box 2, and the second image shows the box 2 from the complete screen.
Note: if the user asks about the location, based on the layout, explain where box 1 is in box 2 and then explain where box 2 is in the overall screen.
Note: don't mention box 1, box 2 or the red dot in the output.

User question: (1) what is this? (2) where it is located in the screen? Your output should be in format (1) . . . (2) . . .

Figure 12: Prompt used by ToL agent



You are a smart screen reader that outputs concise natural language to answer questions from users based on the area pointed by the user at (x,y) = ($\{\{x\}\}$,$\{\{y\}\}$), where x and y are the normalized coordinates of the image.

Note: if the user asks about the location, based on the layout, explain where the target area is located locally and then explain where it is in the overall screen.
Note: don't mention the area directly in the output, instead, use "it", "this".

User question: (1) what is this? (2) where it is located in the screen? Your output should be in format (1) . . . (2) . . .

Figure 13: Prompt used by LlaVA-NEXT and GPT-4o

Inference Examples and Evaluation Results After training, we utilize the GUI region detection model on unseen OS, Web, and mobile screenshots in our ScreenPR benchmark. We show four examples of the detection results in Figure 17. The identified local regions are marked by green rectangles, and the global regions are highlighted in red. Additionally, we manually annotate global regions for 20% of randomly selected data from ScreenPR benchmark. With the human-annotated regions for the target points, we calculate the accuracy of our trained GUI region detection model as shown in Figure 10. Trained on only mobile data, our model shows generalization ability to other unseen domains.

Except for some undetected ones, many local and global regions were successfully identified with promising confidential scores. We assume that GUI elements share some common design patterns learned by DINO and guarantee good performance for zero-shot inference on heterogeneous platforms.

What is indicated by the point at (x,y) = ({{x}},{{y}}), where x and y are the normalized coordinates of the image.
and where it is located in the screen?

Figure 14: Prompt used by CogAgent

Given the following information in a mobile navigation task:
Historical action and region description: {{haction}}
Task Goal: {{goal}}
Current region: {{region}}

The agent now is going to interact with the "Current region" with the action: {{action}}. Should the agent proceed?
Note: The agent should not proceed if the "Current region" is repeated too often in "Historical action and region description".
Note: The agent may proceed if the "Current region" aligns with "Task Goal".

Please provide your answer in the following JSON format:

```
{
"Analysis": "...",
"Answer": "yes/no"
}
```

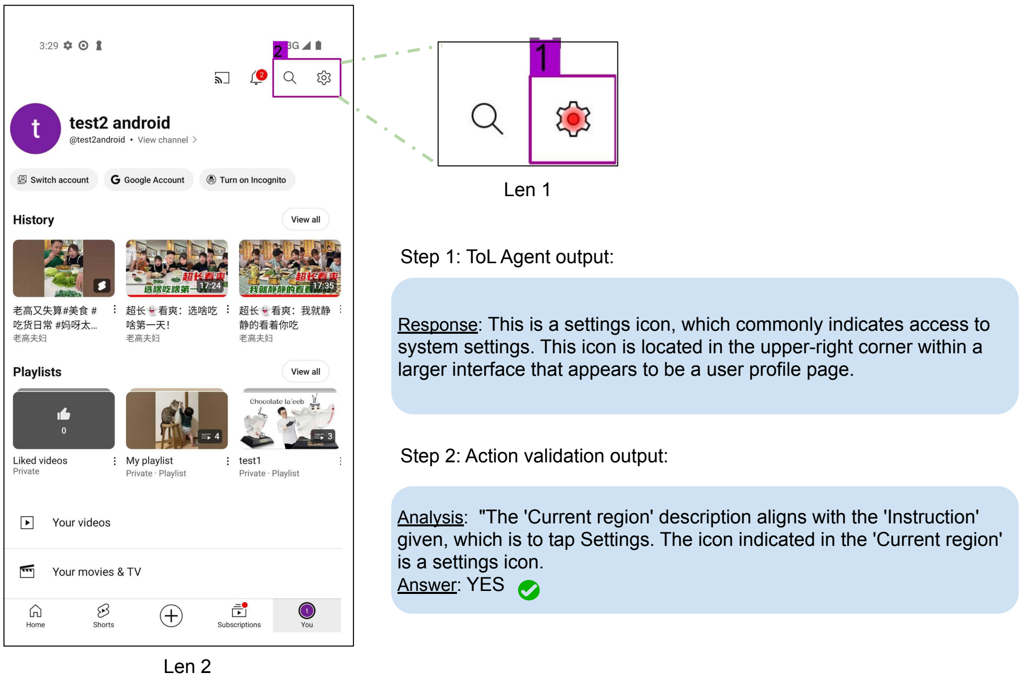
Figure 15: Prompt used by GPT-4o in Action Validation

