

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

Anonymous ACL submission

Abstract

Graphical User Interfaces (GUIs) are central to our interaction with digital devices. Recently, 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 (SPR) task. 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 SPR 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 SPR 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 will be released.

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

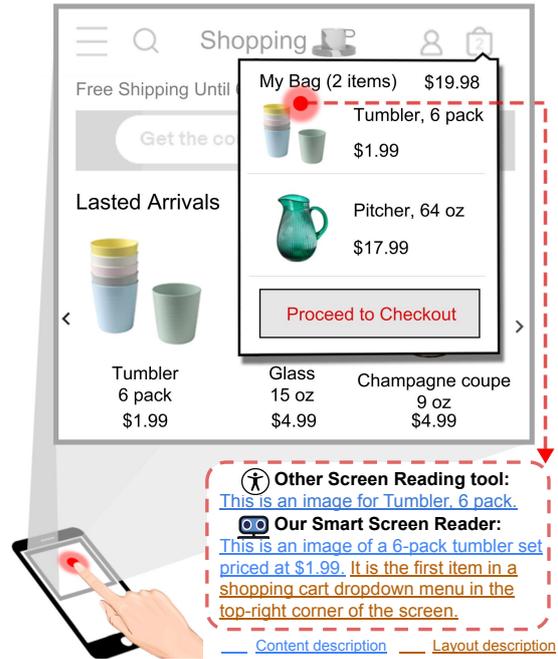


Figure 1: Our ToL agent describes the region on the screenshot indicated by a user’s point. Distinguished from other screen reading tools, our ToL agent can output layout-aware descriptions for points anywhere on the screen.

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 and 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 (SPR) task.

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

053	Language Models (MLLMs), the ToL agent is built	strengthening the performance of mobile agents	105
054	for GUIs screenshots across domains with a point	with sophisticated, layout-aware GUI understand-	106
055	anywhere on the screen as input. The output from	ing capabilities.	107
056	the ToL agent is natural language descriptions for	In conclusion, the contributions of our paper are	108
057	both the content of the indicated region and the re-	as follows:	109
058	lated information about the screen layout, as shown		
059	in Figure 1. Especially, we find that it is critical	• We develop the Tree-of-Lens (ToL) ground-	110
060	to articulate the layout between elements in the	ing method and build the ToL agent that gener-	111
061	screenshot for users to gain a comprehensive under-	ates descriptions that effectively combine both	112
062	standing of the interface and avoid ambiguity. For	content and layout information of the pointed	113
063	example, with only content description, one can-	screen regions.	114
064	not distinguish the two identical “Tumbler pack”		
065	shown in Figure 1.	• We propose the Hierarchical Layout Tree to	115
066	Our ToL agent employs a ToL grounding mech-	represent the underlying hierarchical structure	116
067	anism, constructing a Hierarchical Layout Tree for	of the screenshot. We collect the Android	117
068	each input screenshot to depict its underlying struc-	Screen Hierarchical Layout dataset and train a	118
069	ture. The nodes in the tree represent regions of	model to build the Hierarchical Layout Tree.	119
070	varying scales. We build this tree using an object		
071	detection model trained on our newly collected An-	• We introduce the Screen Point-and-Read	120
072	droid Screen Hierarchical Layout (ASHL) dataset,	benchmark, which specifically challenges the	121
073	which contains 50k bounding boxes of hierarchi-	model to produce accurate descriptions that	122
074	cal screen regions from Android screenshots. This	include both content and layout details.	123
075	model enables an automatic extraction of local and		
076	global regions, forming the Hierarchical Layout	• We rigorously test the ToL agent on the Screen	124
077	Tree. After constructing the tree, we extract the tar-	Point-and-Read benchmark and we demon-	125
078	get path based on the region of interest, producing	strate its ability to accurately identify incor-	126
079	lenses as visual prompts for the MLLM, simulating	rect actions in mobile navigation trajectories.	127
080	a human-like, gradually refined focus.		
081	To rigorously evaluate our ToL agent against	2 Related Works	128
082	other baselines, we introduce the Screen Point-and-	MLLMs for GUI Understanding Recent ad-	129
083	Read (SPR) benchmark with the SPR task that re-	vancements in Multimodal Large Language Models	130
084	quires the model to generate descriptions based on	(MLLMs) have led to arising a series of highly ca-	131
085	user-indicated points. This benchmark consists of	capable generalist models (Liu et al., 2024b; Chen	132
086	650 screenshots from web, mobile, and operating	et al., 2023; Ye et al., 2023; Dai et al., 2023;	133
087	system GUIs, manually annotated with 1,500 target	Bai et al., 2023; Liu et al., 2024a). Spurred by	134
088	points and regions. With the SPR benchmark, we	the strong general capability of MLLMs, recent	135
089	show that our ToL agent achieves the best perfor-	works have been focused on utilizing and improv-	136
090	mance compared with the baselines of other general	ing MLLMs for GUI tasks. Some focus mainly on	137
091	and GUI-specialized MLLMs, with over 15% and	a specific GUI task such as GUI navigation (Wang	138
092	30% improvements respectively in terms of the ac-	et al., 2024; Yang et al., 2023b; Zheng et al., 2024),	139
093	curacy of content and layout descriptions compared	GUI referring expression comprehension (Cheng	140
094	with vanilla text-prompting GPT-4o.	et al., 2024), while some aim at a set of GUI ground-	141
095	In addition, we also test our ToL agent with tra-	ing and referring tasks simultaneously, such as Co-	142
096	jectories from a mobile GUI navigation agent. By	gAgent (Hong et al., 2023), which is Multi-task	143
097	applying ToL agent to describe the point of each	Fine-tuned on CogVLM (Wang et al., 2023), and	144
098	action taken in the trajectory, we demonstrate its	Ferret-UI (You et al., 2024), ensuing Ferret (You	145
099	utility in identifying incorrect actions along the ex-	et al., 2023; Zhang et al., 2024), tailored for GUI	146
100	ecution path. This capability is pivotal for refining	domain. In our work, we focus on the Screen Point-	147
101	the development of mobile agents, as it provides	and-Read (SPR) task, a referring task in which	148
102	clear, actionable feedback on navigation decisions.	inputs are point locations on the screen with the	149
103	This application not only validates the ToL agent’s	corresponding screenshots, and develop a specific	150
104	effectiveness but also opens up possibilities for	benchmark to evaluate the model’s performance in	151

