Evaluating and Easing Hallucinations for GUI Grounding

Zicheng Zhang*[1], Hongyi Jing*[2], Rui Lv[2], Shuo Fang[2], Shiai Zhu[2], Junying Wang[4],
Chunyi Li[1,3], Xiaohong Liu[3], Chenguang Ma†[,2], and Guangtao Zhai†[,1,3]
Shanghai AI Lab[1], Ant Group[2], Shanghai Jiao Tong University[3], Fudan University[4]
Equal Contributions*, Corresponding Authors†

Abstract

Existing GUI benchmarks primarily focus on evaluating models' comprehensive capabilities but largely overlook hallucination phenomena in grounding tasks, which are crucial to the reliability of GUI understanding. In this work, we expose two major types of hallucinations in GUI grounding: 1) ***Confusion Hallucination****, where distractor elements are mistakenly selected, and 2)* ***Fabricated Hallucination****, where nonexistent elements are hallucinated with plausible coordinates. To systematically investigate their origins, we introduce GUI-HalluBench, a benchmark comprising two complementary subsets: a parsing subset for assessing structural representation of GUI elements and a hallucination subset for measuring grounding robustness under challenging conditions. To further address these hallucinations, we propose two approaches for alleviating hallucination behaviors: a training-free Parsing-guided Prompt (PGP) approach and a training-based Hallucination-aware Fine-Tuning (HFT) approach. Experiments on state-of-the-art models confirm that deficiencies in parsing lead to both confusion and fabricated hallucinations, while our easing strategies effectively reduce these hallucinations, enhancing grounding reliability. Data available at* https://github.com/aiben-ch/GUI-HalluBench.

1. Introduction

Large multimodal models (LMMs) are increasingly deployed in graphical user interfaces (GUIs), enabling a wide range of applications such as automated software operation, intelligent assistants, and interactive agents [2, 31, 32, 48, 55, 56]. In the context of GUIs, parsing involves identifying and categorizing interface elements (such as buttons, icons, and text fields) based on the visual content of the screen. Grounding, on the other hand, refers to mapping these elements to the corresponding functions or semantics as described by user instructions. To evaluate such emerging capabilities, recent studies have proposed a series of GUI benchmarks [6, 10, 18, 22, 23, 39, 41, 50], covering tasks like understanding, grounding and navigation. These benchmarks emphasize comprehensive competence, aiming to test whether LMMs can perceive interface structures, follow instructions, and accomplish user goals from an all-around perspective. While such evaluations are valuable for measuring overall utility, they pay limited attention to reliability aspects, particularly hallucinations. Within the domain of GUIs, this omission is critical: unlike minor perception errors, hallucinations can mislead models into confidently outputting incorrect or fabricated targets, directly undermining trustworthiness in real interactions.

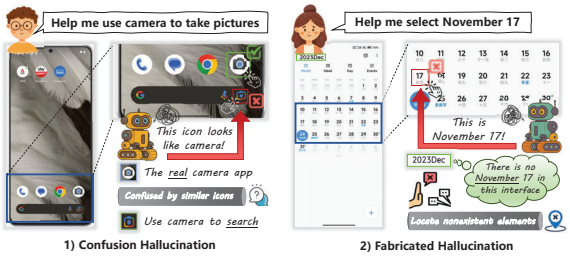

Figure 1. Illustrative cases of two typical hallucination types when LMMs interact with GUIs. (a) *Confusion Hallucination*: the model mistakes a visually similar icon for the camera app. (b) *Fabricated Hallucination*: the model imagines a nonexistent date ("November 17") in the calendar interface.

Grounding, as a core capability in GUI interaction, is particularly susceptible to hallucinations, yet not thoroughly investigated. From our empirical observations and experimental analysis, we find that LMMs tend to exhibit two common types of hallucinations as shown in Fig. 1. The first is **Confusion Hallucination**, where a distractor element is incorrectly chosen due to semantic understanding ambiguity or high visual similarity. The second is **Fabricated Hallucination**, where the model hallucinates a nonexistent element and even produces plausible coordinates for it, despite the absence of any valid target. However, no current benchmarks are specifically designed to analyze these hallucination phenomena, leaving their root

causes and systematic evaluation largely unexplored.

To address this gap, we propose **GUI-HalluBench**, a dedicated benchmark for evaluating LMMs in GUI scenarios and analyzing the correlations between parsing and grounding hallucinations. We observe that grounding hallucinations are not isolated but closely tied to deficiencies in parsing, the ability to correctly interpret interface structures and relationships. Parsing errors often manifest as confusion hallucinations when the model misidentifies similar elements, or as fabricated hallucinations when it generates entirely nonexistent ones. To capture this dependency, our benchmark includes two complementary subsets: a **Parsing** subset to test structural understanding of GUI elements, and a **Hallucination** subset to directly evaluate grounding robustness under challenging conditions. This design enables both independent ability evaluation and systematic analysis of how parsing deficiencies propagate into hallucinations.

In addition to this diagnostic framework, we propose two complementary strategies for easing hallucinations. The first is a training-free **Parsing-guided Prompt (PGP)** approach that guides the model to first parse the interface and then perform grounding with only prompt engineering, ensuring more reliable element identification. The second is a training-based **Hallucination-aware Fine-Tuning (HFT)** approach that improves model robustness against hallucinations by leveraging a specialized training process using hallucination-labeled data. These approaches vary in cost and significantly enhance the ability of LMMs to handle hallucination scenarios and improve overall GUI grounding reliability. Extensive experiments are conducted on state-of-the-art LMMs using GUI-HalluBench, and the results consistently validate our previous point: LMMs that struggle in parsing tasks are more prone to both confusion hallucination and fabricated hallucination. Furthermore, our proposed easing strategies effectively reduce these behaviors. These findings not only highlight the current limitations of LMMs in GUI understanding but also provide actionable insights into the origins of hallucinations and their mitigation. In summary, our contributions are threefold:

- To the best of our knowledge, we present the first systematic study of hallucinations in GUI grounding, categorizing them into two major failure modes: **Confusion Hallucination** & **Fabricated Hallucination**.
- We introduce **GUI-HalluBench**, the first benchmark dedicated to hallucination evaluation in GUI scenarios, comprising both parsing and hallucination subsets.
- We empirically demonstrate how deficiencies in parsing contribute to the propagation of hallucinations and propose effective strategies (ranging from training-free to training-based approaches) designed to mitigate hallucinations. These approaches vary in cost and aim to enhance the reliability and hallucination-resistance of LMMs for more robust GUI interactions.

Table 1. Comparison of GUI-HalluBench with existing common hallucination and GUI benchmarks, where 'Desk.', 'Mob.', 'Hallu.', 'Lang.', 'Num.' stand for 'Desktop', 'Mobile', Hallucination', 'Language', and 'Number' respectively. Prior benchmarks either focus on natural image hallucination or general GUI understanding, lacking systematic hallucination analysis in structured GUI environments. The GUI-HalluBench uniquely introduces hallucination and bilingual coverage for GUI understanding.

Benchmark	Format	Concern	Lang.	Num.
Common Hallucination Benchmarks				
POPE [26]	Natural Image	Object Hallu.	EN	3,000
HallusionBench [15]	Natural Image/Video	Visual Illusion	EN	1,129
MMHal-Bench [35]	Natural Image	Multi-modal Hallu.	EN	96
HalluQA [7]	Natural Image	Adversial Hallu.	CN	450
GUI Benchmarks				
ScreenSpot [6, 44]	Web/Desk./Mob.	GUI Ground.	EN	1,272
ScreenSpot-Pro [22]	Desk.	GUI Ground.	EN	1,581
CAGUI-Grounding [54]	Mobile	GUI Ground.	CN	1,500
MMBench-GUI-L2 [41]	Web/Desk./Mob.	GUI Ground.	EN	3,574
AITZ [50]	Mob.	GUI Navigation	EN	2,504
AndroidControl [23]	Mob.	GUI Navigation	EN	15,283
Mind2Web [10]	Web	GUI Navigation	EN	2,350
OSWorld [1]	Desk.	GUI Navigation	EN	369
GUI-HalluBench (Ours)	Web/Desk./Mob.	GUI Ground. Hallu.	EN+CN	2,000

2. Related Works

2.1. GUI Benchmarks

Recent progress in GUI-related research has produced various benchmarks targeting different aspects of interface understanding and interaction. Early datasets such as Rico [9] and Screen2Words [39] focused on layout parsing and visual–semantic mapping, while later efforts like ScreenSpot [6, 44], ScreenSpot-Pro [22] and MMBench-GUI-L2 [41] emphasized element grounding and referring comprehension. Navigation-centric benchmarks such as AITZ [50], AndroidControl [23] and Mind2Web [10] evaluate task understanding and next-step prediction, yet they rely primarily on single-platform environments with limited GUI diversity. Despite these advances, existing GUI benchmarks remain fragmented and domain-specific. Most concentrate on either grounding [6, 22], navigation [10, 34], or question answering [18], lacking a unified framework for assessing robustness or hallucination behaviors. In contrast, **GUI-HalluBench** expands the focus from interaction capability to reliability by systematically examining how LMMs hallucinate within realistic GUI contexts, which complements prior benchmarks by introducing bilingual coverage and dual-stage annotations (parsing and hallucination).

