UI-Vision: A Desktop-centric GUI Benchmark for Visual Perception and Interaction

Shravan Nayak [*1 2 3] **Xiangru Jian** [*4 3] **Kevin Qinghong Lin** [5] **Juan A. Rodriguez** [1 3 6] **Montek Kalsi** [3]
Rabiul Awal [1 2 3] **Nicolas Chapados** [3] **M. Tamer Özsu** [4] **Aishwarya Agrawal** [1 2 7] **David Vazquez** [3]
Christopher Pal [1 8 3 7] **Perouz Taslakian** [3] **Spandana Gella** [3] **Sai Rajeswar** [3 1 2]

Abstract

Developing autonomous agents that can navigate diverse Graphical User Interfaces (GUIs) and automate complex tasks like document editing and file management is crucial for accelerating computer use workflows. Current research in GUI automation often overlooks desktop environments despite their prevalence, primarily due to licensing restrictions and challenges in data collection. In this work, we introduce **UI-Vision**, *the largest comprehensive desktop-centric and license-permissive GUI benchmark* designed to comprehensively evaluate models in diverse and complex desktop environments. UI-Vision features: (**i**) high-quality bounding boxes and labels for UI elements, along with action trajectories capturing clicks, drags, and keyboard actions from human demonstrations across 83 software applications. (**ii**) Three well-designed GUI tasks – **Element Grounding, Layout Grounding, and Action Prediction** – accompanied by well-defined evaluation metrics for a comprehensive diagnosis of GUI agents. Our evaluation exposes critical limitations in state-of-the-art models like UI-TARS-72B, including poor fine-grained understanding of professional softwares, inaccurate spatial understanding, and difficulty with dragging actions. These findings highlight the considerable progress needed to develop truly autonomous GUI agents capable of visual perception and interaction. Our data and resources will be open-source to support the broader research community.

https://uivision.github.io

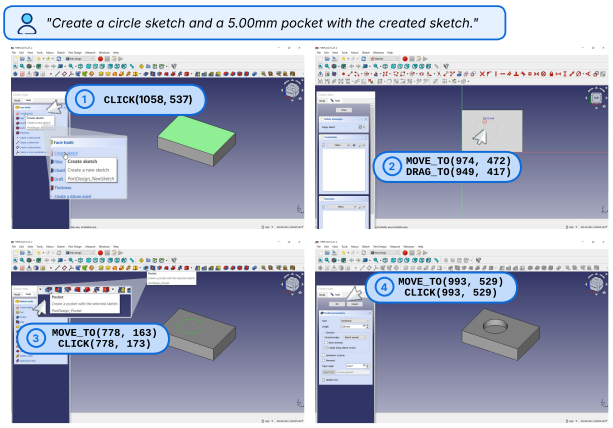


Figure 1: **UI-Vision's Action Prediction task.** Given a task instruction, a screenshot of the current UI state, and a history of previous interactions, the agent must predict the next action necessary to progress toward task completion. We show here an example of a successful episode from a task based on FreeCAD software[1]

1. Introduction

Graphical User Interfaces (GUIs) have become the dominant way users interact with digital worlds, replacing command-line interfaces with visually intuitive environments that enhance usability across desktops, web browsing, and mobile devices (Jansen, 1998). To accelerate digital workflows and assist users, there has been rapid progress in developing intelligent GUI agents—AI systems capable of automating GUI interactions, from simple tasks like auto-filling forms and navigating menus to complex operations such as configuring software settings. Recent advances in Large Language Models (LLMs) have significantly expanded the capabilities of these agents, allowing them to reason and follow natural language instructions (Wei et al., 2022; Yao et al., 2022b; Ouyang et al., 2022; OpenAI, 2021; Schick et al., 2023). However, LLMs that rely only on text struggle with GUI automation, as they lack the ability to interpret visual layouts, spatial relationships, and non-textual UI elements like icons (Gou et al., 2024b). Humans navigate

[*]Equal contribution [1]Mila - Quebec AI Institute [2]Université de Montréal [3]ServiceNow [4]University of Waterloo [5]Show Lab, National University of Singapore [6]École de Technologie Supérieure [7]CIFAR AI Chair [8]Polytechnique Montréal. Correspondence to: Shravan Nayak <shravan.nayak@mila.quebec>, Xiangru Jian <x2jian@uwaterloo.ca>.

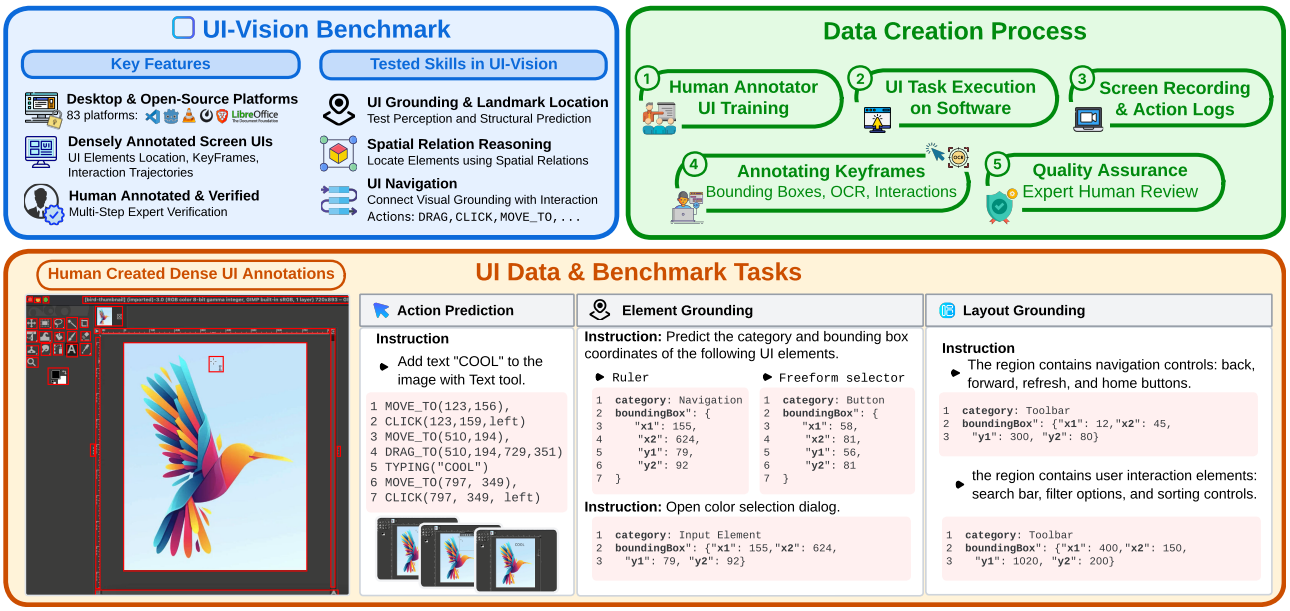


Figure 2: **UI-Vision Benchmark Overview.** Human annotators perform GUI tasks across various desktop software platforms, generating raw trajectories through screen recordings. The collected data undergoes expert verification and is then annotated with multiple layers of information, including keyframe detection, bounding box annotations with OCR, and action logs to capture user interactions.

software visually, recognizing elements by appearance and position. To replicate this, automation must incorporate vision, allowing models to process visual information beyond text. Multimodal LLMs have demonstrated effectiveness in this domain, successfully automating web browsing (Shi et al., 2017; Yao et al., 2022a; Gou et al., 2024a; Lù et al., 2024) and software control (Rodriguez et al., 2023; Li et al., 2023; Hong et al., 2024a; Qin et al., 2025), opening new possibilities for human-computer interaction.

Despite progress in LLM- and multimodal-driven automation, evaluating GUI agents for desktop remains challenging as existing benchmarks primarily focus on web and mobile environments. Web-based benchmarks like MiniWoB++ (Shi et al., 2017), Mind2Web (Deng et al., 2024), and WebArena (Lù et al., 2024; Zhou et al., 2023) assess agent performance using Document Object Model (DOM) structures and HTML metadata. While effective for web interfaces, these approaches fail for desktop GUIs, which lack standardized text-based representations like HTML. Accessibility (A11y) trees provide a semantic alternative but are often inconsistent and incomplete, limiting their reliability for UI understanding (Gou et al., 2024a; Dardouri et al., 2024). Similarly, mobile-focused benchmarks (Li et al., 2020b; Toyama et al., 2021) emphasize touchscreen interactions, making them less applicable to desktop environments where interactions rely on precise mouse control and keyboard interactions. Furthermore, desktop applications lack standardized automation APIs, unlike web automation, which benefits from tools like Selenium [2] and Playwright [3]. This lack of standardization hinders large-scale data collection and automation. Given these gaps, a dedicated desktop-centric benchmark is needed to evaluate models on real-world software variability, visual perception, and open-ended GUI interactions.

To bridge the gap in evaluating GUI agents for desktops, we introduce **UI-Vision**, a benchmark spanning 83 software applications designed to assess GUI agents' ability to perceive and interact with desktop environments. To build UI-Vision, participants design user tasks across these applications that reflect real-world software interactions. They record demonstrations for these tasks and annotate them by labeling UI elements and capturing action sequences, including interactions like clicking, dragging, and typing. Using this data, we define three key tasks that each test a distinct aspect of an agent's ability to navigate and operate within desktop environments. First, **Element Grounding** evaluates how well agents can identify UI elements on screen. Next, **Layout Grounding** assesses the ability to group semantic and functional elements to comprehend overall screen structure. Finally, **Action Prediction** challenges agents to execute interactions like clicking and typing based on planning and UI understanding. Together, these benchmarks provide a structured approach to evaluating GUI agents across desk-

[2] https://www.selenium.dev/
[3] https://playwright.dev/

top interaction tasks. Figure 2 provides an overview of the data creation process. Unlike web and mobile interfaces, these tasks remain largely unexplored in desktop environments. UI-Vision fills this gap by providing a large-scale benchmark with extensive UI element coverage, spanning **83 platforms**, **450 recorded videos**, and **8267 annotated samples**, creating a robust evaluation framework for multimodal GUI agents.

We evaluate state-of-the-art GUI agents on our UI-Vision benchmark. Our evaluation reveals significant gaps across all three tasks. **(i) Element Grounding** remains particularly difficult where even the best-performing model, UI-TARS (Qin et al., 2025), achieves only 25.5% accuracy. This becomes more challenging when an interface is functionality rich and visually complex, with increased number of UI elements. **(iii) Layout Grounding** remains an unsolved problem, with Gemini-1.5-Pro (Team et al., 2024) achieving only 30.8 IoU, indicating that models lack high-level visual comprehension and struggle to recognize structured UI regions. Lastly, **(ii) Action Prediction** is similarly challenging, with Gemini-1.5-Pro (Team et al., 2024) achieving only 13.0% recall on click actions. Moreover, all models struggle with drag actions, exposing weaknesses in handling motion-based interactions. Further, integrating GUI-grounding models like UGround-v1 (Gou et al., 2024a) with planners like Gemini or GPT-4o significantly improves performance on click actions by 2× to 5×, highlighting the strength of planners in structuring actions but their limitations in precise grounding. These findings underscore the need for improved screen parsing, and better integration between planning and grounding mechanisms to enhance GUI perception and interaction.