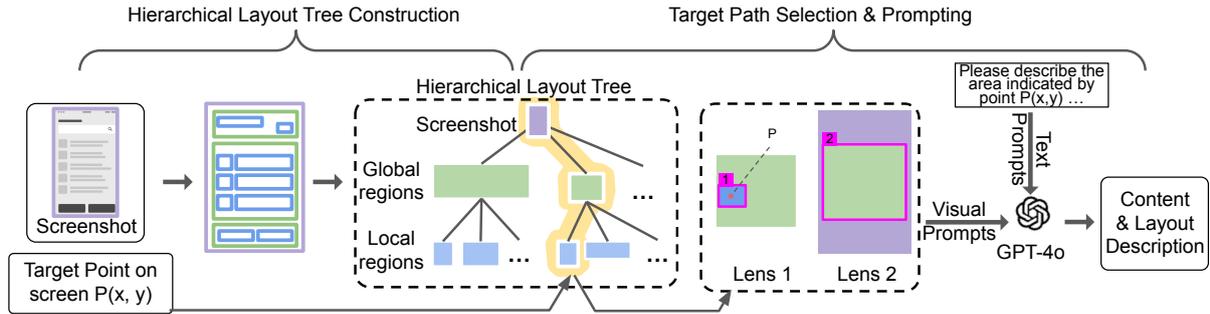


Figure 2: Pipeline of the Tree-of-Lens agent. The Hierarchical Layout Tree is first constructed based on detected global and local regions from the input screenshot. Then, a set of hierarchical lenses with various field widths is generated from the tree and sent as visual prompts to GPT-4o to generate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

152 such tasks. Also, inspired by big challenges that
 153 MLLMs are facing in understanding images with
 154 complex layout information (Fan et al., 2024), we
 155 propose the Tree-of-Lens (ToL) agent that gener-
 156 ates layout-aware description, which distinguishes
 157 from all other GUI agents with GUI referring abil-
 158 ity.

159 **Chain-of-Thoughts for Vision** The Chain of
 160 Thoughts (CoT) (Wei et al., 2022) methodology in
 161 natural language processing (NLP) has spurred vari-
 162 ous inspiring ideas in this direction. (Creswell et al.,
 163 2022) applied it to interpretable logical reasoning,
 164 (Zhou et al., 2022) improved methods for break-
 165 ing down complex problems, (Wang et al., 2022)
 166 focused on enhancing the "exploring-selecting"
 167 mechanism in reasoning path, and (Yao et al., 2024;
 168 Besta et al., 2024) brought in more sophisticated
 169 patterns for reasoning. To embrace CoT in the
 170 vision area, Xi et al. (2023) uses Chain-of-Look
 171 prompting for surgical scene understanding to learn
 172 rich semantics from surgical endoscopic videos. In
 173 comparison, Chen et al. (2024) designated an it-
 174 erative step-by-step reasoning manner for VQA
 175 problem, letting MLLMs "See-Think-Confirm" to
 176 generate a better rationale for the decided answer.
 177 Recently, Shao et al. (2024) proposed a multi-turn
 178 processing pipeline that dynamically focuses on
 179 visual inputs and provides interpretable thoughts.
 180 Moving along with the direction, we aim to facil-
 181 itate MLLMs on the GUI referring task by gener-
 182 ating a chain of lenses with varying field widths,
 183 ranging from fine-grained to coarse views. The
 184 chain of lenses demonstrates the hierarchical lay-
 185 out of the GUI, simulating a human-like sequence
 186 of attention, improving MLLMs on producing ro-
 187 bust and accurate descriptions with the content and
 188 layout information regarding the target point input

in the accessible screen reading task.

3 Tree-of-Lens Agent

3.1 Task Definition

189 We aim to develop an agent to interpret design-
 190 ated regions on GUIs indicated by points, which we
 191 refer to as the Screen Point-and-Read (SPR) task.
 192 This agent needs to process input consisting of
 193 a screenshot S_i and a point coordinate P_i on the
 194 screen, where the point represents a region of the
 195 user’s interest. Two target outputs are language de-
 196 scriptions \hat{D}_i^c for the content of the targeted region
 197 and \hat{D}_i^l for the layout information of the screen
 198 related to the targeted region.
 199

3.2 Method Overview

200 Targeting the SPR task, we propose the Tree-of-
 201 Lens (ToL) agent equipped with the ToL grounding
 202 mechanism, as shown in Figure 2. Since our agent
 203 relies on the pure visual modality of the GUI and
 204 leverages the strong visual referring capabilities of
 205 MLLMs, such as GPT-4o, it can effectively read
 206 anywhere pointed on screens from any domain,
 207 including but not limited to web, mobile, and oper-
 208 ating systems.
 209

210 The three distinct design goals distinguish our
 211 LoT agent from previous models. First, our LoT
 212 agent outputs rich, natural language descriptions
 213 rather than fixed template descriptions from acces-
 214 sible screen reading tools such as VoiceOver¹. Sec-
 215 ond, it can handle any point location on the screen
 216 as input by dynamically detecting the pointed re-
 217 gion, which might be as small as an icon or a
 218 broader area, such as a full-screen window. Last
 219 but not least, the descriptions generated by our
 220 agent encompass both the content and layout per-
 221

¹ <https://support.apple.com/guide/voiceover/welcome/mac>

spectives, 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 $\hat{R}_j, j = \text{ArgMax}_j \text{IoU}(R_i, R_j)$.