2.2. Hallucination Benchmarks

Hallucination evaluation has become a central issue for LMMs. Early benchmarks such as Object-HalBench [35] and POPE [26] quantify object- and attribute-level hallucinations in image captioning and grounding tasks. Subsequent works, including HallusionBench [15] and Hal-

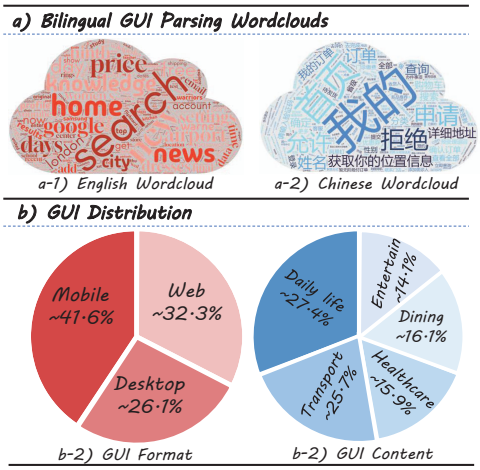

Figure 2. The overview of the bilingual wordclouds and format/-content distributions for the proposed GUI-HalluBench.

luQA [7] broaden the taxonomy to relational, commonsense, and instruction-based hallucinations. More recent datasets like MMHal-Bench [36] and LLaVA-Bench [28] evaluate hallucination under open-ended multimodal reasoning and dialogue settings, revealing the persistent gap between perception and generation. However, these benchmarks primarily focus on natural image scenarios, where hallucinations manifest as visual misidentification or text–image inconsistency. They rarely consider structured visual domains such as graphical user interfaces, where hallucinations take the form of functional confusion or *fabricated interface elements*.

2.3. GUI LMMs

Several works [6, 33, 44, 46, 49] have fine-tuned LMMs to improve the basic grounding and navigation competencies within GUI environments through the aggregation of data from diverse platforms. Seeclick [6] represents the first work to accomplish location and navigation tasks exclusively from *GUI screenshots and natural-language user instructions*, without dependence on plain-text representations such as HTML or XML. Ferret-UI [49] processes screenshot information using a *dynamic resolution* mechanism to capture fine-grained visual features. UI-TARS [33] aggregate extensive multi-platform datasets and incorporate *reasoning mechanisms* alongside reinforcement learning into GUI scenarios, thereby enhancing overall model performance. OmniParser [38] trains a *specialized detection model* based on YOLOv8 to extract all icons from interfaces, then LMMs are employed to comprehend the associated semantic context, and finally GPT-4V [47] is leveraged to support decision-making in grounding tasks.

3. Evaluating Hallucinations

3.1. GUI Pool

To broaden application coverage, we adopt a bilingual data collection approach, including both English and Chinese interfaces. For English GUIs, we directly collect samples from the ScreenSpot [6] and AMEX [4] datasets, which cover the most common interaction scenarios across *Mobile*, *Web*, and *Desktop* platforms. For Chinese GUIs, we manually collect data to better reflect local user contexts. Based on the statistics of popular consumer applications in China [11], we select the top five representative domains (*Daily Life, Transportation, Healthcare, Dining, and Entertainment*) as our primary sources. In total, we collect approximately 5,000 interfaces across these diverse scenarios. The statistics of the collected GUIs are shown in Fig. 2.

3.2. Annotation

As shown in Fig. 3, the annotation process consists of two major stages: *Parsing Annotation* and *Grounding Hallucination Annotation* (*more details in App. 9*).

Parsing Annotation. In this stage, we perform parsing annotation for all 5,000 collected GUIs through a combination of automatic detection and human verification. Specifically, we first apply *Grounding DINO* [30] to detect icons and obtain their corresponding bounding boxes, after which *PaddleOCR* [8] is used to recognize textual elements associated with these icons. The detected objects are then refined via *Non-Maximum Suppression (NMS)* to remove redundancy, followed by an *LMM-based module* that unifies annotation formats and conducts semantic verification to ensure consistency. Finally, human annotators manually review the outputs to correct residual errors and confirm the final results. Through this solid process, we obtain complete and high-quality parsing annotations for all GUIs.

Grounding Hallucination Annotation. Not all GUIs are suitable for hallucination annotation, so we first apply an *LMM-based filtering step* to assess their suitability. This module jointly analyzes each GUI and its corresponding parsing annotation to determine whether it can be transformed into a potential hallucination case (*prompt in App. 9.1*). For the qualified samples, the model further generates *hallucination suggestions* by simulating either a confusion hallucination or a fabricated hallucination scenario (*prompt and cases in App. 9.2*). These automatically generated suggestions are then passed to human annotators, who carefully review, validate, and finalize the hallucination labels. Each hallucination label is reviewed by at least three human annotators to ensure correctness. Through this collaborative process, we obtain consistent and high-quality hallucination annotations across the benchmark.

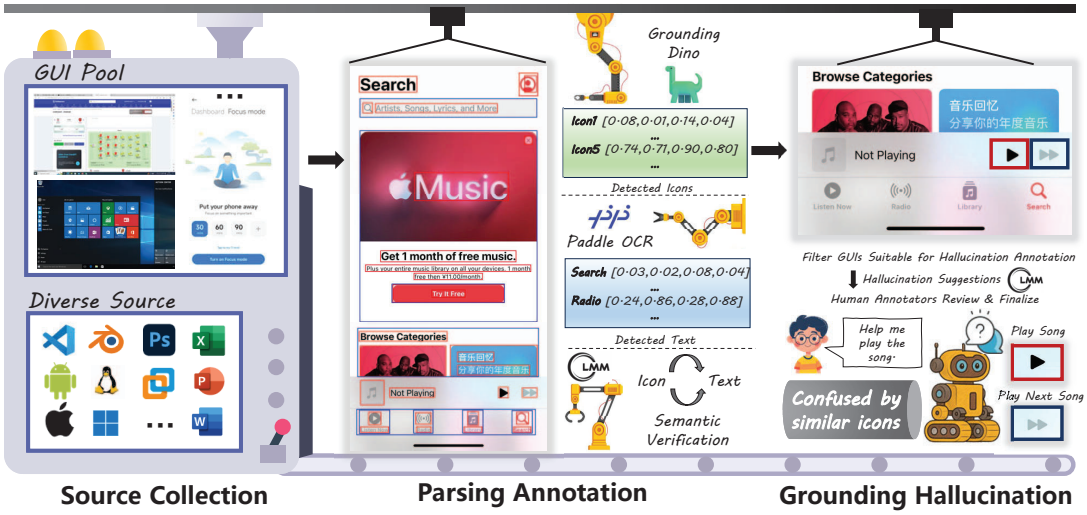

Figure 3. Overview of the annotation pipeline. Around 5,000 GUI samples are collected from diverse sources. Icon bounding boxes and textual elements are detected using *Grounding DINO* [30] and *PaddleOCR* [8], followed by semantic verification to ensure alignment. GUIs that meet the criteria for hallucination annotation are then filtered, after which grounding hallucination labeling is performed.

Sampling & Statistics. With the parsing and grounding hallucination annotations in place, we perform a sampling procedure guided by the principles of *redundancy* [57] and *information density* [21], which are commonly adopted in LMM benchmarks[42, 51–53]. Specifically, we compute diversity quantitative indicators for all annotated instances, and then apply a balanced sampling strategy that aims to distribute these indicators as uniformly as possible across the selected subset based on these attributes. Following this procedure, we obtain a balanced subset consisting of *1,000 English* and *1,000 Chinese* instances, each containing both a parsing annotation and a grounding hallucination annotation. Among them, approximately *40%* are labeled as confusion hallucination and *60%* as fabricated hallucination, reflecting the fact that fabricated cases are more prevalent in real-world GUI scenarios (*samples in Fig. 4*).