In summary, our major contributions include:

- We introduce UI-Vision, the largest and most diverse desktop GUI benchmark, spanning 83 real-world environments. It enables a comprehensive evaluation of models across three core tasks for GUI agents: **Element Grounding, Layout Grounding, and Action Prediction**.
- Our dataset provides dense annotations with unmatched UI element coverage, allowing models to be tested on spatial relationships and UI layouts areas often overlooked in prior benchmarks. Built from **open-source and permissive data**, it ensures accessibility and reproducibility.
- We identify major weaknesses in state-of-the-art models across core GUI tasks. In **Element Grounding**, models struggle with fine-grained understanding and fail to accurately interpret spatial relationships. In **Layout Grounding**, models lack proper recognition of UI regions and their significance. Finally, in **Action Execution**, GUI agents fail at precise drag operations that require accurate start and end points.

2. Related Work

2.1. GUI Agents

Recent research highlights the expanding capabilities of large language models (LLMs) beyond conventional linguistic tasks. A key milestone in task-driven GUI navigation is MiniWoB++ (Shi et al., 2017), a web-based environment. Advances in LLM-powered automation (Wei et al., 2022; Yao et al., 2023; Zeng et al., 2023; Yang et al., 2024a; Gao et al., 2023) demonstrate their ability to orchestrate complex workflows autonomously, driving progress in GUI automation. Current methodologies broadly fall into two categories.

Pure Language Agents extract UI metadata from HTML structures, accessibility trees, or vision-language techniques such as OCR (Lee et al., 2023) and spatial-semantic models (Yang et al., 2023; Yan et al., 2023; Zheng et al., 2024; Lu et al., 2024). LLMs then process this structured data to generate task-specific actions. While flexible, these approaches rely on closed-source models (OpenAI, 2023), limiting generalizability, as real-world applications often operate on raw visual inputs rather than structured metadata.

Multimodal Agents attempt to overcome these limitations by using multimodal datasets, pairing images with textual descriptions (Hong et al., 2023; Cheng et al., 2024; You et al., 2024; Li et al., 2024; Gou et al., 2024a; Lin et al., 2024b; Yang et al., 2024b; Wu et al., 2024; Qin et al., 2025). This approach enables pixel-level element grounding and context-aware interface navigation. However, progress remains constrained by the lack of large-scale visual data and standardized benchmarks for real-world UI interactions in desktop environments. Our work fills this gap by introducing a benchmark tailored to desktop applications.

2.2. GUI Benchmarks

Existing benchmarks evaluate different aspects of GUI agent capabilities but remain fragmented in their focus. Element grounding benchmarks (Cheng et al., 2024; Li et al., 2025) test an agent's ability to locate UI elements like icons and text. Action prediction datasets (Deng et al., 2024; Rawles et al., 2023; Zhang et al., 2024) assess task understanding and next-step inference based on screenshots and history. However, layout grounding is often overlooked—current coordinate-based methods (Li et al., 2025) fail to capture structural relationships, limiting spatial reasoning capabilities in GUI tasks (Wu et al., 2023; Rodriguez et al., 2024). Most benchmarks specialize in either grounding (Cheng et al., 2024; Li et al., 2025), navigation (Rawles et al., 2023; Deng et al., 2024), or understanding (Hsiao et al., 2024), lacking a comprehensive evaluation framework. Furthermore, GUI benchmarks are disproportionately skewed towards **Web** (Hao et al., 2011; Wu et al., 2023; Zhou et al., 2023; Koh et al., 2024) and **Mobile** (Deka et al.,

Benchmarks	Environments			Tasks			Statistics			Data Collection	
	Platform	# SW/App	Settings	Ground. (multi.)	Action	Layout	# Sample	Avg. Ele	Avg. Steps	Human	Open License
MiniWoB++ (Shi et al., 2017)	Web	N/A	Online	✗	✓	✗	125	N/A	2.3	N/A	N/A
Mind2Web (Deng et al., 2024)	Web	N/A	Offline	✗	✓	✗	2,350	1	7.3	✓	✓
AITW (Rawles et al., 2023)	Mobile	159	Offline	✗	✓	✗	30,378	1	6.5	✓	N/A
OmniAct (Kapoor et al., 2024)	Desktop, Web	38	Offline	✗	✓	✗	9,802	18.6	1	✓	N/A
OS-World (Xie et al., 2024)	Desktop	8	Online	✗	✓	✗	369	N/A	N/A	✓	✗(windows)
VideoGUI (Lin et al., 2024a)	Desktop	11	Offline	✗	✓	✗	86	N/A	22.7	✓	N/A
ScreenSpot (Cheng et al., 2024)	Desktop, Web, Mobile	N/A	Offline	✓(✗)	✗	✗	1,272	1	N/A	✓	N/A
ScreenSpot-Pro (Li et al., 2025)	Desktop	23	Offline	✓(✗)	✗	✗	1,581	1	N/A	✓	✓
UI-Vision (ours)	Desktop	83	Offline	✓(✓)	✓	✓	8267	71	7.3	✓	✓

Table 1: **Comparison of existing GUI benchmarks with our UI-Vision.** We evaluate their inclusion of desktop platforms, noting a gap in recent works. Additionally, we examine permissive licensing (Open License), highlighting that UI-Vision focuses on open-source platforms. Lastly, we assess the presence of relevant annotations, particularly those related to grounding, text, and OCR. **Multi.** refers to Multi-purpose query for grounding.

2017; Li et al., 2020a;b; Toyama et al., 2021) platforms, primarily due to their structured metadata and accessibility. **Desktop environments, despite their significance in professional workflows, remain underexplored.** Existing desktop benchmarks (Xie et al., 2024; Bonatti et al., 2024) primarily focus on online interactions, while others are constrained by small-scale datasets (Kapoor et al., 2024; Lin et al., 2024a).

Table 1 compares existing benchmarks, highlighting the lack of comprehensive desktop datasets with diverse applications, annotations and real-world complexity. To address these gaps, we introduce **UI-Vision**, the **largest desktop-centric benchmark** to date, spanning **83 software environments** and featuring **rich human annotations**. With 6,484 samples across three tasks and a broad software coverage, UI-Vision establishes a new standard for evaluation in desktop environments, filling a critical gap in the field.

3. UI-Vision

This section introduces UI-Vision, a large-scale benchmark for GUI navigation and visual grounding across 83 desktop applications. We first describe the data collection and annotation in Section 3.1 and in Section 3.2 explain how these annotations are used to construct the UI-Vision benchmark tasks for model evaluation.

3.1. Data Collection

We first collect data from users interacting with desktop software to achieve a goal, capturing their actions and annotating critical elements. This involves recording user interactions, extracting action trajectories that document step-by-step decisions to achieve a goal, and labeling UI components with bounding boxes and functional descriptions. We partner with a data-sourcing company for this process, ensuring a diverse and well-trained annotator pool. Details about the annotators and their qualifications are provided in the Appendix A. Below, we describe the key steps involved in data collection.

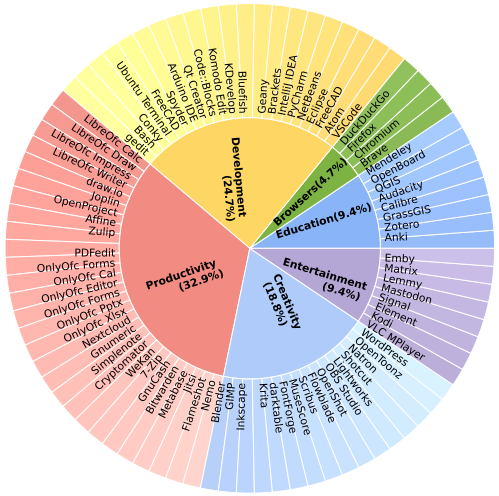


Figure 3: **Distribution of Software Platforms in UI-Vision**. The outer ring displays 83 different software platforms spanning six categories shown in the inner ring.

Selecting Desktop Applications. We curate data from 83 open-source desktop platforms across six domains: Productivity, Development, Creativity, Education, Browsers, and Social Media/Entertainment, ensuring diverse platforms for comprehensive benchmarking and permissive licensing (Figure 3 and Table 6).

Designing User Tasks. User tasks are designed around real-world workflows, ranging from basic actions (e.g., renaming a folder) to complex operations (e.g., applying subtitles to a video). We ensure that each task is well-defined, unambiguous, and comprehensive. Each platform includes 5–7 well-defined user tasks.

Capturing and Annotating User Interactions. Expert annotators perform user tasks while capturing (i) task videos,

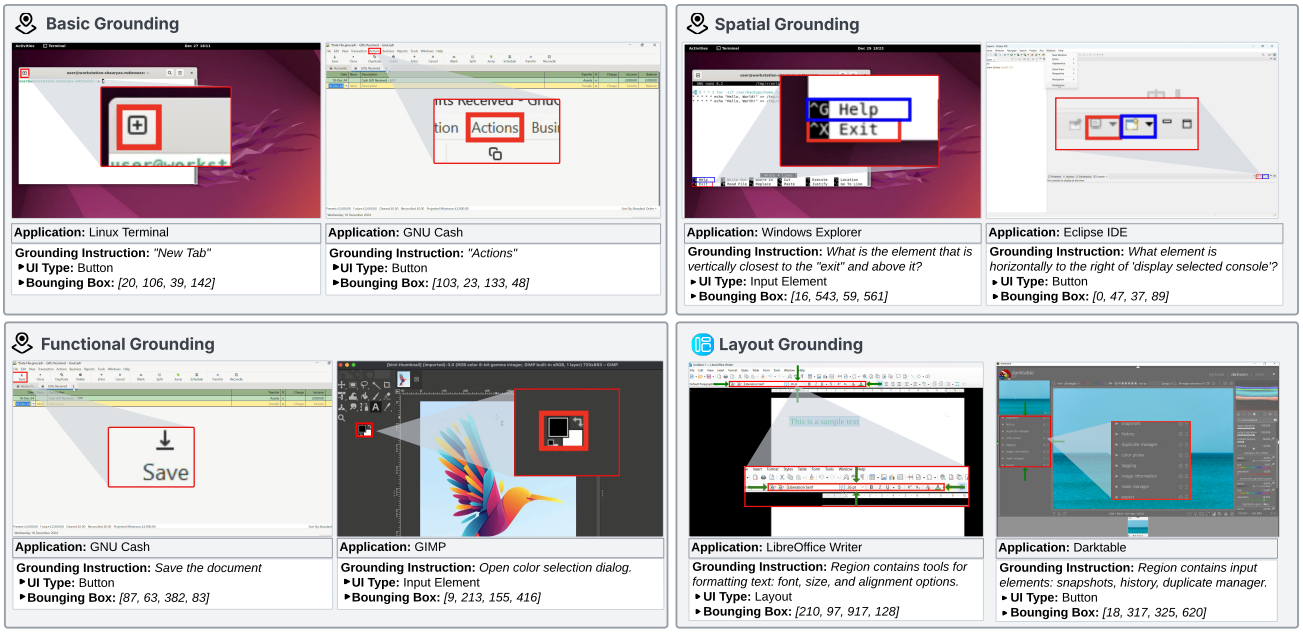


Figure 4: **Examples of Grounding Tasks in UI-Vision.** The dataset features three GUI grounding tasks: **Basic**, involving locating elements' bounding boxes; **Spatial**, focusing on determining positions relative to other elements; and **Functional**, requiring identification of click locations for specific actions. **Layout Grounding** locates larger regions given a description.