IoU	Area	Average
0.50:0.95	all	0.9410
0.50	all	0.9620
0.75	all	0.9470
0.50:0.95	small	0.702
0.50:0.95	medium	0.897
0.50:0.95	large	0.943

Table 9: Average precision on the validation set for GUI region detection model. According to the definition in the Microsoft COCO dataset, the small and medium areas in an image are set as less than 1024 (32*32), 9216 (96*96), and others are large areas in the MMDetection framework.

Domains:	Mobile	Web	OS	Avg.
Local Region	38.4%	39.4%	10.0%	29.3%
Global Region	45.6%	35.8%	51.0%	44.1%

Table 10: Accuracy of the generated local region and global regions from our trained GUI region detection model compared with human annotation with IoU@0.5. Since the local regions are small, it brings a great challenge.



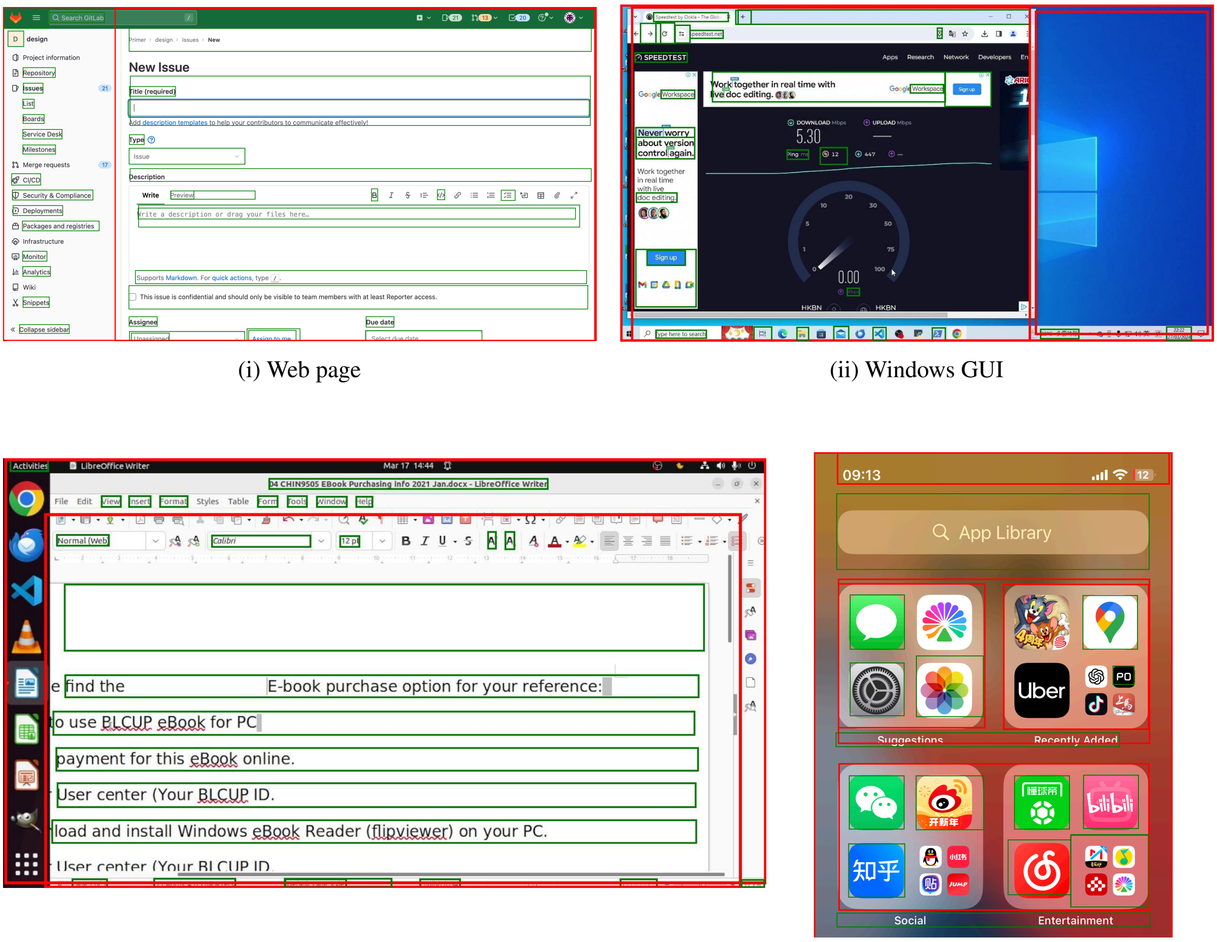
Step 1: ToL Agent output:

Response: This is a settings icon, which commonly indicates access to system settings. This icon is located in the upper-right corner within a larger interface that appears to be a user profile page.

Step 2: Action validation output:

Analysis: "The 'Current region' description aligns with the 'Instruction' given, which is to tap Settings. The icon indicated in the 'Current region' is a settings icon.
Answer: YES ✓

Figure 16: Given the goal of "pausing historial track in Youtube App", mobile Agent on MagicWand plans to click on the screen. ToL Agent generates descriptions for the action planed. The description is sent as input for action verification reasoning, where GPT-4o model output the analysis with decision, which is shown on the bottom right. The GPT-4o concludes that the click action is correct.


Search GitLab
design
Project information
Repository
Issues
List
Boards
Service Desk
Milestones
Merge requests
CI/CD
Security & Compliance
Deployments
Packages and registries
Infrastructure
Monitor
Analytics
Wiki
Snippets
Collapse sidebar
New Issue
Title (required)
Add description templates to help your contributors to communicate effectively!
Type
Issue
Description
Write
Preview
Write a description or drag your files here…
Supports Markdown. For quick actions, type /.
This issue is confidential and should only be visible to team members with at least Reporter access.
Assignee
Due date
SPEEDTEST
Work together in real time with live doc editing.
Never worry about version control again.
Sign up
DOWNLOAD Mbps
UPLOAD Mbps
5.30
0.00
HKBN
Type here to search
Activities
LibreOffice Writer
04 CHIN9505 EBook Purchasing info 2021 Jan.docx - LibreOffice Writer
File
Edit
View
Insert
Format
Styles
Table
Form
Tools
Window
Help
Normal (Web)
Calibri
12 pt
e find the E-book purchase option for your reference:
o use BLCUP eBook for PC
payment for this eBook online.
User center (Your BLCUP ID.
load and install Windows eBook Reader (flipviewer) on your PC.
User center (Your BLCUP ID.
09:13
App Library
Uber
Suggestions
Recently Added
知乎
bilibili
Social
Entertainment