3.3. Prompt

3.3.1. Parsing Prompt

To standardize model inputs, we design a unified prompt that explicitly instructs the model to identify all GUI elements and report their semantics with spatial coordinates. This ensures consistency across platforms and provides a structured format for evaluating element recognition and localization accuracy. The prompt is formatted as follows:

#User: Parse the semantics of all elements (including icons and texts) on the interface and their corresponding coordinates. The format of the coordinates is [x1,y1,x2,y2]. The following is an example output: icon: play music [x1,y1,x2,y2] text: search for my favorites [x1,y1,x2,y2] text: Today's Hot Songs [x1,y1,x2,y2] icon: play the next song [x1,y1,x2,y2]

3.3.2. Grounding Prompt

The grounding prompt focuses on task-oriented localization, requiring the model to output the coordinates corresponding to a specific functional description (*denoted as* {*FUNC_DESC*}). This concise prompt enables direct evaluation of grounding accuracy and hallucination behavior under different instruction conditions, which is derived as:

#User: Output the corresponding coordinates based on the interface and the given instruction: {*FUNC_DESC*}*. The output must be only [x1,y1,x2,y2].*

3.4. Performance Metrics

3.4.1. Parsing Performance

Parsing performance is primarily evaluated based on the accuracy of bounding-box detection, which reflects the model's ability to localize GUI elements correctly. Two complementary localization metrics (element recall & element precision) are employed to measure detection coverage and reliability, respectively. In addition, to account for the semantic faithfulness of identified elements, the function similarity is introduced to assess the alignment between predicted and ground-truth semantics.

Element Recall. Element recall measures the proportion of ground-truth elements that are correctly localized:

$$R = \frac{N_m}{N_{gt}}, \tag{1}$$

where N_m denotes the number of predicted bounding boxes that successfully match a ground-truth element (e.g., IoU $\geq$

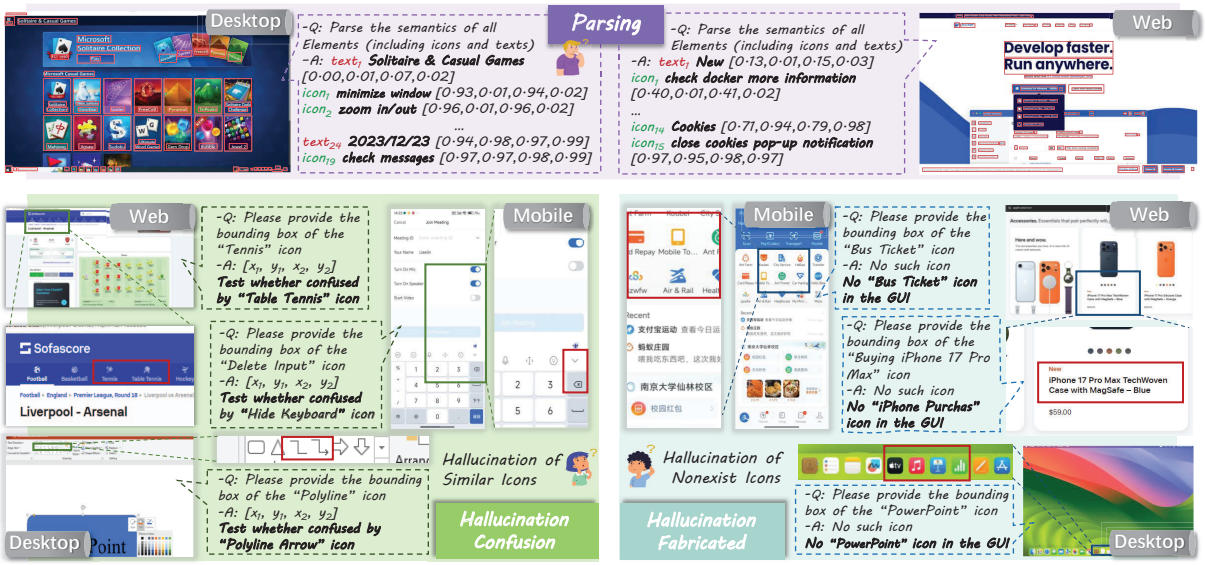

Figure 4. Examples from **GUI-HalluBench** illustrating parsing and grounding hallucination cases across *Desktop*, *Web*, and *Mobile* platforms. The top row shows **Parsing** prompts where models must recognize all interface elements and output their semantics with coordinates. The bottom row shows two hallucination types: **Confusion Hallucination** (misidentifying similar icons, e.g., *Tennis* vs. *Table Tennis*) and **Fabricated Hallucination** (hallucinating nonexistent icons, e.g., *Bus Ticket* or *PowerPoint*).

0.5), and N_{gt} is the total number of ground-truth elements. High recall indicates that the model can detect most of the existing GUI components.

Element Precision. Element precision quantifies the proportion of predicted elements that correspond to valid ground-truth instances:

$$P = \frac{N_m}{N_{pr}}, \tag{2}$$

where N_{pr} is the total number of predicted elements. High precision reflects the model's ability to minimize false positives and maintain localization reliability.

Function Similarity. Function similarity evaluates the semantic alignment between predicted and ground-truth elements for matched bounding boxes:

$$FS = \frac{1}{N_m} \sum_{i=1}^{N_m} \text{sim}\left(f_i^{pred}, f_i^{\text{gt}}\right), \tag{3}$$

where $\text{sim}(\cdot,\cdot)$ denotes the text embedding cosine similarity between the predicted function description f_i^{pred} and its ground-truth counterpart f_i^{gt}. This metric captures the model's capability to produce semantically faithful descriptions of localized elements, which is essential for advanced GUI understanding and interaction.

3.4.2. Grounding Hallucination Performance

To comprehensively assess grounding hallucination behaviors, tailored evaluation metrics are designed for each hallucination type according to their distinct characteristics.

Confusion Hallucination. Localization accuracy is used to measure the correctness of grounding between predicted and ground-truth elements:

$$LA = \frac{N_{cor}}{N_{tot}}, \tag{4}$$

where N_{cor} is the number of correctly grounded instances, and N_{tot} is the total number of evaluated samples.

Fabricated Hallucination. The rejection rate metric quantifies the model's capacity to correctly abstain from producing nonexistent or spurious elements:

$$RR = \frac{N_{rej}}{N_{fab}}, \tag{5}$$

where N_{rej} denotes the number of fabricated elements that the model successfully refuses to recognize, and N_{fab} is the total number of fabricated cases. A higher rejection rate indicates stronger robustness against over-generation and false perception.

Table 2. Performance comparison on GUI-HalluBench. The best results are marked in bold, and the best results among LMMs *without* hallucination-easing approaches are underlined. Certain post-trained models are exclusively fine-tuned on GUI grounding or navigation tasks, which limits their generalization and instruction-following ability. Therefore, their parsing performance is left blank.

Model	**Parsing** *(English)*			**Parsing** *(Chinese)*			*Avg. (%)*	**Hallu.** *(English)*		**Hallu.** *(Chinese)*		*Avg. (%)*
	P (%)	*R (%)*	*FS (%)*	*P (%)*	*R (%)*	*FS (%)*		*LA (%)*	*RR (%)*	*LA (%)*	*RR (%)*	
Metric Index	A	B	C	D	E	F	G	H	I	J	K	L
closed-source LMMs												
GPT-4o (with grounding) [20]	4.3	10.2	57.4	3.4	6.3	39.4	20.2	47.2	69.4	44.8	67.4	57.2
Claude Computer Use [2]	16.5	34.8	75.4	18.5	30.6	50.7	37.7	48.1	66.7	45.5	65.9	56.6
Gemini-2.0 (Project Mariner) [12]	19.8	35.7	80.1	25.6	37.5	51.2	41.6	50.7	71.6	49.5	70.0	60.5
Open-source LMMs												
InternVL3.5-8B [40]	20.4	38.6	73.6	44.1	72.6	84.7	55.7	58.5	64.0	63.9	66.4	63.2
Qwen3-VL-8B [14]	28.6	40.2	74.5	44.5	74.1	86.4	58.1	62.2	67.8	64.4	69.6	66.0
SeeClick-9.6B [6]	/	/	/	/	/	/	/	38.3	25.1	34.9	22.8	30.3
CogAgent-9B [17]	15.8	19.4	66.1	23.5	37.3	68.0	38.4	43.4	33.6	48.0	36.4	40.4
OS-Atlas-7B [44]	2.2	8.3	44.6	3.1	5.4	33.2	16.1	45.0	34.7	47.2	37.5	41.1
UI-TARS-7B [33]	2.1	4.8	70.7	5.3	7.4	69.8	26.7	50.8	41.1	50.4	38.1	45.1
Aguvis-7B [46]	/	/	/	/	/	/	/	48.2	38.3	45.6	37.5	42.4
UGround-7B [13]	/	/	/	/	/	/	/	55.7	38.0	50.5	37.6	39.1
JEDI-7B [45]	12.6	14.1	72.6	19.6	29.8	61.9	35.1	63.3	43.5	61.3	39.1	51.8
GUI-Actor-7B [43]	22.1	34.6	70.2	37.8	66.4	79.5	51.8	64.9	62.5	62.1	56.9	61.6
GUI-Owl-7B [48]	25.4	37.4	73.2	41.6	70.5	84.5	55.4	66.0	62.5	67.4	63.7	64.9
Proposed Training-Free Approach												
GPT-4o (with grounding) (**PGP**)	4.3	10.2	57.4	3.4	6.3	39.4	20.2	50.4	70.2	46.1	68.2	58.7
Claude Computer Use (**PGP**)	16.5	34.8	75.4	18.5	30.6	50.7	37.7	51.5	69.2	48.4	67.6	59.2
InternVL3.5-8B (**PGP**)	20.4	38.6	73.6	44.1	72.6	84.7	55.7	59.8	65.0	66.2	68.3	64.8
Qwen3-VL-8B (**PGP**)	28.6	40.2	74.5	44.5	74.1	86.4	58.1	65.3	68.5	66.6	70.2	67.7
Proposed Training-Based Approach												
InternVL3.5-8B (**HFT**)	58.2	58.7	84.8	66.9	75.0	87.7	71.9	68.6	67.9	69.7	70.3	69.1
Qwen3-VL-8B (**HFT**)	**58.4**	**59.3**	**85.2**	**68.0**	**75.1**	**88.2**	**72.3**	**69.1**	**74.3**	**71.7**	**76.9**	**73.0**