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

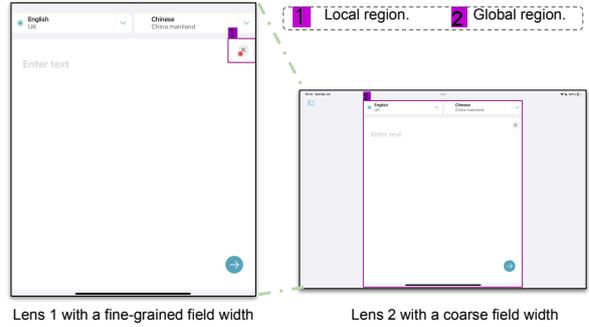


Figure 3: Example of the lenses generated from the Hierarchical Layout Tree based on a point coordinate. Lens 2 can be seen as a zooming-out from Lens 1. ToL agent’s corresponding output is in Appendix A.

bounding boxes from 2,180 screenshots, spanning all 18 different Android applications.

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 $\text{Parent-Child}(R_i, R_j)_{\text{establish}} = 1$ 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

$$R_{i+j} = \text{Max}(R_i, R_j),$$

$$\text{if } \text{IoU}(R_i, R_j) > 0.9$$

$$\& \text{Parent-Child}(R_i, R_j) == 1. \quad (1)$$

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:

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

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

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

3.4 Target Path Selection and Multi-lens 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-lens 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 Tree². 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 C.

4 Screen Point-and-Read Benchmark

4.1 Overview

We introduce the Screen Point-and-Read (SPR) benchmark to rigorously evaluate our ToL agent on the SPR 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 annotated with an average of around 2 target points and their corresponding local regions on the screenshot. Figure 4i illustrates the normalized coordinates of target points P_i in the SPR benchmark, demonstrating thorough and comprehensive cover-

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.

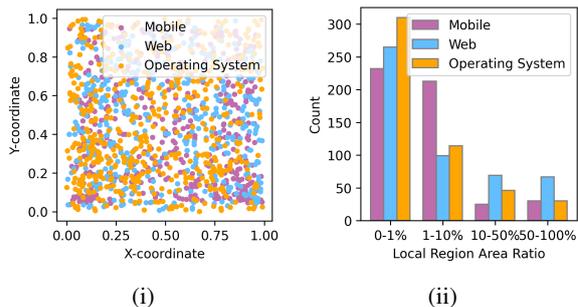


Figure 4: (i) shows the normalized locations of target points P_i in the input screenshot S_i . (ii) shows distributions of the area of local regions R_i in our SPR benchmark.

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 SPR Benchmark

age 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.

4.2 Data Collection

We first gather mobile and web screenshots from Screenshot (Cheng et al., 2024) and operating system screenshot from the data explorer videos³ of OSWorld (Xie et al., 2024). Then we randomly generate 3 or 4 candidates of the target point P_i for each screenshot. Then, to maximize the variety of the region indicated by points, we hire 3 students to check these candidate points, where they assess all the candidate points on each screenshot and remove those that represent duplicated regions. Then, the remaining points are regarded as the target point P_i , and the annotators draw the bounding boxes of the corresponding local regions R_i . After the annotation, scripts were used to spot errors, such as ensuring that local regions R_i included P_i inside. The annotated data is then manually examined by authors to ensure quality.

4.3 Evaluation

The SPR 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

³ <https://os-world.github.io/explorer.html>

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 had 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-4o 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

$$\text{Multi-choice question} = Q \quad (4)$$

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

where Q is the fixed question and $Crop(S_i, R_i)$ 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-turbo. 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 point, is 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.⁴ The multi-choice selection task presented to the auxiliary model is

$$\text{Multi-choice question} = Q_{\text{gen}}(\hat{D}^l_i, \hat{D}^l_{i'}) \quad (6)$$

$$\text{Multi-choice candidates} = C, \quad (7)$$

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. 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 C.

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, 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 SPR 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

⁴ There are in total 1,117 target points P_i are accompanied by a manually selected reference point $P_{i'}$, due to R_i is taking most area of the full screen.

	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
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 SPR 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
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	65.6 (-9.4)	37.6 (-2.1)
+ w/o local & global regions marks	35.1 (-29.9)	21.9 (-18.6)

Table 3: Ablation results. We gradually refine the ablation to exclude various processes in our ToL grounding mechanism for ToL agent on the SPR benchmark. The results show the effectiveness of various components.

	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 4: 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.

469 model. Furthermore, the overall scores from the hu-
470 man evaluation indicate that descriptions generated
471 by our ToL agent are the most favorable. Interest-
472 ingly, although the results show that the CogAgent
473 performs slightly worse than LlaVA-NeXT in terms
474 of the content and layout description accuracy, the
475 overall scores from human judges slightly prefer
476 LlaVA-NeXT. We believe this preference is due
477 to the LlaVA-NeXT model’s tendency to produce
478 longer descriptions with more information about
479 the screen, creating a more comprehensive feel.

480 Additionally, we leverage language similarity
481 scores, including BERTScore and ROUGE-L, to
482 compare the generated content descriptions with
483 human-verified descriptions. These descriptions
484 are captions generated using GPT-4V with the di-
485 rect input of $Crop(S_i, R_i)$, and we manually ex-
486 amined and corrected mistakes. We observe that
487 the overall trend from the language similarity eval-
488 uation results matches the content evaluation re-
489 sults from human and cycle consistency evalua-
490 tions. However, the differences in language simi-
491 larity scores between models are relatively indis-
492 tinguishable. We believe this is because the cap-
493 tions are more comprehensive, sometimes redun-
494 dant, whereas the screen reading output is more
495 focused on delivering the key features of the con-
496 tent to help users understand it.