(ii) logged actions (11 predefined types; Table 2 in Appendix) with metadata such as mouse click type and frequency and exact timestamp of the action in the video, and (iii) keyframe screenshots, which are screenshots taken just before an action, such as a click, is performed. Post-task, annotators label UI elements with bounding boxes and descriptions using a proprietary tool. A multi-stage quality check by separate annotators and periodic verification by authors ensures completeness, accuracy, and consistency. The final dataset consists of 450 high-quality demonstrations across 83 applications.

Dataset Statistics. In Figure 3, we present the taxonomy and software distribution within UI-Vision, with the largest portions corresponding to 33.7% productivity, 25.3% development, and 19.3% creativity. The key characteristic statistics of the dataset including the distribution of bounding boxes per keyframe, task video durations, and the number of action steps needed for task completion are presented in Figure 13 in the appendix. Most frames contain 5 to 200 bounding boxes, with a mean of 71, indicating a dense annotation setup. The video duration distribution reveals a right-skewed trend, with most videos under 50 seconds, and a mean duration of 37.6 seconds. Finally, the distribution of action steps per user task shows that most user tasks are completed within 25 steps, with a mean of 14 steps highlighting the complexity of the dataset, which require careful planning to execute successfully.

3.2. UI-Vision Benchmark

Building on the rich annotations from the previous section, we focus on three key tasks essential for an agent's capabilities: Element Grounding, to identify and localize UI elements from textual queries; Layout Grounding, to recognize structural relationships and group UI elements into functional clusters; and Action Prediction, to determine the correct interaction needed to achieve a goal. The first two address perception and structural prediction, while the third connects perception to interaction. See Figure 4 for examples of Element and Layout Grounding and Figure 1 for an example on the Action Prediction task.

Task 1: Element Grounding

This task evaluates a model's ability to predict the bounding box of a UI component in a screenshot given a short query (e.g., "Add New Tab"). We leverage the dense bounding box annotations collected during the initial data collection stage for UI-Vision to introduce three grounding subtasks—basic, functional, and spatial—to assess different aspects of GUI understanding beyond simple textual queries. To create these subtasks, we implement a multi-stage process that involves sampling challenging UI elements followed by thorough human evaluation. Details on task creation can be found in Appendix B.1. The **basic** setting evaluates a model's ability to predict the location of a UI element (e.g., button, sidebar, input field) given a minimal textual description, such as "Actions" (Figure 4). The **functional** setting

requires identifying UI elements based on their function rather than direct labels; for instance, instead of "Save Button," the query specifies "Save the current document or data." Functional descriptions are generated using GPT-4o (OpenAI, 2023) and then validated by human reviewers. Both basic and functional grounding subtasks consist of 1,772 query-label pairs. The **spatial** setting assesses a model's ability to locate UI elements based on their spatial relationships with neighboring components. Queries are generated by detecting the nearest bounding boxes in four directions (left, right, top, bottom), e.g., "What is directly above the 'Exit' button?" This setting contains 1,935 query-label pairs.

TASK 2: LAYOUT GROUNDING

GUI layouts define how interface components, such as buttons, text fields, and containers, are arranged into cohesive regions (e.g., "Formatting Tools" or "Main Navigation Bar"). This novel task evaluates a model's ability to cluster UI elements into functional and semantic groups and predict bounding boxes that encapsulate them, a capability not explicitly explored in previous GUI-related benchmarks (Figure 4, bottom right). We again leverage the dense bounding box annotations collected during the initial data collection phase and provide them as input to LLAMA-3.3-70B (Meta, 2024), which generates non-overlapping functional or semantic clusters along with textual descriptions (e.g., "Formatting Tools"). These textual descriptions serve as queries for the model, and the model must predict bounding boxes that encapsulate entire functional or semantic areas. The dataset comprises 311 human verified query-label pairs from 77 platforms, with each functional or semantic cluster containing 5–10 UI elements. More details on the dataset creation can be found in Appendix B.2. This task extends beyond individual element grounding to capture the higher-level structure of a GUI, emphasizing both functionality and semantic organization.

TASK 3: ACTION PREDICTION

Building on the previous perception tasks, action prediction involves solving UI-based goals through a structured sequence of interactions. Unlike static recognition tasks, this agentic task evaluates a model's capability to interpret a given goal (e.g., "Apply a transparency of 45.9% to the layer") and determine the correct sequence of actions.

Action	Description
`Move(x, y)`	Move the mouse to the specified coordinates.
`Click(x, y, button)`	Click the specified button at the given coordinates.
`Typing('Hello')`	Types a specified string.
`Hotkey('ctrl', 'c')`	Performs individual or combination hotkeys.
`Drag([x1,y1], [x2,y2])`	Drags the mouse from start $[x1, y1]$ to end $[x2, y2]$.

Table 2: **Action Prediction task**, action space.

We use the action trajectories collected in the initial data collection stage to create this task. Each step in the trajectory is treated as an independent prediction, with the model receiving the screenshot of current UI state and previous actions as input. More concretely, given a task query $\mathcal{Q}$ and a screenshot $\mathcal{I}i$ at the i-th state, along with the previous action history $\mathcal{H} = \{a_j\}_{j \leq i-1}$, the model is required to output $\tilde{a}_i$. This predicted action is then matched with the ground truth action a_i to calculate scores. Each action a_i includes both the action type and its parameters. The details of creating the task including choosing the right screenshots from videos and processing action trajectories are detailed in Appendix B.3. As a result, we obtain 3231 action-annotation pairs over 442 user tasks. In Table 2, we explicitly define our action spaces and their corresponding parameters.

4. Experiments

4.1. Baselines

We tested a number of open-source VLMs with GUI capabilities, including Qwen2-VL-7B and Qwen2.5VL-7B (Wang et al., 2024), MiniCPM-V-2.6-8B (Yao et al., 2024) and InternVL2-8B (Chen et al., 2024); open source GUI agents, including CogAgent-9B (Hong et al., 2023), SeeClick-9.6B (Cheng et al., 2024), UGround-7B and 72B (Gou et al., 2024b), UI-TARS 7B and 72B (Qin et al., 2025), OSAtlas-7B (Wu et al., 2024), ShowUI-2B (Lin et al., 2024b), Aria-UI-25.3B(3.9B active parameters) (Yang et al., 2024b) and Aguvis-7B (Xu et al., 2024). We also considered closed source models such as GPT-4o (OpenAI, 2024), Claude-3.5-Sonnet and Claude-3.7-Sonnet (Anthropic, 2024), and Gemini-1.5-pro and Gemini-Flash-2.0 (Team et al., 2024). Each model used its recommended format for prompting.

4.2. Evaluation Metrics

Element Grounding. Following prior works (Li et al., 2025; Cheng et al., 2024; Wu et al., 2024), a prediction is considered *correct* if the predicted point (x_i, y_i) falls within the ground-truth bounding box: Correct $= \mathbb{1}(x_{\min} \leq x_i \leq x_{\max} \wedge y_{\min} \leq y_i \leq y_{\max})$. We compute accuracy across all UI components. As prior analysis (Cheng et al., 2024) showed minimal gains from bounding-box-based evaluation, we adopt point-based evaluation throughout.

Layout Grounding. We assess predicted bounding boxes using **Intersection over Union (IoU)**, **precision**, and **recall**, measuring alignment with ground truth.

Action Prediction. We take inspiration from action evaluation metrics used by (Lin et al., 2024a) to develop individual metrics for each action, we next introduce them respectively.

• **`Click & Move:`** We evaluate coordinate predictions $[x, y]$ using two metrics: **Distance (Dist.)** – normalized Euclidean distance between predicted and ground-truth locations: Dist $:= \frac{D}{L}$, where L is the maximum possible

distance to any corner. **Recall@d** – a click is correct if within d pixels of the ground truth. We calculate d as the average bounding box size throughout the dataset.

• **`Drag`:** Evaluated using: **Distance (Dist.)** measures average displacement error: Dist $:= \frac{1}{2}\left(\frac{\Delta s}{L_s} + \frac{\Delta e}{L_e}\right)$. **Recall@$d$** checks if both start and end positions are within d pixels.

• **`Typing & Hotkey`:** Evaluated using **Action Accuracy (Acc.)**, **Action Precision (Prec.)**, and **Correctness (Corr.)**, which verifies exact string or keystroke matches.

We finally calculate the step success rate, 1 when the action is executed correctly. This gives the overall performance of the model across our benchmark

4.3. Results

Performance on Element Grounding. As shown in Table 3, open-source VLMs, such as Qwen-2VL—which forms the backbone for developing effective GUI agents. Building on these models, GUI agents like UI-TARS (25.5) and UGround (23.2) exhibit superior performance across all three tracks, significantly surpassing closed-source API-based models, with the best-performing Claude model scoring 8.3. These findings underscore the challenges posed by UI-Vision's samples. Across the three settings, functional grounding yields scores comparable to those of basic grounding. This indicates that existing VLMs possess sufficient reasoning capabilities to accurately interpret user intent and map queries to relevant elements, suggesting that functionality is not a limiting factor. However, the **key challenge lies in spatial grounding, where the highest score reaches only 14.9%**. This highlights the need to expand the training corpus with more spatial relationships—such as directional references—to enhance grounding capabilities and push the boundaries of performance.

Performance on Layout Grounding. Table 4 gives the summary of evaluation results on Layout Grounding. We evaluate three types of models that can generate bounding box, closed-source/open-sourced VLMs and GUI agents. The key finding is that **VLMs detect UI layout better than GUI agents**. To be specific, closed-source VLMs achieve state-of-the-art performance, as their robust general visual understanding capabilities provide a strong initialization for spatial reasoning and interface element placement. Open-source VLMs like Qwen-2VL and MiniCPM-V-8B also exhibit performance somehow comparable to closed-source models. But for GUI agents, both CogAgent-9B and SeeClick-9.6B demonstrate subpar performance. While OSAtlas-7B is the exception, it is trained upon Qwen-2VL and hasn't shown any significant improvement compared to the latter. This parity suggests that GUI agent frameworks primarily prioritize element grounding tasks (e.g., refining click coordinate accuracy) while achieving less significant advancements in layout grounding and bounding box prediction.