4. Easing Hallucinations

Hallucinations in GUI grounding often arise from incomplete structural perception and insufficient reasoning consistency. To address these issues, we propose two complementary strategies that enhance grounding reliability from both inference-time and training-time perspectives: a training-free approach, *Parsing-guided Prompt (PGP)*, which introduces explicit structural reasoning through prompt design, and a training-based method, *Hallucination-aware Supervised Fine-tuning (HFT)*, which leverages automatically generated hallucination data to improve robustness during model adaptation.

4.1. Training-Free Approach

The training-free strategy mitigates hallucinations purely through the **Parsing-guided Prompt (PGP)** without any additional model tuning. Instead of directly instructing the model to localize a functional element, we reformulate the instruction into a *single-step structured prompt* that enforces an explicit *parsing-before-grounding* reasoning process within a single inference round. The prompt compels the model to first parse the interface and enumerate all visible GUI elements (icons and texts) with their semantics and spatial coordinates, and then to identify the coordinates corresponding to the specified functional description (*More details in App. 10.1*). The prompt is formatted as follows:

#User: First, parse all interface elements (including icons and texts) on the screen and provide their semantics with coordinates in the format [x1,y1,x2,y2]. Then, based on this parsed information, output the coordinates of the element that matches the given instruction: {FUNC_DESC}. The output must be only [x1,y1,x2,y2].

4.2. Training-Based Approach

While the above approach enhances reasoning consistency during inference, we further reduce hallucinations through **Hallucination-aware Supervised Fine-tuning (HFT)**. To prevent data leakage, we create an *independent interface dataset* whose sources are entirely separate from those of GUI-HalluBench. Following the design outlined in Section 3.2, we first collect and annotate 20K interfaces with parsing information, and then we proceed to annotate these interfaces with hallucinations..

Specifically, when LMMs generate hallucination transformation suggestions (whether involving confusion or fabrication) these outputs are recorded. To ensure the quality of the generated labels, while also enhancing scalability and reducing manual effort, we implement a *single annotator validation process*. After the initial labels are generated, a trained annotator reviews each automatically produced hallucination, with the primary goal of identifying and removing labels of particularly poor quality. As a result, we obtain a final set of 10K interfaces with hallucination annotations. (*More details and samples in App. 10.2*).

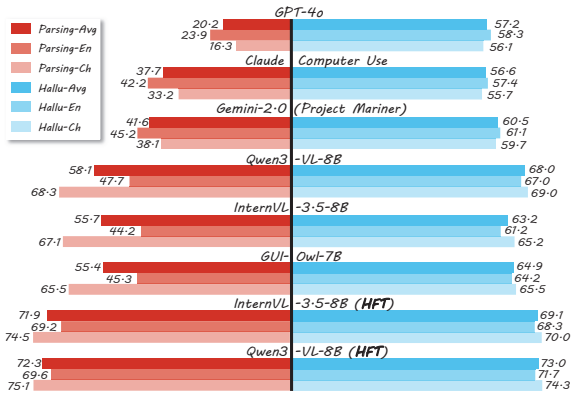

Figure 5. Overview of overall and bilingual performance across *closed-source* models, leading *open-source* models, and LMMs enhanced by the proposed hallucinations easing approaches.

5. Experiment

5.1. Benchmark Candidates

For empirical evaluation, we include a suite of representative baselines that are widely recognized in the literature and have demonstrated strong performance on GUI-related tasks, allowing for a thorough comparison with established methodologies. These include *closed-source* commercial models such as GPT-4o [20], Claude Computer Use [2], and Gemini 2.0 (Project Mariner) [12], as well as *open-source* models like Qwen3-VL-8B [14], InternVL3.5-8B [40], SeeClick [6], CogAgent [17], OS-Atlas [44], UI-TARS [33], Aguvis [46], UGround [13], JEDI [45], GUI-Actor [43], and GUI-Owl [48].

5.2. HFT Setup

Source Datasets. As detailed in Table 3, we leverage not only self-built data but also a curated collection of publicly available datasets, encompassing both GUI-specific resources including WidgetCaption [25], UI RefExp [3], RICO-Semantics [37], RICO-SCA [24], SeeClick-Web [6], Os-Atlas [44], GUIEnv [5], ScreenQA [18], and ShowUI [27], as well as general resources such as ShareGPT-Computer [16] and LLaVA-Instruct [29]. This data composition is designed to enhance the model's capacity for comprehensive GUI understanding while simultaneously maintaining its instruction-following proficiency.

Implementation Details. Qwen3-VL-8B [14] & InternVL3.5-8B [40] are adopted and fine-tuned via *LoRA* [19]. Experiments are performed on 8 NVIDIA 80G A100, with the parameters of the *ViT* and *aligner* module frozen. The LoRA rank r and α are configured to *8* and *32* respectively, setting the learning rate to 1×10^{-4} with a warmup ratio of *0.03* and a total training epochs of *3*.

Table 3. Datasets composition for our proposed **Hallucination-aware Fine-Tuning (HFT)**.

Data Source	Platform	Task Type	Samples
Existing Public Datasets			
WidgetCaption [25]	Mob.	Grounding	100K
UI RefExp [3]	Mob.	Grounding	16K
RICO-Semantics [37]	Mob.	Grounding	31K
RICO-SCA [24]	Mob.	Grounding	71K
Os-Atlas [44]	Mob. & Desk.	Grounding	339K
ShowUI [27]	Mob. & Desk. & Web	Grounding	19K
GUIEnv [5]	Web	Grounding	340K
SeeClick-Web [5]	Web	Grounding	54K
ScreenQA [18]	Mob.	VQA	62K
ShareGPT-Computer [16]	–	General	26K
LLaVA-Instruct [29]	–	General	150K
Self-built Datasets (More details in Sec. 10.2)			
Global Parsing	Mob. & Desk. & Web	Parsing	20K
Hallucination-aware Grounding	Mob. & Desk. & Web	Grounding	10K

5.3. Findings

Open-source LMMs Outperform Closed-source Alternatives from Overall Performance Landscape. As shown in Table 2, both *closed-source* and *open-source* LMMs exhibit clear performance stratification across parsing and hallucination subtasks. Surprisingly, *the closed-source models (despite their superior general reasoning capabilities) do not lead in either subtask.* This contrast highlights that strong general multimodal reasoning does not directly translate into reliable GUI understanding, which inherently requires domain-adapted structural perception and fine-grained visual grounding. Among closed-source LMMs, *Gemini-2.0 (Project Mariner)* achieves the highest overall accuracy (60.5%), surpassing the others but still lagging behind the top-performing open-source counterparts. In particular, *GPT-4o*, though renowned for its reasoning strength, exhibits extremely low element precision (4.3%), revealing limited grounding robustness under cluttered interface conditions. By contrast, open-source models such as *InternVL3.5-8B* and *Qwen3-VL-8B*, which have undergone targeted domain alignment and structural supervision, deliver more consistent and accurate performance across both parsing and hallucination subtasks, demonstrating that domain-specific adaptation is crucial for mastering GUI-oriented perception and interaction.