5.2 Ablation Study 497

Effectiveness analysis 498 To assess the effective-
499 ness of the ToL design, we conduct an ablation
500 study where we progressively refine the ablation
501 to exclude key design components and evaluate
502 these ablated versions on our SPR benchmark us-
503 ing the cycle consistency evaluation, as shown in
504 Table 3. First, we modify our ToL agent to ex-
505 clude the multi-lens visual prompting. This change
506 necessitates annotating bounding boxes for the lo-
507 cal region, global region, and the target point on
508 the complete screenshot as a single visual prompt.
509 Then we further remove the red dot mark from
510 the visual prompt, resulting in a design similar to
511 the Set-of-Mark (SoM) prompting method (Yang
512 et al., 2023a). As a result, these changes lead in per-
513 formance degradation in both content and layout
514 description accuracy, indicating the effectiveness
515 of our visual prompting strategies. Finally, we ab-
516 late the input visual prompt to remove marks of
517 local and global regions that we extract from our
518 Hierarchical Layout Tree. This drastic modifica-
519 tion results in a significant decrease in performance,
520 underscoring the importance of our Hierarchical
521 Layout Tree in the of the ToL design.

Bottleneck analysis 522 To examine the bottlenecks
523 in our ToL agent, we replace the local and global re-
524 gions generated from the Hierarchical Layout Tree

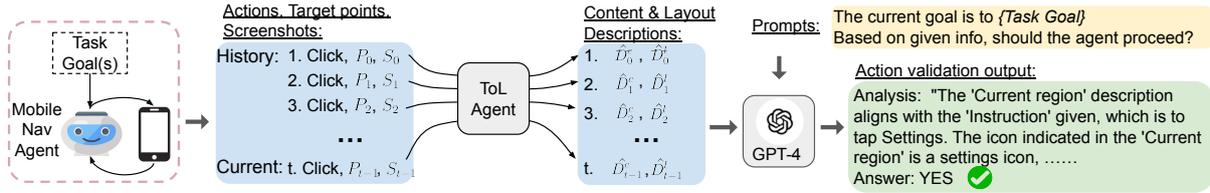


Figure 5: Pipeline of employing our ToL agent in verifying the actions from a mobile navigation agent.

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 SPR benchmark. As shown in Table 4, 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 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 D.1. To incorporate our ToL agent with the verification purpose, 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 and sent to the GPT-4-Turbo model. The GPT-4-Turbo model, based on the task goal, instructions, and descriptions from ToL agent, decides whether the last action from the MagicWonder is correct or not.

Data sampling We select 52 mobile agent trajectories with specific task goals from instruction-guided executions on MagicWand platform (Ding et al., with more details in Appendix D.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

	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 5: 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

incorrect actions in the chosen trajectories to get the ground truth labels.

Result We perform 10 end-to-end evaluations and present the mean value and standard deviation in Table 5. Despite minor fluctuations, while 92% correct actions can be stably recognized, about 38% incorrect actions can be identified. This reveals that incorporating the description from the ToL agent can significantly and robustly improve mobile navigation agents’ capability. 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 fall into this category, 44% ± 0.04 can be successfully detected.

6 Conclusion and Discussion

In this work, we propose a novel Tree-of-Lens (ToL) grounding method and build a ToL agent for the Screen Point-and-Read (SPR) task, an important screen referring task. Our ToL agent outputs layout-aware descriptions for pointed regions on screenshots. With our newly proposed Screen Point-and-Read (SPR) benchmark, we show that our ToL agent outperforms all other baselines in terms of content and layout description accuracy, demonstrating great potential to improve accessibility of digital devices for visually impaired users. We also apply the ToL agent to identify incorrect actions in the execution trajectories of a mobile navigation agent, achieving promising results.

7 Limitations

In this work, we propose a novel Tree-of-Lens (ToL) agent for the Screen Point-and-Read (SPR) task. Leveraging GPT4-o to generate content and layout descriptions allows us to set up the pipeline quickly and validate ToL’s effectiveness 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 leverages 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

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond.

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. Minigt-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 mm-lab 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. Seeclck: 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*.

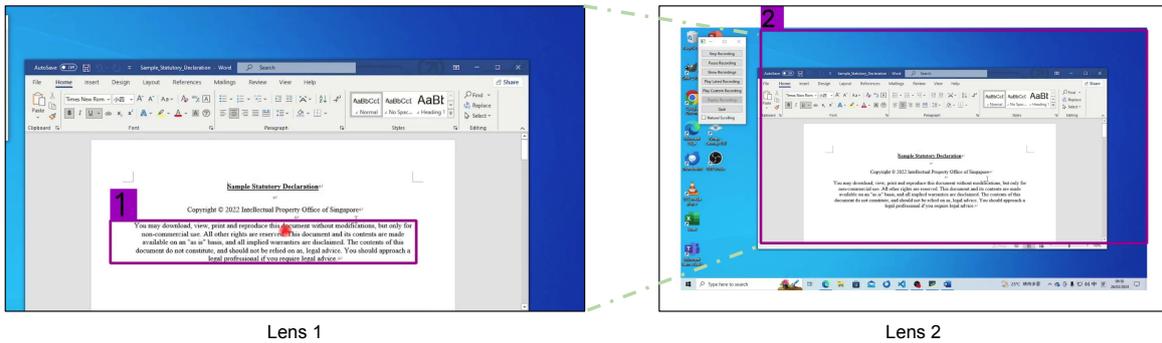
Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024a. *Llava-next: Improved reasoning, ocr, and world knowledge*.

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*.