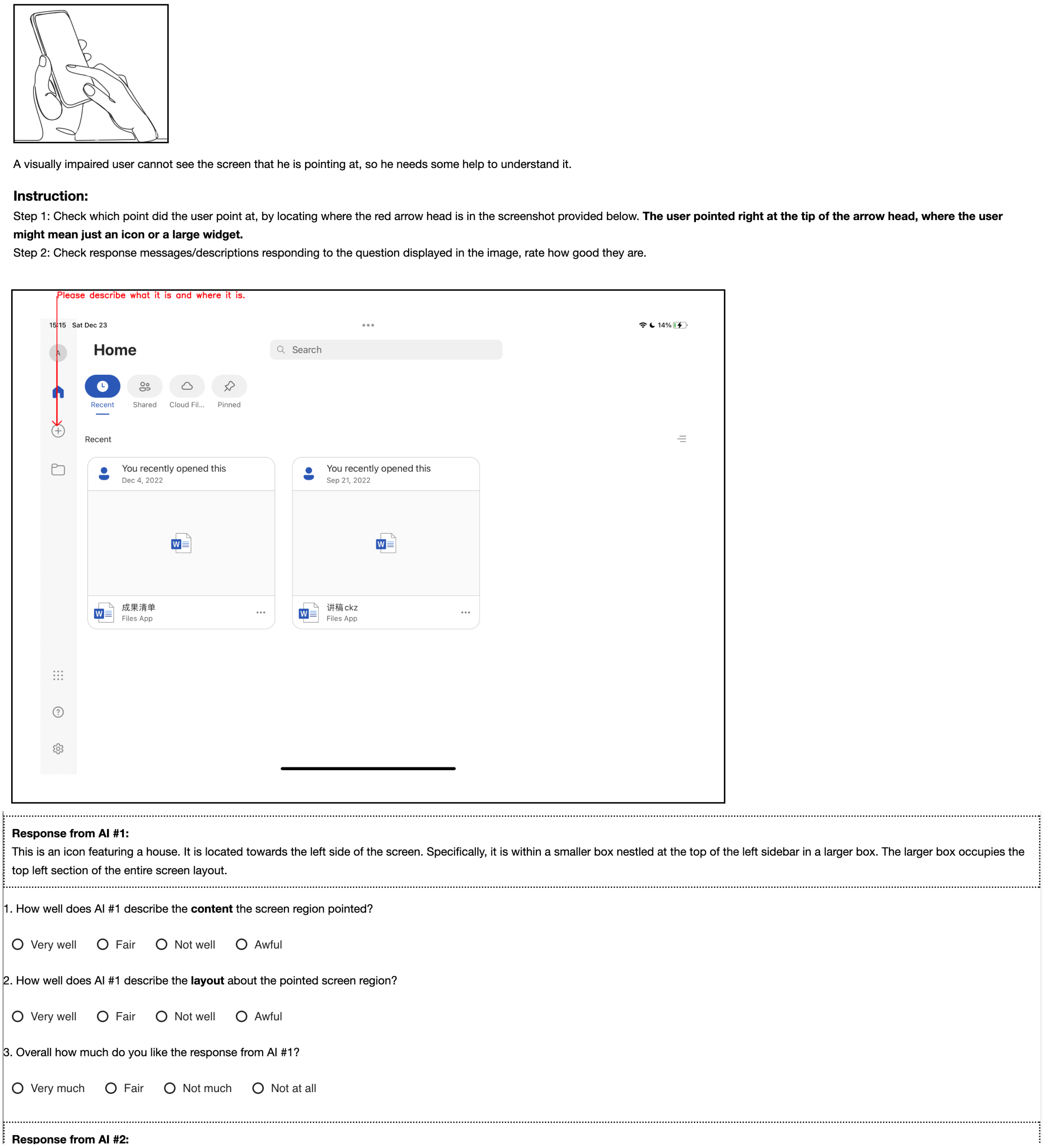
(i) Web page

(ii) Windows GUI

(iii) Ubuntu GUI

(iv) Unseen Mobile GUI

Figure 17: Inference results on Web, OS, and unseen Mobile GUI



A visually impaired user cannot see the screen that he is pointing at, so he needs some help to understand it.

Instruction:

Step 1: Check which point did the user point at, by locating where the red arrow head is in the screenshot provided below. **The user pointed right at the tip of the arrow head, where the user might mean just an icon or a large widget.**

Step 2: Check response messages/descriptions responding to the question displayed in the image, rate how good they are.

Response from AI #1:

This is an icon featuring a house. It is located towards the left side of the screen. Specifically, it is within a smaller box nestled at the top of the left sidebar in a larger box. The larger box occupies the top left section of the entire screen layout.

1. How well does AI #1 describe the **content** the screen region pointed?

○ Very well ○ Fair ○ Not well ○ Awful

2. How well does AI #1 describe the **layout** about the pointed screen region?

○ Very well ○ Fair ○ Not well ○ Awful

3. Overall how much do you like the response from AI #1?

○ Very much ○ Fair ○ Not much ○ Not at all

Response from AI #2:

Figure 18: Screenshot of the survey we used in the human evaluation based on our ScreenPR benchmark. Due to the limited page length, questions are not shown completely, but the rest of the questions follow the same format as the one shown.