Bilingual Inconsistency. As shown in Fig. 5, across all evaluated models, a consistent English–Chinese performance gap emerges in both parsing and hallucination subtasks. Two observations can be drawn as follows: 1) The bilingual gap on the parsing subtask is notably larger than that on hallucination, suggesting that structural perception and element recognition are more sensitive to language-dependent cues such as embedded text, icon semantics, and OCR noise. In contrast, hallucination robustness relies more on visual reasoning and rejection ability, which are relatively language-agnostic. 2) The models developed

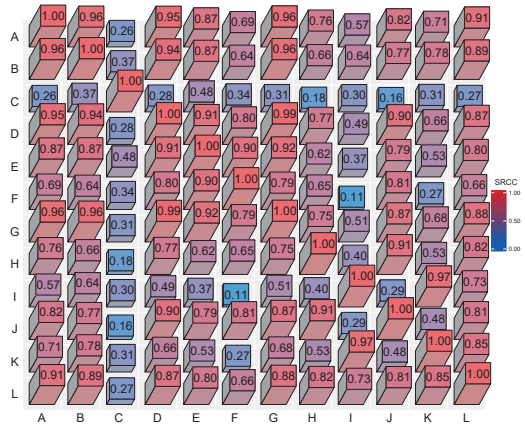

Figure 6. Metric correlation (*SRCC*) heatmap, where higher values indicate stronger correlations between corresponding metrics. The metric index can be found in Table 2.

by English-speaking companies (e.g., *GPT-4o*, *Claude*) achieve significantly better performance on English interfaces, whereas those trained by Chinese companies (e.g., *Qwen3-VL*, *InternVL*) show the opposite trend. This pattern indicates that current LMMs still inherit strong linguistic and cultural biases from their training corpora, underscoring the need for truly balanced multilingual GUI evaluation.

Hallucinations Correlate with Parsing. To investigate the relationship between parsing and grounding hallucinations, we follow the correlation analysis setup outlined in [57] and find that: A strong correlation is observed between **L** (parsing average performance) and **G** (hallucination average performance). Models with higher element precision and recall in the parsing task tend to achieve better robustness against hallucinations. This finding further supports our introductory statement that *grounding hallucinations are not isolated but closely tied to deficiencies in parsing*. Additionally, we observe that metrics **C** and **F** exhibit weaker correlations with hallucination metrics. This can be attributed to the fact that **C** and **F** measure function similarity, which assesses the semantic alignment between predicted and ground-truth elements for matched bounding boxes. However, hallucinations are more closely tied to issues in element detection and contextual reasoning, which are not directly captured by function similarity. As such, the underlying capabilities assessed by function similarity and hallucination metrics differ fundamentally, resulting in the observed weaker correlation between the two.

Effectiveness of Hallucination Easing Approaches. The proposed hallucination easing approaches effectively reduce hallucination behaviors at varying costs, with performance improvements illustrated in Fig. 7. The *Parsing-Guided Prompt (PGP)* enhances inference reliability by explicitly enforcing a 'parse-then-ground' reasoning sequence, resulting in modest gains in **H-K** (LA/RR) without requiring any retraining. While the improvements are not substantial, the approach avoids the need for additional training. This effect is particularly evident for **I** and **K** metrics (rejection rate for English and Chinese), as parsing improves the model's decision-making process by ensuring more accurate identification and localization of relevant elements, reducing errors that contribute to hallucinations in both languages. The *Hallucination-aware Fine-Tuning (HFT)* further amplifies these gains, yielding up to a 7% absolute improvement over base models. For example, Qwen3-VL-8B shows an increase from 66.0% to 73.0% in average accuracy, demonstrating that the hallucination simulation dataset serves as an effective tool for enhancing robustness through data augmentation. Additionally, HFT generalizes well across languages, confirming that hallucination suppression can be effectively transferred cross-lingually once structural perception is reinforced.

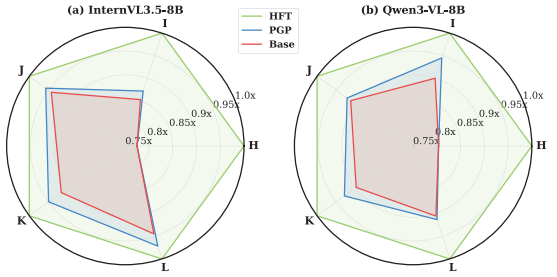

Figure 7. Improvement illustration of the proposed hallucination easing approaches, with performance normalized as the value divided by the maximum value for each metric dimension. The metric index can be referred to in Table 2.

6. Conclusion

This paper presents the first systematic study of hallucinations in large multimodal models (LMMs) for graphical user interface (GUI) grounding, categorizing them into **Confusion Hallucination** and **Fabricated Hallucination**. We then introduce **GUI-HalluBench**, a benchmark designed to evaluate and analyze hallucinations in GUI scenarios, and demonstrate how parsing deficiencies contribute to hallucinations. Additionally, two strategies (Parsing-guided Prompt (PGP) and Hallucination-aware Fine-Tuning (HFT)) are proposed to effectively mitigate these hallucinations. Experiments show that improving parsing capabilities reduces hallucination rates, providing valuable insights for enhancing LMM reliability in GUI interactions.

Acknowledgements

This work was supported by New Generation Artificial Intelligence-National Science and Technology Major Project (2025ZD0124104) in collaboration with Shanghai Artificial Intelligence Laboratory, in part by the China Postdoctoral Science Foundation under Grant 2025M781485 and National Natural Science Fund of China No. 62501337.

References

[1] Reyna Abhyankar, Qi Qi, and Yiying Zhang. Osworld-gold: Benchmarking the efficiency of computer-use agents. In *ICML 2025 Workshop on Computer Use Agents*, 2025. 2

[2] Anthropic. Developing a computer use model. Technical report, Anthropic, 2024. 1, 6, 7

[3] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*, 2021. 7, 5

[4] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2138–2156, 2025. 3

[5] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language model to versatile gui agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21936–21959, 2025. 7, 5

[6] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*, 2024. 1, 2, 3, 6, 7, 5

[7] Qinyuan Cheng, Tianxiang Sun, Wenwei Zhang, Siyin Wang, Xiangyang Liu, Mozhi Zhang, Junliang He, Mianqiu Huang, Zhangyue Yin, Kai Chen, et al. Evaluating hallucinations in chinese large language models. *arXiv preprint arXiv:2310.03368*, 2023. 2, 3

[8] Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, et al. Paddleocr 3.0 technical report. *arXiv preprint arXiv:2507.05595*, 2025. 3, 4

[9] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 845–854, 2017. 2

[10] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023. 1, 2

[11] Enying Gong, Zongmuyu Zhang, Xurui Jin, Yishan Liu, Lumin Zhong, Yao Wu, Xuefeng Zhong, Lijing L Yan, and Brian Oldenburg. Quality, functionality, and features of chinese mobile apps for diabetes self-management: systematic search and evaluation of mobile apps. *JMIR mHealth and uHealth*, 8(4):e14836, 2020. 3

[12] GoogleDeepmind. Gemini-2.0 (project mariner). Technical report, GoogleDeepmind, 2024. 6, 7, 1

[13] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*, 2024. 6, 7

[14] Alibaba Group. Qwen3-vl. https://github.com/QwenLM/Qwen3-VL, 2025. 6, 7, 1

[15] Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385, 2024. 2

[16] HF Datasets Community. Sharegpt-computer. https://huggingface.co/datasets/svjack/ShareGPT-computer-en-zh-26k-human-gpt, 2024. 7, 5

[17] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024. 6, 7, 1

[18] Yu-Chung Hsiao, Fedir Zubach, Gilles Baechler, Victor Carbune, Jason Lin, Maria Wang, Srinivas Sunkara, Yun Zhu, and Jindong Chen. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199*, 2022. 1, 2, 7, 5

[19] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. 7

[20] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 6, 7, 1

[21] Chunyi Li, Xiaozhe Li, Zicheng Zhang, Yuan Tian, Ziheng Jia, Xiaohong Liu, Xiongkuo Min, Jia Wang, Haodong Duan, Kai Chen, et al. Information density principle for mllm benchmarks. *arXiv preprint arXiv:2503.10079*, 2025. 4

[22] Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use. *arXiv preprint arXiv:2504.07981*, 2025. 1, 2

[23] Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on computer control agents. *arXiv e-prints*, pages arXiv–2406, 2024. 1, 2

[24] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*, 2020. 7, 5

[25] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. *arXiv preprint arXiv:2010.04295*, 2020. 7, 5

[26] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023. 2

[27] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for generalist gui agent. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024. 7, 4, 5

[28] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023. 3

[29] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 7, 5

[30] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 3, 4

[31] Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024. 1

[32] OpenAI. Operator. https://openai.com/index/introducing-operator, 2025. 1

[33] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025. 3, 6, 7, 1

[34] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728, 2023. 2

[35] Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. Object hallucination in image captioning. *arXiv preprint arXiv:1809.02156*, 2018. 2