Performance on Action Prediction. Table 5 summarizes the results of the Action Prediction benchmark. We evaluate two baselines: a random baseline and GPT-4o without images, to analyze the impact of action history on model performance. Additionally, we assess various closed-source VLMs, including GPT-4o, Gemini, and Claude, alongside open-source agentic VLMs such as ShowUI and UI-TARS. Finally, we explore whether pairing action planning abilities of VLMs GPT-4o and Claude with grounding models enhances their accuracy in click actions. The key findings are as follows:

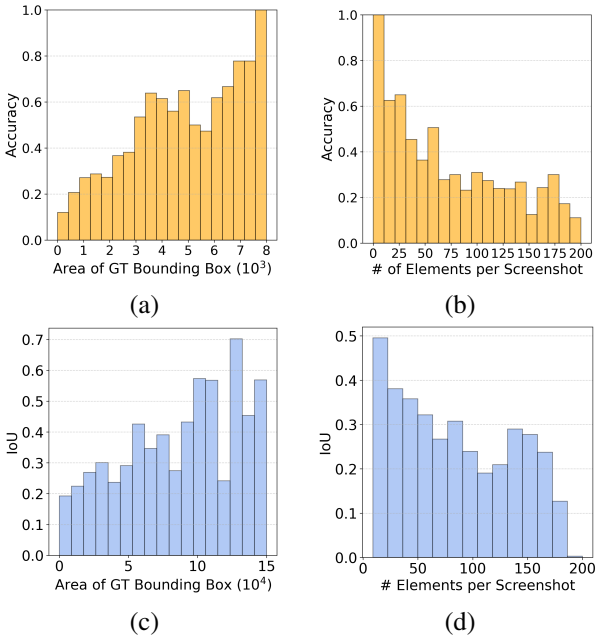


Figure 5: Analysis on Element Grounding (UI-TARS-72B under **Basic setting**) in terms of (a) Area of GT. Bbox and (b) # Ele. per screenshot. ; Layout Grounding (Gemini-pro-1.5) in terms of (c) Area of GT. Bbox and (d) # Ele. per screenshot. To enhance clarity, we do not display long-tail examples below 5%.

Action history impacts model performance. We provide only the task and the sequence of actions taken in previous steps as input to GPT-4o and ask it to predict the next action. The results show that the model significantly outperforms the random baseline, indicating that action history offers valuable contextual feedback.

Closed source models fail at grounding actions on the right elements. Table 5 shows that while models like GPT-4o, Gemini, and Claude achieve good accuracy and precision for click actions, they struggle with grounding these

Model	Basic							Functional							Spatial							Final Avg
	Ed (215)	Br (56)	De (376)	Pr (605)	Cr (438)	En (82)	Overall (1772)	Ed (215)	Br (56)	De (376)	Pr (605)	Cr (438)	En (82)	Overall (1772)	Ed (212)	Br (31)	De (338)	Pr (740)	Cr (586)	En (28)	Overall (1935)	
Closed-Source VLMs																						
GPT-4o (OpenAI, 2024)	2.23	0.00	1.86	1.16	1.14	4.88	1.58	1.40	0.00	3.19	0.83	0.91	3.66	1.52	0.94	0.00	1.48	1.22	0.51	3.57	1.03	1.38
Gemini-1.5-pro (Team et al., 2023)	0.47	0.00	1.60	0.83	0.46	0.00	0.79	0.00	1.79	0.27	0.17	0.46	0.00	0.28	0.94	0.00	0.89	0.54	0.34	0.00	0.57	0.55
Gemini-Flash-2.0 (Team et al., 2023)	0.00	0.00	0.27	0.66	0.68	0.00	0.45	0.47	1.79	0.00	0.66	0.23	0.00	0.40	0.00	0.00	0.00	0.14	0.00	0.00	0.05	0.30
Claude-3.5-Sonnet (Anthropic, 2024)	3.26	16.1	5.32	6.94	1.83	4.88	5.08	5.12	19.6	4.79	5.95	2.51	4.88	5.19	2.83	9.68	5.03	2.43	2.56	7.14	3.15	4.47
Claude-3.7-Sonnet (Anthropic, 2024)	**6.51**	12.5	**7.98**	**11.24**	**9.13**	**11.0**	**9.48**	5.12	7.14	**8.24**	**9.92**	**6.16**	**4.88**	**7.73**	**6.60**	**9.68**	**7.69**	**7.43**	**7.85**	**10.7**	**7.60**	**8.27**
Open-Source VLMs																						
Qwen-2.5VL-7B (Wang et al., 2024)	0.47	0.00	1.33	1.65	0.68	1.22	1.24	0.47	0.00	0.80	1.16	0.46	1.22	0.79	0.47	0.00	1.48	0.00	0.51	0.00	0.51	0.85
InternVL2-8B (Chen et al., 2024)	0.00	0.00	0.00	0.017	0.00	0.14	0.11	0.00	0.00	0.27	0.00	0.00	1.22	0.11	0.00	0.00	0.00	0.14	0.00	0.00	0.05	0.09
Qwen-2VL-7B (Wang et al., 2024)	2.79	7.14	3.72	3.97	0.68	12.2	3.44	2.79	12.5	3.19	3.97	0.68	6.10	3.22	0.47	**3.23**	**2.37**	**1.08**	0.51	3.57	1.45	2.70
MiniCPM-V-8B (Yao et al., 2024)	**4.19**	**21.4**	**7.71**	**7.44**	**3.65**	**18.3**	**7.11**	**4.19**	**19.6**	**6.38**	**4.63**	**2.97**	**11.0**	**5.30**	**0.47**	3.23	1.78	0.27	0.17	**3.57**	**1.45**	**4.34**
Open-Source GUI Agents																						
AriaUI-25.3B (Yang et al., 2024b)	10.7	23.2	13.0	12.9	8.22	20.7	12.2	12.6	19.6	15.4	14.6	10.5	22.0	14.0	3.77	9.68	4.44	4.86	2.22	7.14	3.98	10.1
UGround-v1-7B (Gou et al., 2024a)	11.6	35.7	19.7	15.0	11.0	18.3	15.4	15.4	33.9	22.3	16.5	11.6	19.5	17.1	4.25	6.45	9.76	6.35	4.44	14.3	6.25	12.9
OSAtlas-7B (Wu et al., 2024)	10.7	23.2	13.3	12.6	8.22	22.0	12.2	11.6	16.1	11.4	12.7	7.5	13.4	11.2	3.77	6.45	5.62	3.65	2.22	7.14	3.67	9.02
UGround-7B (Gou et al., 2024a)	10.2	23.2	14.9	10.6	7.53	19.5	11.5	12.1	25.0	15.2	11.2	7.99	20.7	12.2	2.36	0.00	4.14	2.70	2.22	7.14	2.79	8.83
Aguvis-7B (Xu et al., 2024)	**16.7**	37.5	**22.3**	16.2	12.6	26.8	17.8	17.2	35.7	21.5	18.0	13.0	24.4	18.3	5.19	9.68	6.51	4.05	4.78	14.3	5.06	13.7
UI-TARS-7B (Qin et al., 2025)	15.4	**41.1**	21.8	**21.2**	**13.2**	**39.0**	**20.1**	**20.5**	**41.1**	**25.5**	**26.5**	**16.0**	**45.1**	**24.3**	**6.60**	**12.9**	**11.0**	**9.2**	**5.8**	**17.9**	**8.37**	**17.6**
CogAgent-9B (Hong et al., 2024b)	11.2	14.3	12.5	13.7	8.68	15.9	12.0	11.6	14.3	11.4	14.7	8.22	18.3	12.2	3.30	0.00	1.18	4.05	1.37	7.14	2.63	8.94
SeeClick-9.6B (Cheng et al., 2024)	7.44	23.2	13.0	8.43	5.48	17.1	9.42	4.65	7.14	5.32	3.97	4.34	7.32	4.68	0.47	6.45	3.25	1.22	2.73	3.57	2.07	5.39
UGround-v1-72B (Gou et al., 2024a)	27.0	42.9	31.7	26.6	22.8	40.2	27.9	25.1	33.9	30.6	26.6	21.0	40.2	26.7	**15.1**	**25.8**	**19.8**	13.4	12.8	25.0	**14.9**	23.2
UI-TARS-72B (Qin et al., 2025)	**30.7**	**48.2**	**32.7**	**33.6**	**21.9**	**51.2**	**31.4**	**29.8**	**46.4**	**30.9**	**34.1**	**22.6**	**36.6**	**30.5**	13.7	16.1	19.2	**15.4**	**11.1**	**25.0**	14.7	**25.5**

Table 3: Element Grounding results by category for Basic, Functional, and Spatial subtasks. For each subtask, the first six columns (Ed, Br, De, Pr, Cr, En) present the fine-grained breakdown, followed by an overall score column. The number of samples in each category is noted under the column name. The final column shows the overall average. Abbreviated category labels: Ed (Education), Br (Browsers), De (Development), Pr (Productivity), Cr (Creativity), En (Entertainment). The best model within each size category is highlighted in **bold**, and the runner-up is underlined. Models are categorized by size: gray marks closed-source models, green indicates open-source VLM models, blue represents open-source GUI Agent models below 8B parameters, orange denotes open-source GUI Agent models above 8B parameters.

Model	Layout Grounding IoU↑	Precision↑	Recall↑
Closed-Source VLMs			
GPT-4o (OpenAI, 2024)	20.0	59.6	24.1
Claude-3.5-Sonnet (Anthropic, 2024)	22.4	64.3	26.8
Claude-3.7-Sonnet (Anthropic, 2024)	17.6	31.5	34.1
Gemini-1.5-pro (Team et al., 2023)	**30.8**	**67.8**	36.9
Gemini-2.0-flash (Team et al., 2023)	28.3	63.0	34.2
Open-Source VLMs			
Qwen-2VL-7B (Wang et al., 2024)	24.3	65.7	33.4
MiniCPM-V-8B (Yao et al., 2024)	16.3	25.7	**43.6**
Open-Source GUI Agents			
CogAgent-9B (Hong et al., 2024b)	6.22	7.99	42.9
SeeClick-9.6B (Cheng et al., 2024)	5.11	6.32	30.1
OSAtlas-7B (Wu et al., 2024)	28.2	66.4	41.6

Table 4: **UI-Vision Leaderboard on Layout Grounding**. We present the performance results of state-of-the-art UI understanding models that can predict bounding boxes. Results are averaged across IoU, precision, and recall.

actions on the right UI elements, as reflected in their low recall. This indicates they often fail to precisely locate the target on the screen. Additionally, all models demonstrate significant difficulty in predicting drag actions, suggesting that continuous interactions pose a greater challenge than discrete clicks.

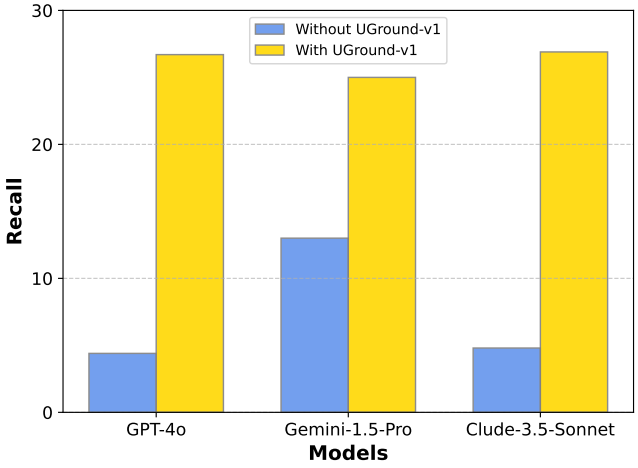


Figure 6: Recall improvement for Click action for different models with and without the UGround-v1 planner.