703	Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Span-	760
704	dana Gella, Robinson Piramuthu, Gokhan Tur, and	761
705	Dilek Hakkani-Tur. 2022. Teach: Task-driven em-	762
706	bodied agents that chat. In <i>Proceedings of the AAAI</i>	763
707	<i>Conference on Artificial Intelligence</i> , volume 36,	
708	pages 2017–2025.	
709		
710	Hao Shao, Shengju Qian, Han Xiao, Guanglu Song,	
711	Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng	
712	Li. 2024. Visual cot: Unleashing chain-of-thought	
713	reasoning in multi-modal language models. <i>arXiv</i>	
714	<i>preprint arXiv:2403.16999</i> .	
715	Peter Shaw, Mandar Joshi, James Cohan, Jonathan Ber-	
716	rant, Panupong Pasupat, Hexiang Hu, Urvashi Khan-	
717	delwal, Kenton Lee, and Kristina N Toutanova. 2023.	
718	From pixels to ui actions: Learning to follow in-	
719	structions via graphical user interfaces. <i>Advances in</i>	
720	<i>Neural Information Processing Systems</i> , 36:34354–	
721	34370.	
722	Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan,	
723	Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang.	
724	2024. Mobile-agent: Autonomous multi-modal mo-	
725	bile device agent with visual perception. <i>arXiv</i>	
726	<i>preprint arXiv:2401.16158</i> .	
727	Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi	
728	Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei	
729	Zhao, Xixuan Song, et al. 2023. Cogvlm: Visual ex-	
730	pert for pretrained language models. <i>arXiv preprint</i>	
731	<i>arXiv:2311.03079</i> .	
732	Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le,	
733	Ed H Chi, Sharan Narang, Aakanksha Chowdhery,	
734	and Denny Zhou. 2022. Self-consistency improves	
735	chain of thought reasoning in language models. In	
736	<i>The Eleventh International Conference on Learning</i>	
737	<i>Representations</i> .	
738	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten	
739	Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou,	
740	et al. 2022. Chain-of-thought prompting elicits rea-	
741	soning in large language models. <i>Advances in neural</i>	
742	<i>information processing systems</i> , 35:24824–24837.	
743	Nan Xi, Jingjing Meng, and Junsong Yuan. 2023. Chain-	
744	of-look prompting for verb-centric surgical triplet	
745	recognition in endoscopic videos. In <i>Proceedings of</i>	
746	<i>the 31st ACM International Conference on Multime-</i>	
747	<i>dia</i> , pages 5007–5016.	
748	Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan	
749	Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhou-	
750	jun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu,	
751	Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caim-	
752	ing Xiong, Victor Zhong, and Tao Yu. 2024. Os-	
753	world: Benchmarking multimodal agents for open-	
754	ended tasks in real computer environments. <i>Preprint,</i>	
755	<i>arXiv:2404.07972</i> .	
756	Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chun-	
757	yuan Li, and Jianfeng Gao. 2023a. Set-of-mark	
758	prompting unleashes extraordinary visual grounding	
759	in gpt-4v. <i>arXiv preprint arXiv:2310.11441</i> .	
	Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Ze-	760
	biao Huang, Bin Fu, and Gang Yu. 2023b. Appa-	761
	gent: Multimodal agents as smartphone users. <i>arXiv</i>	762
	<i>preprint arXiv:2312.13771</i> .	763
	Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran,	764
	Tom Griffiths, Yuan Cao, and Karthik Narasimhan.	765
	2024. Tree of thoughts: Deliberate problem solving	766
	with large language models. <i>Advances in Neural</i>	767
	<i>Information Processing Systems</i> , 36.	768
	Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye,	769
	Ming Yan, Yiyang Zhou, Junyang Wang, An-	770
	wen Hu, Pengcheng Shi, Yaya Shi, et al. 2023.	771
	mplug-owl: Modularization empowers large lan-	772
	guage models with multimodality. <i>arXiv preprint</i>	773
	<i>arXiv:2304.14178</i> .	774
	Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du,	775
	Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-	776
	Fu Chang, and Yinfei Yang. 2023. Ferret: Refer	777
	and ground anything anywhere at any granularity. In	778
	<i>The Twelfth International Conference on Learning</i>	779
	<i>Representations</i> .	780
	Keen You, Haotian Zhang, Eldon Schoop, Floris Weers,	781
	Amanda Swearngin, Jeffrey Nichols, Yinfei Yang,	782
	and Zhe Gan. 2024. Ferret-ui: Grounded mobile ui	783
	understanding with multimodal llms. <i>arXiv preprint</i>	784
	<i>arXiv:2404.05719</i> .	785
	Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su,	786
	Jun Zhu, Lionel Ni, and Heung-Yeung Shum. 2022.	787
	Dino: Detr with improved denoising anchor boxes	788
	for end-to-end object detection. In <i>The Eleventh In-</i>	789
	<i>ternational Conference on Learning Representations</i> .	790
	Haotian Zhang, Haoxuan You, Philipp Dufter, Bowen	791
	Zhang, Chen Chen, Hong-You Chen, Tsu-Jui Fu,	792
	William Yang Wang, Shih-Fu Chang, Zhe Gan, et al.	793
	2024. Ferret-v2: An improved baseline for referring	794
	and grounding with large language models. <i>arXiv</i>	795
	<i>preprint arXiv:2404.07973</i> .	796
	Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and	797
	Yu Su. 2024. Gpt-4v(ision) is a generalist web agent,	798
	if grounded. <i>arXiv preprint arXiv:2401.01614</i> .	799
	Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei,	800
	Nathan Scales, Xuezhi Wang, Dale Schuurmans,	801
	Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2022.	802
	Least-to-most prompting enables complex reasoning	803
	in large language models. In <i>The Eleventh Interna-</i>	804
	<i>tional Conference on Learning Representations</i> .	805