[36] Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, et al. Aligning large multimodal models with factually augmented rlhf. *arXiv preprint arXiv:2309.14525*, 2023. 3

[37] Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Jindong Chen, Abhanshu Sharma, and James WW Stout. Towards better semantic understanding of mobile interfaces. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5636–5650, 2022. 7, 5

[38] Jianqiang Wan, Sibo Song, Wenwen Yu, Yuliang Liu, Wenqing Cheng, Fei Huang, Xiang Bai, Cong Yao, and Zhibo Yang. Omniparser: A unified framework for text spotting key information extraction and table recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15641–15653, 2024. 3

[39] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 498–510, 2021. 1, 2

[40] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025. 6, 7, 1

[41] Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, et al. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. *arXiv preprint arXiv:2507.19478*, 2025. 1, 2, 4

[42] Haoning Wu, Zicheng Zhang, Erli Zhang, Chaofeng Chen, Liang Liao, Annan Wang, Chunyi Li, Wenxiu Sun, Qiong Yan, Guangtao Zhai, et al. Q-bench: A benchmark for general-purpose foundation models on low-level vision. *arXiv preprint arXiv:2309.14181*, 2023. 4

[43] Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *arXiv preprint arXiv:2506.03143*, 2025. 6, 7

[44] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024. 2, 3, 6, 7, 1, 5

[45] Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025. 6, 7

[46] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*, 2024. 3, 6, 7

[47] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1):1, 2023. 3

[48] Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, et al. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*, 2025. 1, 6, 7

[49] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *European Conference on Computer Vision*, pages 240–255. Springer, 2024. 3

[50] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024. 1, 2

[51] Zicheng Zhang, Ziheng Jia, Haoning Wu, Chunyi Li, Zijian Chen, Yingjie Zhou, Wei Sun, Xiaohong Liu, Xiongkuo Min, Weisi Lin, et al. Q-bench-video: Benchmarking the video quality understanding of lmms. *arXiv preprint arXiv:2409.20063*, 2024. 4

[52] Zicheng Zhang, Haoning Wu, Chunyi Li, Yingjie Zhou, Wei Sun, Xiongkuo Min, Zijian Chen, Xiaohong Liu, Weisi Lin, and Guangtao Zhai. A-bench: Are lmms masters at evaluating ai-generated images? *arXiv preprint arXiv:2406.03070*, 2024.

[53] Zicheng Zhang, Haoning Wu, Erli Zhang, Guangtao Zhai, and Weisi Lin. Q-bench$^{\hat{}}$++: A benchmark for multi-modal foundation models on low-level vision from single images to pairs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10404–10418, 2024. 4

[54] Zhong Zhang, Yaxi Lu, Yikun Fu, Yupeng Huo, Shenzhi Yang, Yesai Wu, Han Si, Xin Cong, Haotian Chen, Yankai Lin, et al. Agentcpm-gui: Building mobile-use agents with reinforcement fine-tuning. *arXiv preprint arXiv:2506.01391*, 2025. 2

[55] Zicheng Zhang, Junying Wang, Yijin Guo, Farong Wen, Zijian Chen, Hanqing Wang, Wenzhe Li, Lu Sun, Yingjie Zhou, Jianbo Zhang, Bowen Yan, Ziheng Jia, Jiahao Xiao, Yuan Tian, Xiangyang Zhu, Kaiwei Zhang, Chunyi Li, Xiaohong Liu, Xiongkuo Min, Qi Jia, and Guangtao Zhai. Aibench: Towards trustworthy evaluation under the 45° law. *Displays*, page 103255, 2025. 1

[56] Zicheng Zhang, Junying Wang, Farong Wen, Yijin Guo, Xiangyu Zhao, Xinyu Fang, Shengyuan Ding, Ziheng Jia, Jiahao Xiao, Ye Shen, Yushuo Zheng, Xiaorong Zhu, Yalun Wu, Ziheng Jiao, Wei Sun, Zijian Chen, Kaiwei Zhang, Kang Fu, Yuqin Cao, Ming Hu, Yue Zhou, Xuemei Zhou, Juntai Cao, Wei Zhou, Jinyu Cao, Ronghui Li, Donghao Zhou, Yuan Tian, Xiangyang Zhu, Chunyi Li, Haoning Wu, Xiaohong Liu, Junjun He, Yu Zhou, Hui Liu, Lin Zhang, Zesheng Wang, Huiyu Duan, Yingjie Zhou, Xiongkuo Min, Qi Jia, Dongzhan Zhou, Wenlong Zhang, Jiezhang Cao, Xue Yang, Junzhi Yu, Songyang Zhang, Haodong Duan, and Guangtao Zhai. Large multimodal models evaluation: A survey. *SCIENCE CHINA Information Sciences*, 2025. 1

[57] Zicheng Zhang, Xiangyu Zhao, Xinyu Fang, Chunyi Li, Xiaohong Liu, Xiongkuo Min, Haodong Duan, Kai Chen, and Guangtao Zhai. Redundancy principles for mllms benchmarks. *arXiv preprint arXiv:2501.13953*, 2025. 4, 8

Evaluating and Easing Hallucinations for GUI Grounding

Supplementary Material

7. Performance on GUI Formats

Open-source Models vs. Closed-source Models. The performance of both closed-source and open-source models varies across different platforms, as shown in Table 4. Closed-source models like GPT-4o (with grounding), Claude Computer Use, and Gemini-2.0 (Project Mariner) show good performance on Web and Desktop, but struggle on Mobile, with GPT-4o achieving the highest on Web (69.4%) and Claude Computer Use performing better on Web (66.7%) than Mobile (37.7%). Gemini-2.0 performs better on Mobile (46.6%) but lags behind the others in comparison to Web (65.6%). In contrast, open-source models such as InternVL3.5-8B and Qwen3-VL-8B outperform their closed-source counterparts, particularly on Mobile and Desktop. Qwen3-VL-8B leads with a balanced performance across platforms (58.1% on Mobile, 62.2% on Desktop, and 67.8% on Web), while InternVL3.5-8B performs similarly (57.7% on Mobile, 58.5% on Desktop, and 66.0% on Web), suggesting better generalization across formats. However, SeeClick-9.6B and CogAgent-9B underperform, particularly on Mobile, with SeeClick-9.6B scoring as low as 28.3%, highlighting its limitations across platforms.

Mobile GUIs are More Challenging. The performance comparison in Table 4 highlights that Mobile GUI tasks generally yield lower performance, irrespective of whether they are closed-source or open-source, compared to Desktop and Web. The gap in performance between mobile and the other formats can be attributed to the constraints of mobile devices, such as smaller screen sizes, reduced computational power, and touch-based interaction. These factors require models to be optimized for mobile-specific challenges, which is not always the case, especially for models trained primarily for desktop or web applications.

Effectiveness of Proposed Approaches. The proposed approaches (PGP & HFT) both enhance model performance across different GUI formats, with HFT yielding the best results overall. The training-free approach using PGP boosts performance, particularly on Mobile, with Qwen3-VL-8B (PGP) reaching 60.3%, and InternVL3.5-8B (PGP) showing robust improvements across all formats. However, some models, such as Claude Computer Use (PGP), still exhibit relatively weaker performance on Mobile and Web, indicating that while the PGP method offers clear advantages, it faces challenges in adapting all models to various interface types. On the other hand, the training-based approach using HFT achieves the highest performance, with Qwen3-VL-8B (HFT) leading at 71.2% on Mobile, 74.1% on Desktop, and 73.3% on Web, showcasing significant improvements across all formats.

Table 4. Performance comparison on Mobile, Desktop, and Web GUIs. The results are shown as the average performance values within the corresponding GUI format. The best results are marked in bold.

Model	Mobile *(%)*	Desktop *(%)*	Web *(%)*
closed-source LMMs			
GPT-4o (with grounding) [20]	40.2	60.2	69.4
Claude Computer Use [2]	37.7	58.1	66.7
Gemini-2.0 (Project Mariner) [12]	46.6	60.7	65.6
Open-source LMMs			
InternVL3.5-8B [40]	57.7	58.5	66.0
Qwen3-VL-8B [14]	58.1	62.2	67.8
SeeClick-9.6B [6]	28.3	35.1	34.9
CogAgent-9B [17]	33.4	43.6	48.0
OS-Atlas-7B [44]	35.0	44.7	47.2
UI-TARS-7B [33]	30.8	41.1	50.4
Proposed Training-Free Approach			
GPT-4o (with grounding) (**PGP**)	43.4	61.2	69.3
Claude Computer Use (**PGP**)	40.5	60.2	67.6
InternVL3.5-8B (**PGP**)	59.8	65.0	66.2
Qwen3-VL-8B (**PGP**)	60.3	64.0	69.6
Proposed Training-Based Approach			
InternVL3.5-8B (**HFT**)	69.9	71.6	70.9
Qwen3-VL-8B (**HFT**)	**71.2**	**74.1**	**73.3**

8. Details for Evaluation

In **Fabricated Hallucination** setting, the model is expected to refuse answering when the queried icon does not exist. A response is considered a *correct refusal* if no coordinate pattern of the form $[x_1, y_1, x_2, y_2]$ appears in the output, and the response explicitly indicates rejection (e.g., stating that the icon does not exist or cannot be located).