Closed source models are good planners. We prompt closed-source models to generate a step-by-step plan for the next action, describing it in words and specifying the elements to interact with. Using Uground-v1 to map these planned actions onto the screen results in a 2× to 5× im-

Model	Click / Move				Drag				Typing			Hotkey		
	Dist. ↓	Recall@d ↑	Prec. ↑	Acc. ↑	Dist. ↓	Recall@d ↑	Prec. ↑	Acc. ↑	Corr. ↑	Prec. ↑	Acc. ↑	Corr. ↑	Prec. ↑	Acc. ↑
Navie Baselines														
Random	81.6	0.0	35.2	19.4	94.2	0.0	10.6	23.2	N/A	9.76	20.7	N/A	7.85	17.5
GPT-4o (w/o image)	52.0	3.3	71.8	81.7	72.4	0.0	38.0	13.2	22.7	30.2	30.9	34.0	44.2	22.4
Closed-Source VLMs														
GPT-4o	41.2	4.4	77.1	85.8	63.9	1.5	45.1	20.5	32.1	44.1	63.2	56.5	65.2	15.2
Gemini-1.5-Pro-Vision	38.7	13.0	77.4	82.1	**61.1**	**1.6**	**49.2**	22.3	24.7	39.9	**73.4**	45.3	70.3	16.4
Claude-3.5-Sonnet	41.0	4.8	**78.6**	83.3	61.4	1.1	48.8	**30.2**	29.0	44.4	65.0	39.2	56.2	**25.3**
Open-Source VLMs														
ShowUI-2B	42.8	11.8	74.4	80.8	N/A	N/A	0.0	0.0	15.2	25.7	58.9	**62.5**	**75.0**	2.0
UI-TARS-7B	47.0	19.7	72.8	**91.3**	64.8	3.1	41.5	9.4	**33.8**	**51.7**	38.9	40.5	50.0	6.5

Table 5: **UI-Vision Action Prediction Leaderboard**. We compare the performance of three model categories: Closed-Source VLMs, Open-Source VLMs, and Agentic solutions, across four action types: Click/Move, Drag, Typing, and Hotkey. The evaluation metrics include Distance (Dist.), Recall@d, Precision (Prec.), and Accuracy (Acc.) for all actions, with Correctness (Corr.) additionally reported for Typing and Hotkey.

provement in recall. This suggests that while these models excel at planning actions, their ability to directly ground them remains a challenge.

Open Source Agentic VLMs exhibit biases. Although agentic VLMs demonstrate stronger grounding capabilities than closed-source counterparts, they often struggle to follow instructions consistently. This may be due to extensive training on a specific instruction format, leading to rigid behavior. For instance, ShowUI fails to generate drag actions, while UI-TARS occasionally produces `wait()` and `finish()` actions, even when these were not included in the prompt.

4.4. Ablation Studies

Grounding Analysis. Figure 5 (a-b) presents the relationship between accuracy and two key factors in element grounding: (a) the area of the ground truth (GT) bounding box and (b) the number of UI elements per Screenshot. In (a), accuracy increases with the GT bounding box area, indicating that larger elements are easier to predict, while smaller ones remain challenging. In (b), accuracy decreases dramatically when there are more elements in the screenshot in question, indicating scenario with dense layout of elements will pose a huge difficulty to grounding. While it is commonly assumed that screenshot resolution significantly impacts grounding accuracy, our analysis (see Appendix E.1) reveals that it is not a key factor empirically.

Layout Analysis. Figure 5 (c-d) illustrates the relationship between IoU and two key factors in layout generation: (c) the area of the GT bounding box and (d) the number of elements per screenshot. In (c), IoU increases with GT bounding box area, aligning with the grounding results, as larger elements are easier to localize, while smaller ones pose challenges. Conversely, in (d), IoU decreases as element density rises, suggesting that complex and cluttered UI layouts hinder precise layout generation. These findings highlight that *localizing small elements within dense UI layouts* remains the primary bottleneck in layout generation.

5. Conclusion

We present UI-Vision, a large-scale GUI benchmark covering 83 desktop applications, making it one of the most extensive and diverse dataset of its kind. We use UI-Vision to build benchmarks supporting three structured evaluation tasks: (i) Element Grounding, which measures a model's ability to identify and localize UI elements based on textual queries; (ii) Layout Grounding, which evaluates how effectively a model clusters UI elements into functional groups and defines bounding boxes around them; and (iii) Action Prediction, which tests an agent's ability to predict the correct action required to complete a task. Our experiments with state-of-the-art VLMs highlight significant challenges, such as the difficulty in grounding UI elements and predicting actions accurately. The results underscore key gaps in current model capabilities, emphasizing the need for further research in UI interaction modeling for desktops. To facilitate progress in this domain, we will release UI-Vision to the research community.

6. Impact Statement

This research benchmark and analyses offers contributions that are crucial for the advancement of visual-centric UI assistants, specifically targeting agents for Desktop-based software and their applications. The rise of these agents promises to revolutionize how humans engage with digital software and transform the execution of digital tasks by workers. In the following sections, we will discuss both the potential positive and negative societal impacts stemming from this research.

Positive Impacts: Enabling GUI agents to automate repetitive and complex desktop tasks allows users to achieve

higher productivity and focus on more creative and strategic activities. With more studies and capable GUI agents, software developers can easily finish unit-tests and collect feedback to develop better software or an interface. UI-Vision also provides a standardized desktop benchmark to drive innovation in GUI automation and advance the development of GUI models.

Potential Negative Impacts: While our work advances GUI automation, it also presents several challenges. First, protecting user privacy is crucial when developing, evaluating or deploying GUI agents on personal devices, as these systems may inadvertently expose sensitive information. Second, the high computational cost of vision-language models (VLMs) during inference raises significant environmental concerns due to increased energy consumption, which must be addressed for sustainable deployment. Lastly, reliance on highly capable GUI agents may lead to user over-dependence, diminishing manual proficiency and critical problem-solving skills over time.

7. Acknowledgments

Authors would like to sincerely thank Ghazwa Darwiche, Christian Hudon, Tom Murray, Pierre-Andre, Chao Wang, and Fanny Rancourt for their valuable administrative and technical support. Furthermore, we acknowledge MITACS for supporting the recruitment of visiting student researchers who played a significant role in this project.

References

Anthropic, A. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 2024.

Bonatti, R., Zhao, D., Bonacci, F., Dupont, D., Abdali, S., Li, Y., Lu, Y., Wagle, J., Koishida, K., Bucker, A., et al. Windows agent arena: Evaluating multi-modal os agents at scale. *arXiv preprint arXiv:2409.08264*, 2024.

Chen, Z., Wang, W., Tian, H., Ye, S., Gao, Z., Cui, E., Tong, W., Hu, K., Luo, J., Ma, Z., Ma, J., Wang, J., Dong, X., Yan, H., Guo, H., He, C., Shi, B., Jin, Z., Xu, C., Wang, B., Wei, X., Li, W., Zhang, W., Zhang, B., Cai, P., Wen, L., Yan, X., Dou, M., Lu, L., Zhu, X., Lu, T., Lin, D., Qiao, Y., Dai, J., and Wang, W. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites, 2024. URL https://arxiv.org/abs/2404.16821.

Cheng, K., Sun, Q., Chu, Y., Xu, F., Li, Y., Zhang, J., and Wu, Z. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*, 2024.

Dardouri, T., Minkova, L., Espejel, J. L., Dahhane, W., and Ettifouri, E. H. Visual grounding for desktop graphical user interfaces. *arXiv preprint arXiv:2407.01558*, 2024.

Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., and Kumar, R. 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*, pp. 845–854, 2017.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 2024.

Gao, D., Ji, L., Zhou, L., Lin, K. Q., Chen, J., Fan, Z., and Shou, M. Z. Assistgpt: A general multi-modal assistant that can plan, execute, inspect, and learn. *arXiv preprint arXiv:2306.08640*, 2023.

Gou, B., Wang, R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., and Su, Y. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*, 2024a. URL https://arxiv.org/abs/2410.05243.

Gou, B., Wang, R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., and Su, Y. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*, 2024b.

Hao, Q., Cai, R., Pang, Y., and Zhang, L. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 775–784, 2011.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and Schmidhuber, J. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=VtmBAGCN7o.

Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., et al. Cogagent: A visual language model for gui agents. *arXiv preprint arXiv:2312.08914*, 2023.

Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024b.

Hsiao, Y.-C., Zubach, F., Wang, M., and Chen, J. Screenqa: Large-scale question-answer pairs over mobile app screenshots, 2024.

Jansen, B. J. The graphical user interface. *ACM SIGCHI Bull.*, 30:22–26, 1998. URL https://api.semanticscholar.org/CorpusID:18416305.

Kapoor, R., Butala, Y. P., Russak, M., Koh, J. Y., Kamble, K., Alshikh, W., and Salakhutdinov, R. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. *arXiv preprint arXiv:2402.17553*, 2024.

Koh, J. Y., Lo, R., Jang, L., Duvvur, V., Lim, M. C., Huang, P.-Y., Neubig, G., Zhou, S., Salakhutdinov, R., and Fried, D. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.

Lee, K., Joshi, M., Turc, I. R., Hu, H., Liu, F., Eisenschlos, J. M., Khandelwal, U., Shaw, P., Chang, M.-W., and Toutanova, K. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pp. 18893–18912. PMLR, 2023.

Li, H., Su, J., Chen, Y., Li, Q., and ZHANG, Z.-X. Sheetcopilot: Bringing software productivity to the next level through large language models. In *Advances in Neural Information Processing Systems*, pp. 4952–4984. Curran Associates, Inc., 2023.

Li, K., Meng, Z., Lin, H., Luo, Z., Tian, Y., Ma, J., Huang, Z., and Chua, T.-S. Screenspot-pro: Gui grounding for professional high-resolution computer use, 2025.

Li, Y., He, J., Zhou, X., Zhang, Y., and Baldridge, J. Mapping natural language instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*, 2020a.

Li, Y., Li, G., He, L., Zheng, J., Li, H., and Guan, Z. Widget captioning: Generating natural language description for mobile user interface elements. *arXiv preprint arXiv:2010.04295*, 2020b.

Li, Z., You, K., Zhang, H., Feng, D., Agrawal, H., Li, X., Moorthy, M. P. S., Nichols, J., Yang, Y., and Gan, Z. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967*, 2024.

Lin, K. Q., Li, L., Gao, D., Wu, Q., Yan, M., Yang, Z., Wang, L., and Shou, M. Z. Videogui: A benchmark for gui automation from instructional videos. *arXiv preprint arXiv:2406.10227*, 2024a.

Lin, K. Q., Li, L., Gao, D., Yang, Z., Wu, S., Bai, Z., Lei, W., Wang, L., and Shou, M. Z. Showui: One vision-language-action model for gui visual agent, 2024b. URL https://arxiv.org/abs/2411.17465.

Lù, X. H., Kasner, Z., and Reddy, S. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*, 2024.

Lu, Y., Yang, J., Shen, Y., and Awadallah, A. Omniparser for pure vision based gui agent, 2024. URL https://arxiv.org/abs/2408.00203.

Meta. Introducing meta llama 3: The most capable openly available llm to date, 2024. URL https://ai.meta.com/blog/meta-llama-3/. Accessed: 2024-04-18.

OpenAI. Introducing chatgpt. OpenAI Blog, 09 2021. URL https://openai.com/blog/chatgpt.

OpenAI. Gpt-4 technical report, 2023.