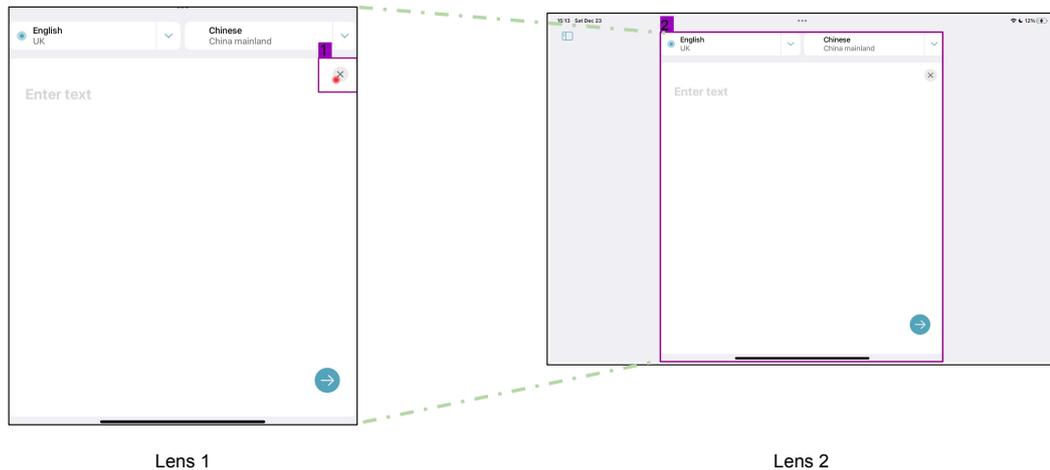
806	A Example outputs of the ToL Agent	D Mobile Navigation Action Verification	849
807	For each GUI domain, we show one example of	D.1 Example	850
808	the output of our ToL agent along with the lenses	Figure 16 illustrates one action verification case in	851
809	generated. We do not deliberately cherry pick the	our experiment: MagicWonder is asked to accom-	852
810	best case, but show examples with both strong and	plish "pause historical track in Youtube App". Len	853
811	weak performances. An example for the operation	1 and Len 2 show the agent was attempting to click	854
812	system domain is shown in Figure 6. An example	the "Setting" icon by giving the instruction "Tap	855
813	for the mobile domain is shown in Figure 7. An	Settings" at the second step along the execution tra-	856
814	example for the web domain is shown in Figure 8.	jectory. The first step in action varification is to get	857
815		the response from ToL Agent that describes Len 1	858
816	B Examples of the Cycle Consistency	and Len 2 correctly with more detailed application	859
	Evaluation	contexts. Using this information, GPT-4o could	860
817	We select one result of our cycle consistency evalu-	correctly judge whether this action on the screen	861
818	ation for the content description and show it in Fig-	was in accordance with the given instruction.	862
819	ure 9. The left four images come from the cropped	D.2 Sample Strategy of Mobile Navigation	863
820	areas from different GUI screens, with only one	Action Execution Trajectory	864
821	of which matches the description in the prompt on	MagicWand simulator can launch mobile agents	865
822	the right side. As shown, GPT4-o can successfully	to take action on devices under the guidance of a	866
823	choose the correct option in the "Answer" field and	series of instructions. The corresponding runtime	867
824	give its reasoning in the "Analysis" field.	GUI context, like GUI screenshot, actionable re-	868
825	For cycle consistency evaluation regarding the	gions, instructions for each step and action details,	869
826	layout description, the question contains the gen-	are saved into a self-descriptive JSON format. We	870
827	erated layout descriptions from models. We let	deliberately choose the instructions from WeWeb	871
828	GPT-4o choose one option from predefined 9 rela-	dataset (Ding et al.) having few overlapping with	872
829	tive positions together with its analysis, referring	our ASHL dataset. After execution on MagicWand,	873
830	to the layout evaluation output in Figure 11. The	we chose those trajectories with task completion	874
831	ground truth is provided with script based on the	rate less than 50% to evaluate whether ToL infor-	875
832	human annotated local regions for the target point	mation was accurate enough. The selected trajec-	876
833	and reference point. The ground truth is also checked	tories cover 11 Android applications from 8 ap-	877
834	by human.	plication domains, where the chosen applications	878
835	Prompt used by LLaVA-NEXT and GPT-4o For	include SYSTEM, Entertainment, Education, Food	879
836	the baseline models, LLaVA-NEXT and GPT-4o	& Drinks, Communication, Shopping, News, Maps	880
837	share the same prompting template as shown in	& Navigation. The chosen Android apps include	881
838	Figure 14.	Messenger, Google Maps, YouTube, Settings, Qui-	882
839	Prompt used by CogAgent In order to drive Co-	zlet, FlipBoard, McDonalds, eBay, Google Chat,	883
840	gAgent to follow instructions, a simplified prompt-	Here WeGo, DoorDash.	884
841	ing template is applied, as shown in Figure 15.	D.3 Action verification Pipeline Details	885
842	C Text Prompt templates	As shown in Figure 5, a set of navigation actions,	886
843	We show the templates of text prompts we used for	target points, and corresponding screenshots, are	887
844	different MLLMs. We substitute the $\{\{x\}\},\{\{y\}\}$	first recorded during the interaction between the	888
845	with the normalized coordinate of the input point.	mobile navigation agent and the mobile screen.	889
846	Prompt used by ToL agent For our ToL agent,	Then, our ToL agent first processes each pair of	890
847	we use the text prompt template shown in Figure	target points and screenshots (P_i, S_i) , including	891
848	13 for GPT-4o.	those from the history path of navigation trajectory	892
		and the currently planned one $(i \in [0, t-1])$, where	893
		the $P_t - 1$ is the target point that the navigation	894
		agent plans to interact with for the next step. As a	895
		result, the layout and content descriptions \hat{D}_i^c, \hat{D}_i^l	896
		are generated. Next, we concatenate all the descrip-	897



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.

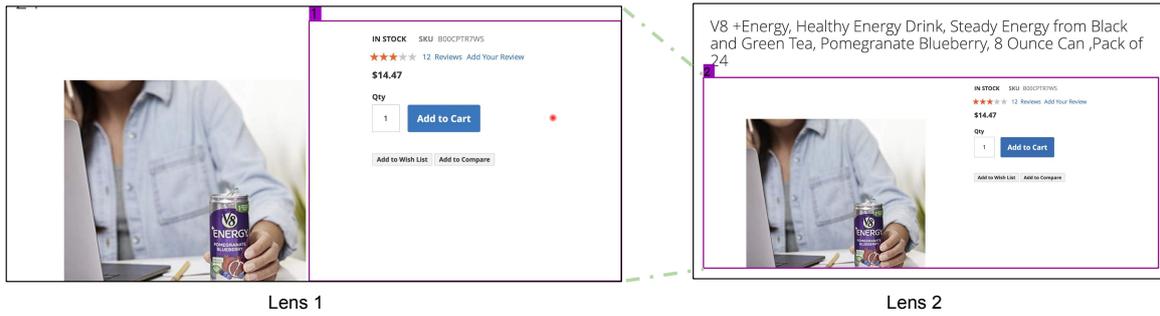
Figure 6: Example of the generated description for an operation system screenshot from our ToL agent with the two lenses generated. The global region generated is not perfect as the bounding box is a little bigger than the application window on the top right part.