To determine whether a model response constitutes a valid refusal, we adopt a two-step hybrid evaluation protocol that combines rule-based detection with LLM-as-a-judge verification: 1) Rule-based detection. We first apply deterministic rules to detect coordinate patterns and explicit refusal expressions. Specifically, we search for bounding-box patterns (e.g., $[x_1, y_1, x_2, y_2]$) as well as refusal-related keywords such as *"no such icon"*, *"not found"*, or similar variants indicating the absence of the queried element. 2) LLM-as-a-judge verification. For ambiguous responses (e.g., paraphrased or indirect refusals that cannot be reliably captured by predefined rules), we employ a lightweight LLM-as-a-judge to perform a binary classification, determining whether the response effectively constitutes a refusal. This hybrid protocol ensures reliable detection of fabricated hallucinations while maintaining robustness to linguistic variations in model outputs.

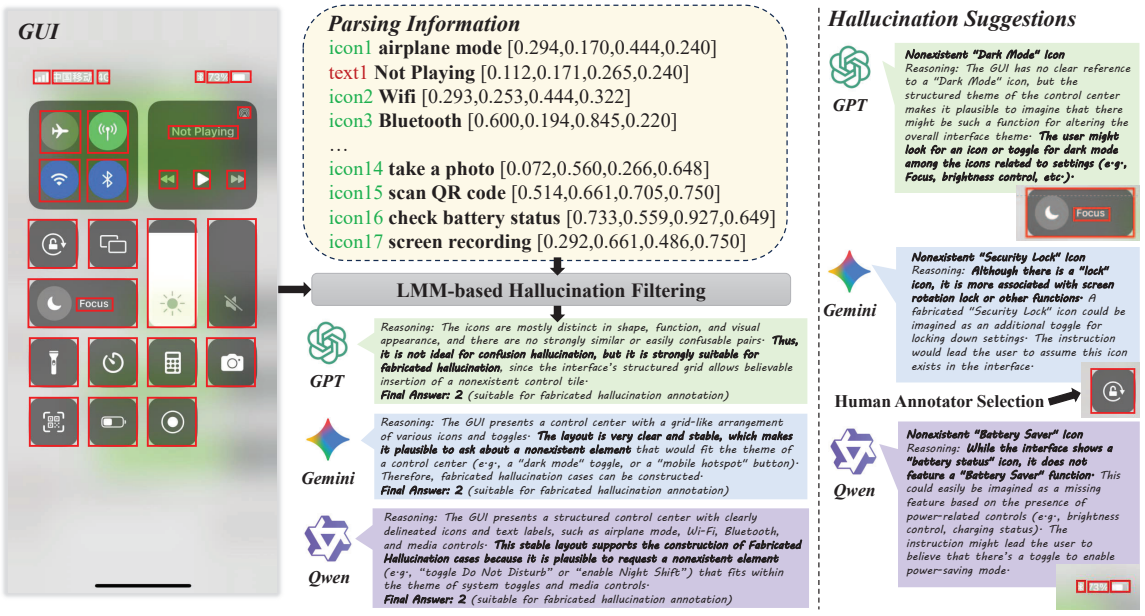

Figure 8. A complete LMM-based hallucination case. The LMMs receive the GUI and parsing information and are prompted to determine whether the GUI is suitable for hallucination generation. Then, the LMMs provide hallucination suggestions for the filtered GUIs. Subsequently, human annotators select the most appropriate suggestion (for example, human annotator select the "Nonexistent Security Lock Icon" suggestion in this case).

9. Details for Annotation

This section provides detailed information on **LMM-based Hallucination Filtering** and **LMM-based Hallucination Suggestions**. A complete LMM-based hallucination case is illustrated in Fig. 8. Following this, human annotators are involved in selecting the most appropriate suggestion, making any necessary modifications, and then performing the final verification.

9.1. LMM-based Hallucination Filtering

To identify which GUI instances are suitable for hallucination construction, we employ an LMM-based filtering module that jointly analyzes each interface and its corresponding parsing annotation. Given the full set of ∼5,000 parsed GUIs, the model evaluates whether an interface contains a clear structural layout and whether its elements can plausibly support either confusion or fabricated hallucination scenarios.

The module takes as input the screenshot and the complete list of parsed elements (icons and texts with bounding boxes). It is prompted to inspect: 1) element diversity, such as the presence of visually similar components that may induce confusion, 2) structural completeness, ensuring that the interface is sufficiently well-formed for reliable transformation, and 3) feasibility of generating unambiguous hallucination instructions. Only GUIs recognized as **hallucination-constructible** are passed to the next annotation stage. This automated filtering reduces human workload and ensures that hallucination annotation is conducted only on clean and structurally suitable GUIs. The filtering prompt is as follows:

You are given:
1) A GUI screenshot. [`Image`]
2) A complete list of parsed GUI elements [`Parsing_Information`], where each element includes:
– its type (icon/text),
– its semantic label,
– its bounding box in the format [x1, y1, x2, y2].

Your task is to carefully analyze whether this GUI is suitable for constructing hallucination evaluation cases. There are two types of hallucination cases we may construct:

1) Confusion Hallucination: This requires the GUI to contain visually or semantically similar elements, such that

a model could plausibly confuse one element for another.

2) Fabricated Hallucination: This requires the GUI to have a clear and stable structure, where it is reasonable to design an instruction about a nonexistent element. The nonexistent element must not appear in the parsing annotation, yet the GUI's theme should make such a fabricated request plausible.

(3) Not Suitable: If the GUI contains too few elements, lacks clear structure, has inconsistent parsing, or does not support either confusion or fabrication scenarios, classify it as unsuitable.

Important rules:
– Base your judgment strictly on the screenshot and the parsed element list.
– Do NOT hallucinate additional elements.
– Do NOT rely on prior knowledge about apps.
– You must logically justify your decision before giving the final label.

Output Format (strict):
First, provide a short reasoning paragraph beginning with Reasoning:.
Then, output only one final label in a new line as:
Final Answer: X = 0 or 1 or 2
where {0, 1, 2} and means:
– 0 = Not suitable for hallucination construction
– 1 = Suitable for Confusion Hallucination
– 2 = Suitable for Fabricated Hallucination

To reduce individual-model bias, we apply the above prompt to three state-of-the-art LMMs (GPT-4o, Gemini-2.0, and Qwen3-Max). A GUI passes the filtering stage only if all three models return a non-zero flag. The final classification, whether as *Confusion Hallucination (1)* or *Fabricated Hallucination (2)*, is determined by a majority vote among the three models.

9.2. LMM-based Hallucination Suggestions

For each GUI selected through filtering, a second LMM-based module is used to generate hallucination transformation suggestions, simulating how an LMM may fail under two representative failure modes: *Confusion Hallucination* and *Fabricated Hallucination*. The module receives the screenshot and its finalized parsing annotation and is prompted to create task instructions that can induce either type of hallucination. To mitigate individual model biases, we similarly select three state-of-the-art LMMs (GPT-4o, Gemini-2.0, and Qwen3-Max) to independently generate suggestions. Afterward, human annotators review the suggestions and select the most appropriate one.

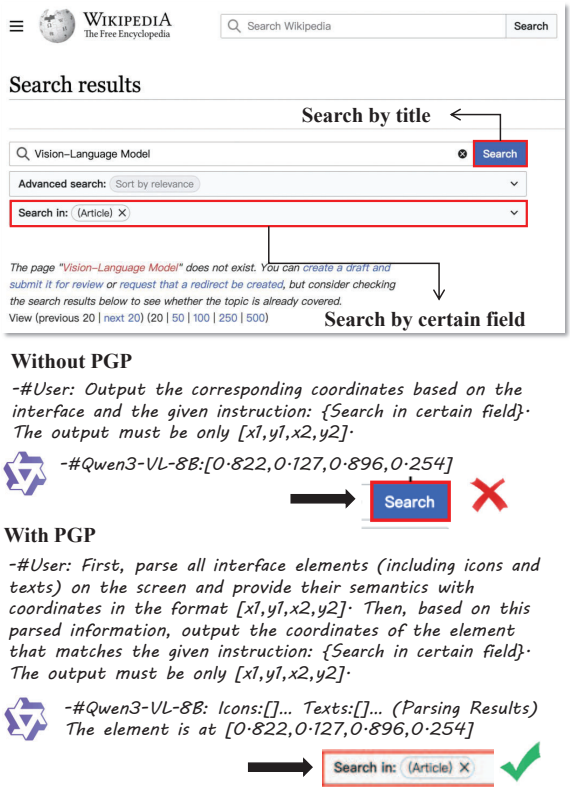

Figure 9. A comparison of confusion hallucination for Qwen3-VL-8B with and without the PGP approach. Without using PGP, Qwen3-VL-8B incorrectly grounds the 'Search by tile' area instead of 'Search by certain field'. With PGP, the model first parses the GUI and then performs grounding, resulting in the correct coordinates for "Search by certain field."