OpenAI. Hello gpt-4o, May 2024. URL https://openai.com/index/hello-gpt-4o/. Accessed: 2024-06-06.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Qin, Y., Ye, Y., Fang, J., Wang, H., Liang, S., Tian, S., Zhang, J., Li, J., Li, Y., Huang, S., Zhong, W., Li, K., Yang, J., Miao, Y., Lin, W., Liu, L., Jiang, X., Ma, Q., Li, J., Xiao, X., Cai, K., Li, C., Zheng, Y., Jin, C., Li, C., Zhou, X., Wang, M., Chen, H., Li, Z., Yang, H., Liu, H., Lin, F., Peng, T., Liu, X., and Shi, G. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025. URL https://github.com/bytedance/UI-TARS.

Rawles, C., Li, A., Rodriguez, D., Riva, O., and Lillicrap, T. Android in the wild: A large-scale dataset for android device control. *arXiv preprint arXiv:2307.10088*, 2023.

Rodriguez, J., Jian, X., Panigrahi, S. S., Zhang, T., Feizi, A., Puri, A., Kalkunte, A., Savard, F., Masry, A., Nayak, S., et al. Bigdocs: An open and permissively-licensed dataset for training multimodal models on document and code tasks. *arXiv preprint arXiv:2412.04626*, 2024.

Rodriguez, J. A., Agarwal, S., Laradji, I. H., Rodriguez, P., Vazquez, D., Pal, C., and Pedersoli, M. Starvector: Generating scalable vector graphics code from images. *arXiv preprint arXiv:2312.11556*, 2023.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Shi, T., Karpathy, A., Fan, L., Hernandez, J., and Liang, P. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pp. 3135–3144. PMLR, 2017.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., Mariooryad, S., Ding, Y., Geng, X., Alcober, F., Frostig, R., Omernick, M., Walker, L., Paduraru, C., Sorokin, C., Tacchetti, A., Gaffney, C., Daruki, S., Sercinoglu, O., Gleicher, Z., and Others. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Toyama, D., Hamel, P., Gergely, A., Comanici, G., Glaese, A., Ahmed, Z., Jackson, T., Mourad, S., and Precup, D. Androidenv: A reinforcement learning platform for android. *arXiv preprint arXiv:2105.13231*, 2021.

Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., Fan, Y., Dang, K., Du, M., Ren, X., Men, R., Liu, D., Zhou, C., Zhou, J., and Lin, J. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wu, J., Wang, S., Shen, S., Peng, Y.-H., Nichols, J., and Bigham, J. P. Webui: A dataset for enhancing visual ui understanding with web semantics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2023.

Wu, P. and Xie, S. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13084–13094, 2024.

Wu, Z., Wu, Z., Xu, F., Wang, Y., Sun, Q., Jia, C., Cheng, K., Ding, Z., Chen, L., Liang, P. P., et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., et al. Os-world: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

Xu, Y., Wang, Z., Wang, J., Lu, D., Xie, T., Saha, A., Sahoo, D., Yu, T., and Xiong, C. Aguvis: Unified pure vision agents for autonomous gui interaction, 2024. URL https://arxiv.org/abs/2412.04454.

Yan, A., Yang, Z., Zhu, W., Lin, K., Li, L., Wang, J., Yang, J., Zhong, Y., McAuley, J., Gao, J., et al. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.

Yang, J., Zhang, H., Li, F., Zou, X., Li, C., and Gao, J. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023. URL https://arxiv.org/abs/2310.11441.

Yang, R., Song, L., Li, Y., Zhao, S., Ge, Y., Li, X., and Shan, Y. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36, 2024a.

Yang, Y., Wang, Y., Li, D., Luo, Z., Chen, B., Huang, C., and Li, J. Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*, 2024b.

Yao, S., Chen, H., Yang, J., and Narasimhan, K. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yao, Y., Yu, T., Zhang, A., Wang, C., Cui, J., Zhu, H., Cai, T., Li, H., Zhao, W., He, Z., et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

You, K., Zhang, H., Schoop, E., Weers, F., Swearngin, A., Nichols, J., Yang, Y., and Gan, Z. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *arXiv preprint arXiv:2404.05719*, 2024.

Zeng, A., Attarian, M., brian ichter, Choromanski, K. M., Wong, A., Welker, S., Tombari, F., Purohit, A., Ryoo, M. S., Sindhwani, V., Lee, J., Vanhoucke, V., and Florence, P. Socratic models: Composing zero-shot multimodal reasoning with language. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=G2Q2Mh3avow.

Zhang, J., Wu, J., Teng, Y., Liao, M., Xu, N., Xiao, X., Wei, Z., and Tang, D. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024.

Zheng, B., Gou, B., Kil, J., Sun, H., and Su, Y. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=piecKJ2DlB.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

Appendix

Table of Contents

	Page
A. Human Annotation	15
B. UI-Vision Benchmark Tasks	15
B.1 Element Grounding	15
B.2 Layout Grounding	17
B.3 Action Prediction	18
C. Dataset Statistics and Examples	19
D. Error Analysis	20
D.1 Element Grounding	20
D.2 Layout Understanding	21
D.3 Action Prediction	21
E. Limitations and Future Work	21

A. Human Annotation

We partnered with Turing[4] (referred to as the "Vendor"), a data labeling company that specializes in data curation for AI applications. The annotation process spanned roughly over three three-month periods, with the initial month dedicated to a pilot phase. During this pilot period, we worked closely with the vendor annotation team conducted detailed reviews, and provided extensive feedback to help annotators better grasp the task requirements.

The Vendor annotation team consisted of 70 annotators. This included a multi-tiered team comprising annotators, quality assurance specialists, and managers. The majority of the team was based in India and Latin America. The annotators were in the 20–35 year-old age group, with an equal distribution of male and female participants. They all had strong proficiency in technical writing and English. All of the annotators hold bachelor's degrees in Engineering, Computer Science, and related disciplines. Annotators also had prior experience in data labeling and UI research.

Category	Software Applications
Education	Anki, Zotero, GrassGIS, Calibre, Audacity, QGIS, OpenBoard, Mendeley
Browsers	Brave, Chromium, Mozilla Firefox, DuckDuckGo
Development	VSCode, Atom, FreeCAD, Eclipse, NetBeans, PyCharm, IntelliJ IDEA, Brackets, Geany, Bluefish, KDevelop, Komodo Edit, Code::Blocks, Qt Creator, Arduino IDE, Spyder, Ubuntu Terminal, Conky, Bash, gedit
Productivity	LibreOffice Calc, LibreOffice Draw, LibreOffice Impress, LibreOffice Writer, draw.io, Joplin, OpenProject, Affine, Zulip, PDFedit, OnlyOffice Calendar, OnlyOffice Document Editor, OnlyOffice Forms, OnlyOffice PDF Forms, OnlyOffice Presentation, OnlyOffice Spreadsheet, Nextcloud, Gnumeric, Simplenote, Cryptomator, WeKan, 7-Zip, GnuCash, Bitwarden, Metabase, Jitsi, Flameshot, Nemo
Creativity	Blender, GIMP, Inkscape, Krita, darktable, FontForge, MuseScore, Scribus, OpenShot, OBS Studio, Lightworks, Shotcut, Natron, OpenToonz, WordPress
Entertainment	VLC Media Player, Kodi, Element, Signal, Mastodon, Lemmy, Matrix, Emby

Table 6: Mapping from coarse-grain categories to platforms. This table lists the categories and their corresponding platforms used in our study.

To ensure high-quality annotations, annotators underwent a training process to familiarize themselves with the platforms and the applications. Annotators were paid per hour, and each task took an average of 60 to 90 minutes to complete, from task creation to quality check. The annotators started with creating user tasks for 83 different software applications highlighted in Table 6. These user tasks were verified by quality assurance specialists and the authors to ensure diversity and coverage. This was followed by the execution of user tasks and screen recording. A proprietary tool captured action trajectories and logged the precise coordinates of actions as the annotators performed the task. Finally, the annotators used a custom annotation tool to capture screenshots (keyframes) of each user interaction and refine the action trajectories. Each of the captured keyframes was annotated by drawing bounding boxes around all interface elements visible on the screen. The entire process underwent rigorous quality assurance, including review by human annotators and custom annotation evaluation scripts.

B. UI-Vision Benchmark Tasks

B.1. Element Grounding

To create the high-quality benchmark, we implement a systematic data workflow to ensure accuracy, diversity, and reliability for evaluating GUI models. The following steps outline this process:

We start with the bounding box annotations obtained during the initial data collection, as highlighted in Section 3.1. Figure 14 illustrates an example screenshot with annotations. Given the large number of annotations, we sample a subset for the benchmark to ensure comprehensive platform coverage while maintaining ease of evaluation.

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

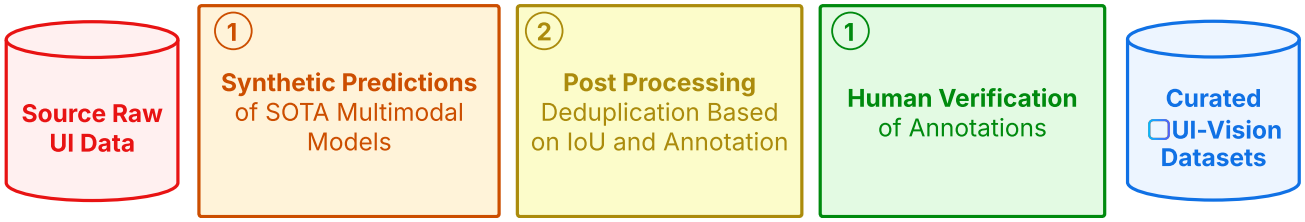


Figure 7: **UI-Vision's Data Curation Workflow.** Step-by-step process from sourcing raw UI data to creating curated datasets. The workflow includes synthetic predictions using state-of-the-art (SOTA) multimodal models, post-processing for deduplication, and human verification to ensure high-quality annotations.

Our analysis of state-of-the-art models like UI-TARS (Qin et al., 2025) reveals that they perform well on naive data samples, particularly those with textual information, but struggle significantly with icon-based elements and smaller UI components. Additionally, models show lower performance on platforms with a high bounding box ratio, such as those in the creativity category like Blender and GIMP. To address these issues, we sample bounding boxes from different image resolutions and sizes, with a focus on smaller screen elements.

To maximize diversity, we apply an Intersection over Union (IoU) threshold within each platform, filtering out redundant samples with similar functionality or spatial positioning. This ensures that the selected samples vary in function and placement within the UI. After the initial sampling, we use an ensemble of four top-performing GUI models (e.g., (Qin et al., 2025; Gou et al., 2024b; Yang et al., 2024b)) to identify additional challenging cases. Samples are classified as *hard* (where all models fail or predictions disagree) or *easy* (where at least one model provides a correct prediction), ensuring a balanced difficulty distribution. Reviewers verify uncertain instances by consulting documentation or online sources.

To expand the evaluation beyond basic queries, we introduce variations to assess model capabilities in **functional** and **spatial** settings. This approach allows us to reuse existing annotations while benchmarking different aspects of GUI understanding.

For the **functional** setting, we use each element's annotation and corresponding screenshot (with the bounding box highlighted) as input to GPT-4o to generate function-based descriptions. These descriptions reflect how the element is used rather than just its label.