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.



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.

tions and send them to the GPT-4-Turbo model 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.

Action verification We prompt GPT-4o by using 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, the instruction for 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.

E Human Evaluation

We leverage human evaluation to test different screen reading models on our SPR benchmark. We leverage Amazon Mechanical Turk⁵ platform to find annotators to complete the survey that we designed for our human evaluation. We pay \$0.15 for each survey and the screenshot of the survey is shown in Figure 18

⁵ <https://www.mturk.com>

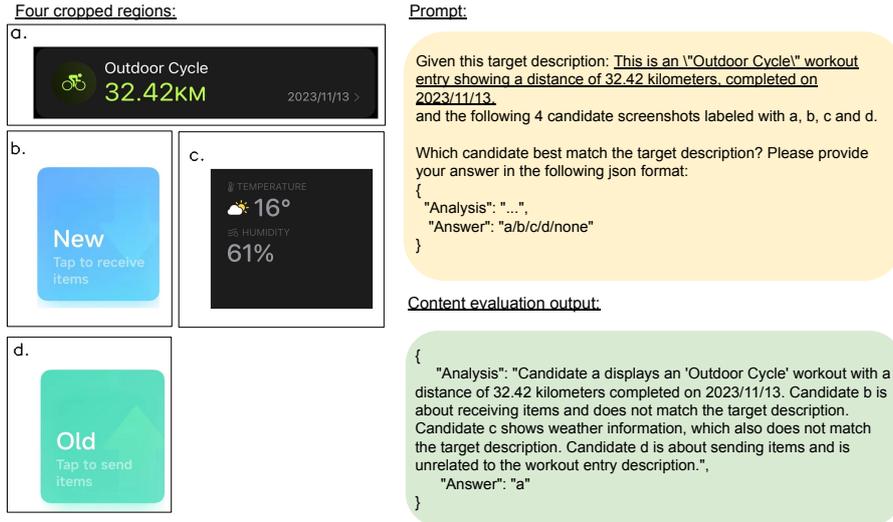


Figure 9: One example of our cycle consistency evaluation for content description. A predicted description is shown on the right and four different cropped screen regions were sent as four separate images to GPT-4o. GPT-4o choose the matched region or answer with "unknown".

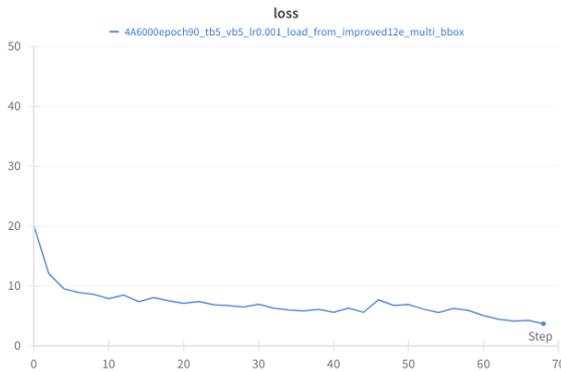


Figure 10: Training Loss of ResNet-50 backbone with epoch 90.

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 6: Average precision on the validation set⁶ for GUI region detection model.

F 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 $\gamma = 0.1$, 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 validation set are shown in Table 6.

Inference Examples and Evaluation Results After training, we utilize the GUI region detection model on unseen OS, Web, and mobile screenshots in our SPR 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 SPR 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 7. 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.

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.

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 12: Prompt used by GPT-4o in Action Validation

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 in format (1) ... (2) ...

Figure 13: 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 in format (1) ... (2) ...

Figure 14: Prompt used by LlaVA-NEXT and GPT-4o

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 15: Prompt used by CogAgent

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 7: 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.