9.2.1. Prompt for Confusion Hallucination Suggestion

You are given:
1) A GUI screenshot. [`Image`]
2) A complete list of parsed GUI elements [`Parsing_Information`], where each element includes:
– its type (icon/text),
– its semantic label,
– its bounding box in the format [x1, y1, x2, y2].

Your task is to analyze the GUI and suggest a Confusion Hallucination scenario. A Confusion Hallucination occurs when the model mistakes a visually or semantically similar element for another.

Based on the given parsed elements and the screenshot, provide a grounding instruction where a model might

confuse two similar elements. Ensure that one of the elements (X) is incorrectly selected in place of another element (Y), either due to visual resemblance or semantic ambiguity.

Your output should be in the following format:
– Suggest one element that is similar to another.
– Reasoning: [explanation of the confusion].

9.2.2. Prompt for Fabricated Hallucination Suggestion

You are given:
1) A GUI screenshot. [`Image`*]*
2) A complete list of parsed GUI elements [`Parsing_Information`*], where each element includes:*
– its type (icon/text),
– its semantic label,
– its bounding box in the format [x1, y1, x2, y2].

Your task is to analyze the GUI and suggest a Fabricated Hallucination scenario. A Fabricated Hallucination occurs when the model generates a completely nonexistent element, providing plausible coordinates for something that does not exist in the interface.

Based on the given parsed elements and the screenshot, provide a suggestion where a fabricated element (one not found in the parsed list) could be generated. The suggested fabricated element should align with the theme or context of the interface, but must not be present in the actual GUI.

Your output should be in the following format:
– Suggest a nonexistent element.
– Reasoning: [explanation of the fabricated element].

9.3. Hallucination Verification

The Hallucination Verification process ensures the quality and reliability of hallucination annotations generated during the LMM-based Hallucination Suggestions phase. After the LMMs (anonymously presented) provide their hallucination suggestions, the next step involves human annotators carefully selecting the most appropriate suggestion and finalizing the hallucination label.

9.3.1. Selection and Finalization

The process begins with the annotator reviewing the hallucination suggestions generated by the three state-of-the-art LMMs. Since the model identities are anonymized, the annotator is unaware of which model produced which suggestion. The annotator evaluates each suggestion, considering factors such as contextual relevance and plausibility, and selects the one they deem most appropriate. After selecting the best suggestion, the annotator finalizes the hallucination label, making any necessary adjustments to ensure that the suggestion is correctly aligned with the GUI structure and user instructions.

9.3.2. Additional Verification by Human Annotators

Once the hallucination label is finalized, it undergoes a secondary verification process. This process involves **three additional human annotators** who independently review the finalized hallucination labels. These annotators assess the label for accuracy and consistency with the original interface. The label is only accepted if all three annotators agree on its correctness. This additional layer of verification ensures that the finalized hallucination annotation is robust and reliable. Based on statistical analysis, the rate of unanimous agreement among the three annotators is 96.7%, demonstrating that the original hallucination labels are of exceptionally high quality.

Table 5. Training Parameters for Fine-tuning.

Parameter	Value
Model	Qwen3-VL-8B, InternVL3.5-8B
Fine-tuning Method	LoRA (Low-Rank Adaptation)
GPUs	8 NVIDIA A100 80GB GPUs
Frozen Modules	ViT (Vision Transformer), Aligner
LoRA Rank (r)	8
Scaling Factor (α)	32
Training Epoch	3
Learning Rate	1×10^{-4}
Warmup Ratio	0.03
Scheduler	CosineLRScheduler

10. Details for Easing Hallucinations

10.1. Training-Free Approach

The Training-Free Approach leverages the Parsing-guided Prompt (PGP) to mitigate hallucinations without the need for model retraining. This approach focuses on improving reasoning consistency during inference by explicitly guiding the model to first parse the GUI interface before grounding. By reformulating the instruction into a structured prompt, we ensure a more reliable and consistent identification of GUI elements, which helps prevent hallucinations caused by misidentification or confusion of interface elements.

10.2. Training-Based Approach

10.2.1. HFT Dataset Construction

The HFT dataset construction process begins with the sampling of source GUI datasets. We sample GUIs from three key datasets: MMBench-GUI-L2 [41], ShowUI [27], and

Os-Atlas [44]. These datasets are selected because they cover a diverse range of interface types, including mobile, desktop, and web environments, and contain a large number of GUI examples. This diversity ensures that the fine-tuning dataset is comprehensive and representative of different real-world GUI scenarios. Importantly, the source datasets used for sampling do not overlap in content with GUI-HalluBench, ensuring that the fine-tuning data is independent and helps avoid any data leakage or bias. For Chinese GUIs, we conducted manual data collection to ensure that the model can handle interfaces specific to Chinese-language environments. This process involves uniform sampling from mobile, desktop, and web interfaces, ensuring that each category is well-represented in the final dataset. Finally, a total of 20,000 GUIs is sampled, with 10,000 in English and 10,000 in Chinese.

The parsing annotation for these GUIs follows the standard procedure outlined in Sec. 3.2, ensuring consistency and accuracy in the representation of GUI elements. However, the process for grounding hallucination annotation differs slightly. The same procedure for hallucination generation described in Sec. 3.2 (LMM-based filtering and hallucination suggestion generation) is applied. Given the large scale of the dataset, **only one human annotator is used to validate the hallucination labels**, which means we do not perform the three-person verification process for hallucination annotations (*as is done for the main GUI-HalluBench dataset*). Ultimately, we obtained 20,000 GUI instances with parsing annotations. After considering the filtering loss, 10,000 instances with hallucination annotations were retained, forming the final HFT dataset. Ultimately, we obtain 20,000 GUI instances with parsing annotations. After accounting for filtering loss, 10,000 instances with hallucination annotations are retained, forming the final HFT dataset.

10.2.2. Implementation Details and Training Procedure

In addition to our self-built dataset, we leverage several publicly available datasets to improve model performance. These datasets include WidgetCaption [25], UI Ref-Exp [3], RICO-Semantics [37], RICO-SCA [24], SeeClick-Web [6], Os-Atlas [44], GUIEnv [5], ScreenQA [18], and ShowUI [27], as well as general resources such as ShareGPT-Computer [16] and LLaVA-Instruct [29]. This diverse collection enhances the model's capacity for comprehensive GUI understanding while maintaining its ability to follow instructions effectively.

The fine-tuning is carried out using Qwen3-VL-8B and InternVL3.5-8B, fine-tuned with LoRA (Low-Rank Adaptation) to optimize memory usage and computational efficiency. The fine-tuning experiments are performed on 8 NVIDIA A100 80GB GPUs, with the ViT (Vision Transformer) and aligner modules frozen during training. The LoRA rank is set to 8, and the scaling factor (α) is set to 32. We use a CosineLRScheduler with a learning rate of 1×10^{-4}, a warmup ratio of 0.03 and a total training epochs of 3. The overview of training parameters is shown in Table 5.

11. Limitations

Under-Covered Specialized Domains. Although GUI-HalluBench offers 2,000 bilingual, multi-platform GUI instances across Web, Desktop, and Mobile scenarios, its coverage is still limited with respect to professional and high-stakes application domains. Many specialized interfaces (such as medical diagnostic dashboards, financial trading terminals, or engineering design tools) are not included. The absence of such professional GUI ecosystems may therefore constrain the benchmark's ability to fully evaluate cross-domain robustness and limit its applicability to safety-critical or industry-level deployment contexts.

Forward-Generalization Limitations. The hallucination cases in GUI-HalluBench are constructed based on current GUI layouts and design conventions. While this ensures realism, it also restricts the benchmark's forward compatibility: as UI paradigms evolve (e.g., adaptive layouts, dynamic widgets, new icon metaphors, or AR-based interfaces), the hallucination patterns derived from today's GUIs may lose relevance or introduce systematic biases. Consequently, hallucination-easing strategies proposed in this paper may exhibit decreased effectiveness when confronted with emerging or rapidly shifting interface styles.