Prompt used with GPT-4o to generate annotation for functional setting

```
You are an assistant tasked with generating a single, precise functional
annotation for a UI element based on its basic textual description.  Your job
is to:
1.  Ensure the functional annotation is accurate, straightforward, and
directly aligns with the provided description and bounding box context before
adding any complexity or diversity.
2.  Focus on clarity and correctness first, avoiding ambiguous or overly
complex annotations.
3.  Use the textual description and bounding box context to infer the most
likely purpose or role of the UI element.
4.  Try best not to directly copy the textual description, but rather provide
a functional interpretation based on the context.

Below is the input for generating a functional annotation:
1.  Screenshot:  Please refer to the provided image.  The UI element is
highlighted by a red bounding box.
2.  Description of the UI element (basic textual content):  "{annotation for
basic setting}".

Ensure the annotation meets the following requirements:
```

```
- It directly reflects the UI element's functionality as described in the
textual content.
- It is clear, concise, and free of unnecessary complexity.
- It avoids speculative or ambiguous statements.
- It clearly reflects how the user interacts with the UI element.

Provide only the JSON output without additional explanations.
```

For the **spatial** setting, we iterate over four directions—`[up, bottom, left, right]`—to generate queries about the closest neighboring element. For example, given an element labeled `"previous annotation options"`, a corresponding query might be: `"What is the element that is immediately to the right of 'previous annotation options' horizontally?"` We retrieve the correct bounding box for the queried element using the bounding boxes collected during initial data creation (Section 3.1). To ensure accuracy and reduce ambiguity, we remove samples where no valid closest element exists or where the distance exceeds a set threshold. The processing scripts will be released alongside the code for reproducibility.

B.2. Layout Grounding

GUI layouts define how interface components, such as buttons, text fields, and containers, are arranged into cohesive regions (e.g., "Formatting Tools" or "Main Navigation Bar"). In our layout grounding task, we start with raw UI element annotations collected during the initial data collection phase, which include dense bounding box information for individual elements. These annotations are then fed into LLAMA-3.3-70B (Meta, 2024) using a predefined prompt (provided below) that instructs the model to cluster UI elements into non-overlapping functional and semantic groups. For each group, the model generates a corresponding textual description and computes an aggregated bounding box that encompasses all the elements within that group, thereby capturing the higher-level structure of the GUI.

Following the automated model inference, the predicted clusters and bounding boxes are subjected to a rigorous human verification process. Expert annotators in our research team review and validate each functional group to ensure that the clusters accurately represent the intended UI regions and that the aggregated bounding boxes fully cover the grouped elements. Specifically, we remove all samples that do not meet the requirements and retain those that do. This two-stage procedure yields a high-quality dataset of 311 human-verified query-label pairs spanning 77 platforms, with each functional group typically containing 5–10 UI elements. The process thus extends individual element grounding to a comprehensive layout grounding task, emphasizing both spatial organization and semantic coherence in desktop interfaces. To the best of our knowledge, this is the first work to introduce this setting and systematically evaluate GUI agents in this context.

Prompt used with LLAMA-3.3-70B to generate samples for layout grounding

```
Below is a list of dictionaries representing UI elements for the software
'{software_name}'.  Please use the knowledge you have on this software to
analyze these elements and group them based on their functionality.

Your task is to:

1.  Analyze each element for text, category, and boundingBox attribute.
2.  Identify the elements that belong to a particular functional group (e.g.,
all the file edit tools).  Name as many as you possibly can, but make sure the
functional groups make sense.
3.  Combine the bounding boxes of all elements belonging to that functional
group into a single bounding box that encompasses them all.
4.  Return a structured response (JSON or similarly structured text) that
includes:  - The name of the group (e.g., "Edit Tools Region").
- An explanation of the group's functionality.  Please be as specific as
possible but do not simply repeat the names of the UI elements in the group.
```

```
The coarse bounding box (x1, y1, x2, y2, etc.).

Here is the list of dictionaries you should work with:
{raw data for the given screenshot}

Please produce your final answer in a structured JSON-like format, such as:

[
    {
        "name": "Edit Tools Region",\\[1ex]
        "explanation": "This region contains all the tools related to
                        editing the files.",
        "boundingBox": \{
            "x1": ...,
            "y1": ...,
            "x2": ...,
            "y2": ...
        \}
    \},
    ...
]

Some additional requirements:

1.  Please try your best to make functional groups that cover all the elements
in the list.
2.  Please make the functional group not overlap with each other.  Try to
prioritize non-overlapping bounding boxes containing more elements.
3.  Make sure the name of each group is descriptive as well as meaningful to
the human user and relevant to the elements it contains.  Please make sure the
name is not too generic.
4.  Make the bounding box complete, based on the bounding boxes of the
elements it contains.

Make sure to provide the accurate bounding box that encloses all relevant
elements for each functional group.  And return only this JSON object.
```

B.3. Action Prediction

We begin with the raw action trajectories collected and annotated by humans during the initial data collection stage. Table 7 lists all recorded actions. Our analysis revealed that current GUI models are limited in their ability to predict a wide range of actions, as they are typically trained on a restricted set. To simplify evaluation, we standardized actions into five categories: click, move to, drag, typing, and hotkey. These actions are common across all models. We excluded tasks involving key down, key up, and scroll actions, reducing the total number of tasks to 442. Additionally, we applied the following preprocessing steps to refine the action data:

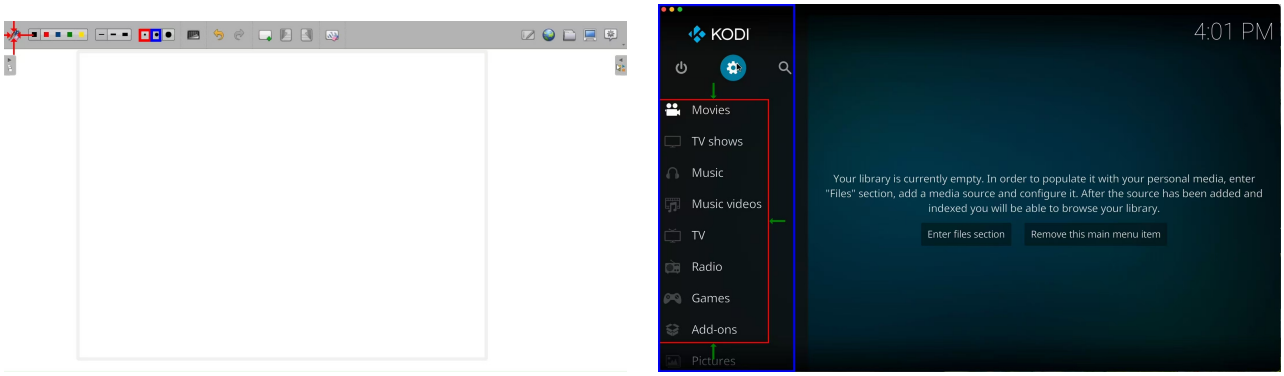
1. **Removing Redundant Move Actions:** If a ”move to” action directly preceded a ”click” action, we removed it. Since the ”click” action already contains positional data, the preceding movement was unnecessary.

2. **Grouping Drag Actions:** A drag action is always preceded by a mouse-down event and followed by a mouse-up event. We merged these three actions into a single ”drag” action. Additionally, since a drag operation requires moving the cursor to the starting position, we included this movement, resulting in a standardized format: drag(x1, y1, x2, y2) where x1, y1 is the start of the drag and x2, y2 is the end coordinates of the drag.

3. **Merging Press and Hotkey Actions:** In the raw trajectories, a ”press” action referred to pressing a single key, whereas a ”hotkey” involved multiple keys. We merged both into a single ”hotkey” category to maintain consistency.

Action Type	Parameters	Description
MOVE_TO	*x, y*	Move the cursor to the specified position
CLICK	*button, x, y, num_clicks*	Click the mouse button at a specified location
MOUSE_DOWN	*button*	Press and hold the specified mouse button
MOUSE_UP	*button*	Release the specified mouse button
DRAG_TO	*x, y*	Drag the cursor to the specified position with the left button pressed
SCROLL	*dx, dy*	Scroll the mouse wheel up or down
TYPING	*text*	Type the specified text
KEY_DOWN	*key*	Press and hold the specified key
KEY_UP	*key*	Release the specified key
HOTKEY	*key/keys*	Press the specified key or key combination

Table 7: Table of original Mouse and Keyboard Actions during human demonstration.



(a) **Element Grounding** with a query: "What is the element that is immediately to the left of the "medium eraser" horizontally?" The prediction result is pointed by red arrows. The element in the blue box is the reference one referring to "medium eraser" while the red one is the ground truth.

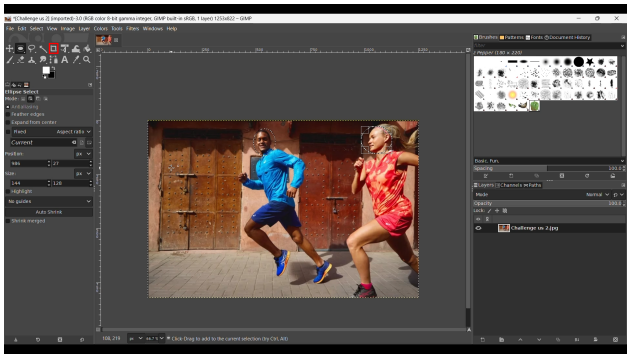
(b) **Layout Generation** with attributes: "Main Menu Region which contains the main menu items for navigating to different sections of the Kodi application, such as movies, TV shows, music, and more." The red is the ground truth while the blue box is the model prediction.

Figure 8: Visualization of Element Grounding and Layout Generation tasks.

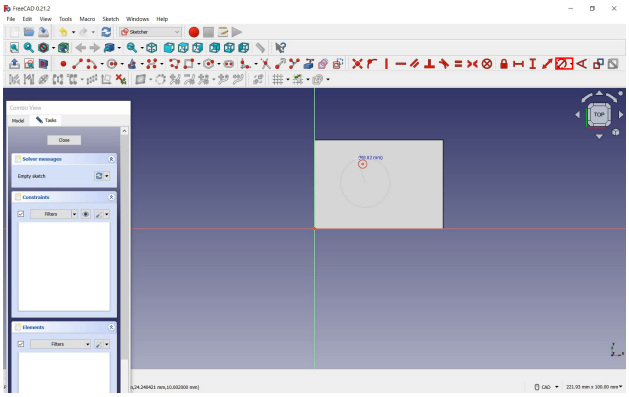
C. Data Statistics and Examples

Data Statistics. Figure.13 presents key statistical distributions within the UI-Vision. (a) Shows the distribution of bounding box number per screenshot, indicating a mean of 74.3, reflecting the dense UI elements present in each frame. (b) Illustrates the distribution of human recording durations in UI-Vision, with a mean of 38.2 seconds, highlighting the variation in GUI navigation task. (c) Depicts the distribution of the number of action steps required to complete tasks, with a mean of 14.0, indicating the long-horizon challenges and interaction complexity within the dataset.