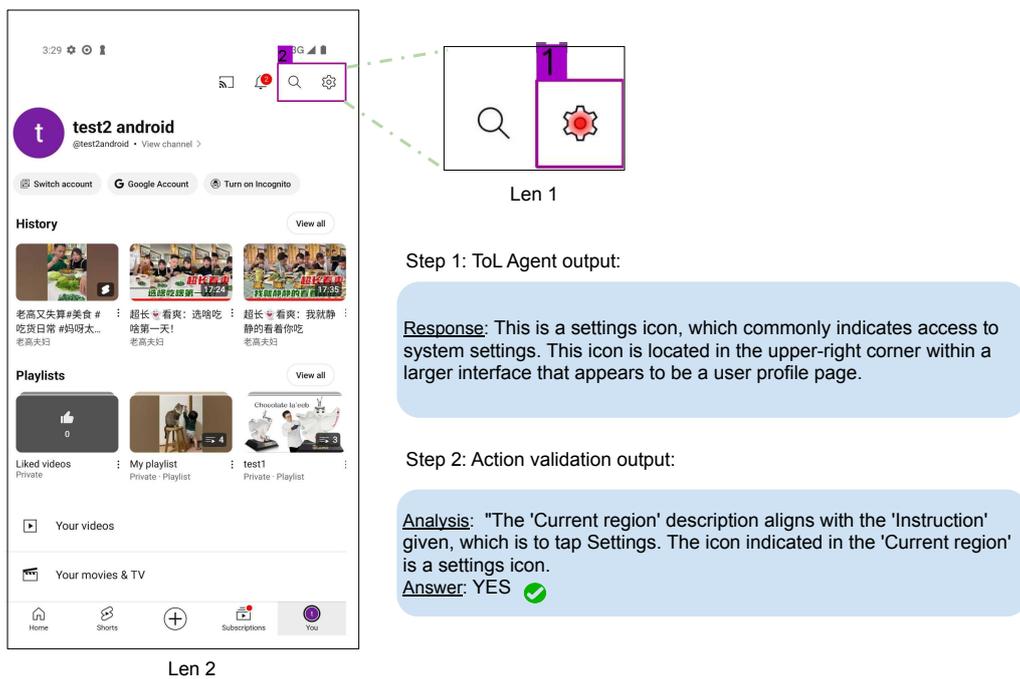
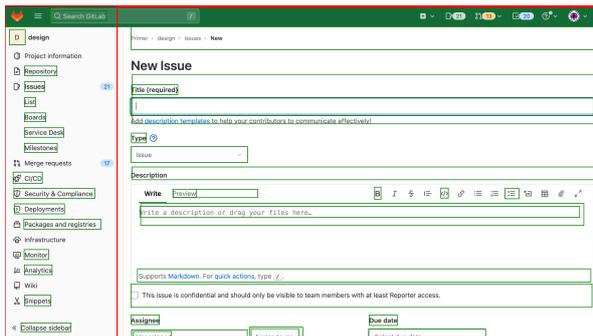
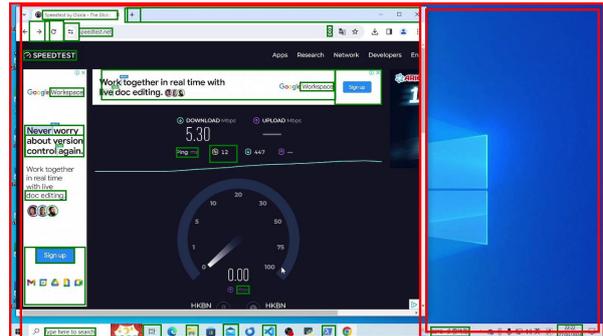


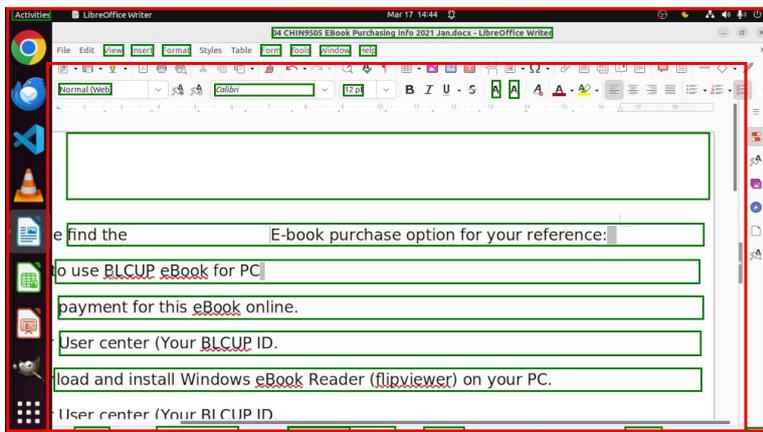
Figure 16: Given the goal of "pausing historical track in Youtube App", mobile Agent on MagicWand plans to click on the screen. ToL Agent generates descriptions for the action planned. The description is sent as input for action verification reasoning, where GPT-4o model output the analysis with decision,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



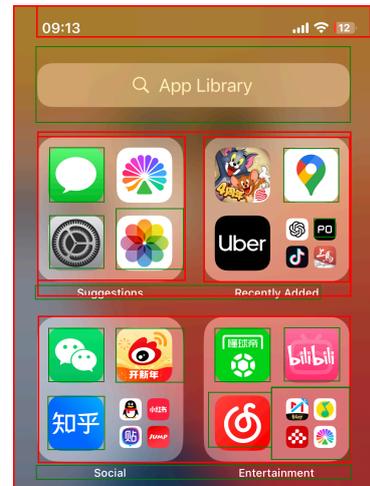
(i) Web page



(ii) Windows GUI

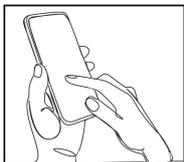


(iii) Ubuntu GUI



(iv) Unseen Mobile GUI

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

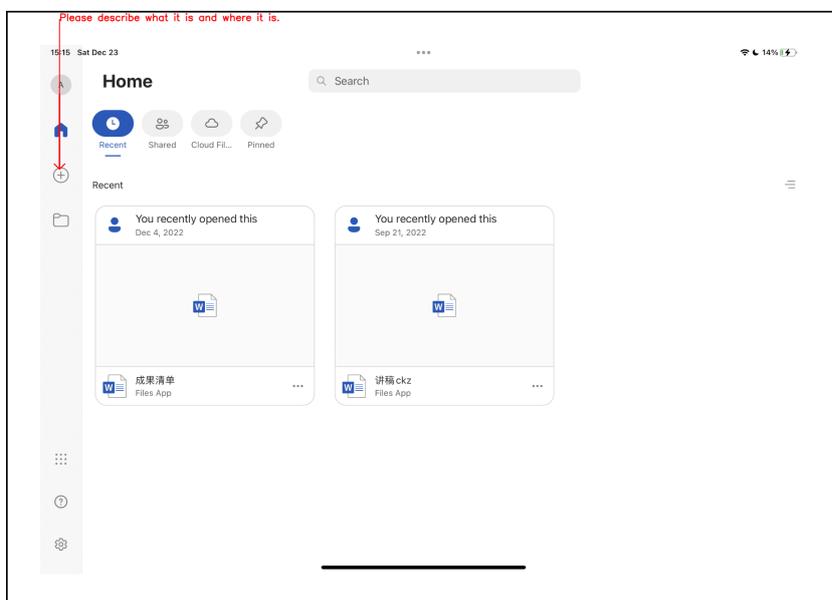


A visually impaired user cannot see the screen that he is pointing at, so he needs some help to understand it.

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 SPR benchmark. Due to the limited page length, questions are not shown completely, but the rest questions follow the same format as the shown one.