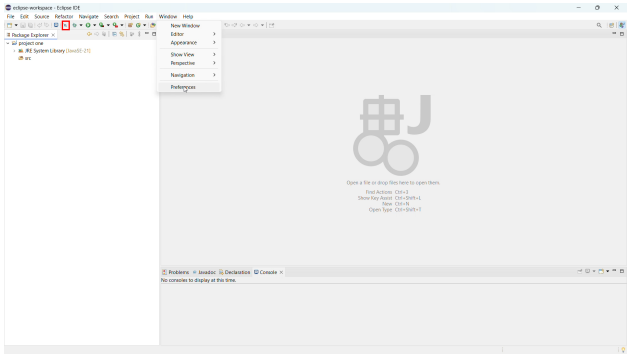
Visualization. In Figure.8, we demonstrate error cases for the Element Grounding (UI-TARS-7B) and layout tasks (Gemini-pro-1.5). In the Element Grounding task (a), which presents the most challenging visual relationship setting, the model correctly identifies the direction ("left") but fails to accurately estimate the distance ("immediately to the left"). Successfully addressing this requires the model to *jointly localize the reference element (medium eraser) while accurately measuring both direction and distance*. In the Layout Grounding task (b), the challenge lies in the model's inability to *detect finer distinctions* (such as "power-off", "setting" and "search") that are not included in the provided attributes. As a result, the model fails to generate a compact bounding box, presenting an even greater challenge.



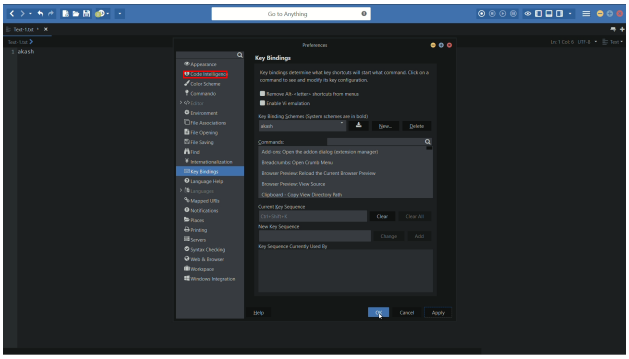
(a) Query: "`crop tool`". The ground truth element is bounded by red bounding box. The platform of the example is **GIMP**.



(b) Query: "`constrain arc or circle`". The ground truth element is bounded by red bounding box. The platform of the example is **FreeCAD**.



(c) Query: "`skip all breakpoints`". The ground truth element is bounded by red bounding box. The platform of the example is **Eclipse**.



(d) Query: "`code intelligence`". The ground truth element is bounded by red bounding box. The platform of the example is **Komodo Edit**.

Figure 9: Element Grounding examples candidate for Basic setting

D. Error Analysis

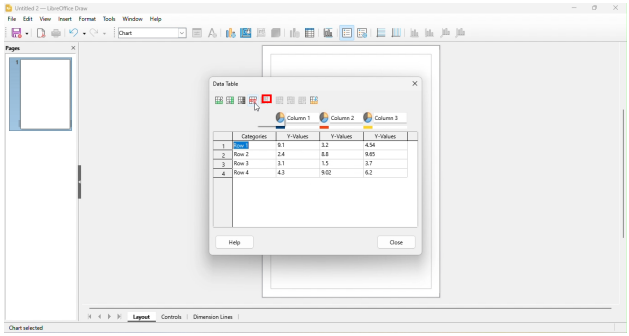
D.1. Element Grounding

Below, we present several failure cases predicted by the top-performing UI-TARS model (Qin et al., 2025) to examine the limitations of current models. We highlight the prediction of the model with a red box. The ground truth is indicated by a red arrow surrounding it.

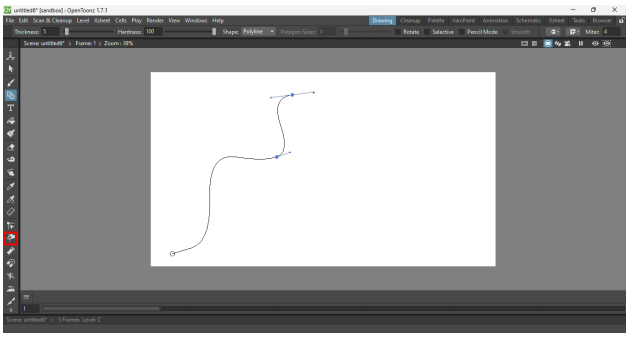
A. Fine-grained Difference: As shown in Fig. 15, the agent effectively understands the query and primarily responds to the nearby region but struggles to recognize the correct target among several similar candidates. This highlights the need for existing agents to develop advanced validation strategies.

B. Lack of Domain knowledge : In Fig.16, We observed that the model's difficulty in identifying the most efficient approach to task completion underscores a deficiency in domain-specific knowledge, such as understanding what the letter "F" signifies. This challenge prompts us to consider developing solutions based on Retrieval-Augmented Generation (RAG) or incorporating external databases, such as integrating software documentation or tutorials, to enhance the model's performance.

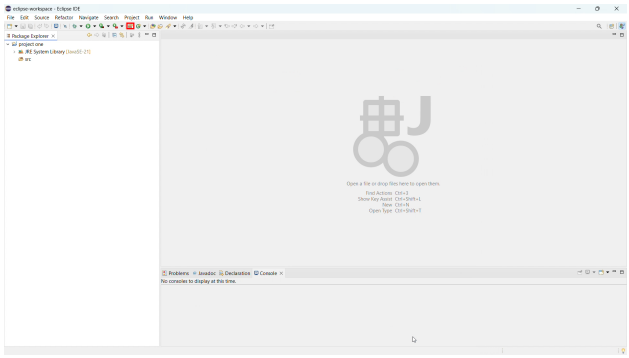
C. Small Items: In Fig.17, we found that model struggling to small visual element especially in high resolution. Potential strategy is to develop iterative zoom-in strategy as V-Search (Wu & Xie, 2024).



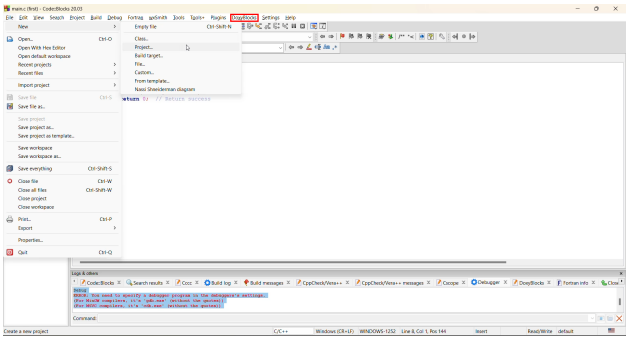
(a) Query: "`Remove data series from the chart`". The ground truth element is bounded by red bounding box. The platform of the example is **LibreOffice Draw**.



(b) Query: "`Create a new Java package`". The ground truth element is bounded by red bounding box. The platform of the example is **Eclipse**.



(c) Query: "`Adjusts path curvature by pinching vector points`". The ground truth element is bounded by red bounding box. The platform of the example is **OpenToonz**.



(d) Query: "`Open the DoxyBlocks plugin menu`". The ground truth element is bounded by red bounding box. The platform of the example is **Code::Blocks**.

Figure 10: Element Grounding examples candidate for Functional setting

D.2. Layout Grounding

A. :

D.3. Action Prediction

E. More Experimental Analysis

E.1. Effect of Screenshot Resolution on Element Grounding Accuracy

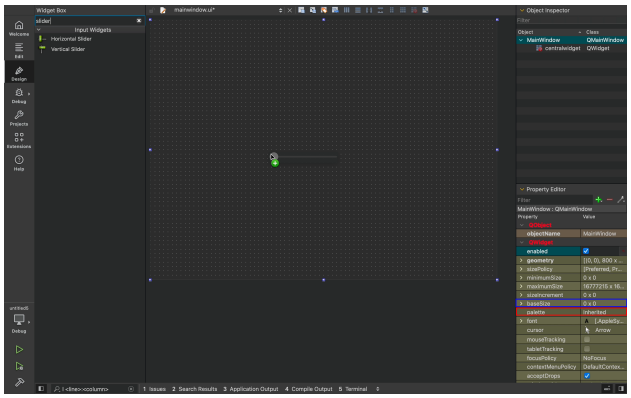
Figure 20 presents the grounding accuracy of the two best-performing models, **UI-Tars-72B** and **UGround-v1-72B**, across different screenshot areas (i.e., resolutions). Observing the accuracy distributions, there is no clear trend indicating a strong correlation between screenshot resolution and grounding accuracy. The accuracy values fluctuate across different resolution levels for both models, suggesting that factors other than resolution may have a more significant impact on performance.

F. Limitation and Future works

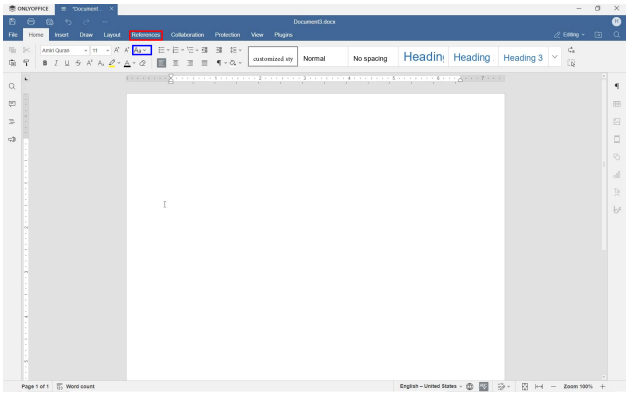
Our benchmark currently includes most visual tasks and offline annotations. However, it is important to extend this to an **online** environment to better evaluate real-world interactions.

Also, for our Element Grounding benchmark, one future direction is how to handle detecting key elements within multiple available elements when creating the annotations to further alleviate the ambiguity in the annotations.

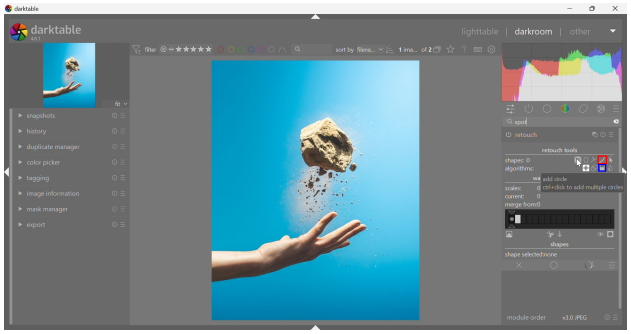
We have not yet leveraged human-recorded videos, which are crucial for assessing model performance in **GUI action understanding**. For instance, providing video clips and querying the model about possible actions could offer deeper



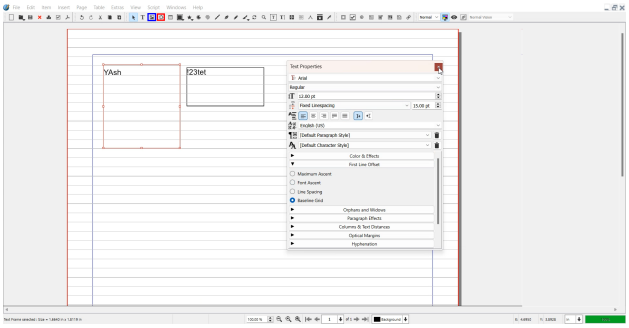
(a) Query: "Directly vertically below the **"baseSize",** can you find the closest element?". The element in the blue box is the reference one referring to "medium eraser" while the red one is the ground truth. The platform of the example is **Qt Creator**.



(b) Query: "What is the element that is vertically closest to the **"change case"** and above it?". The element in the blue box is the reference one referring to "medium eraser" while the red one is the ground truth. The platform of the example is **OnlyOffice Document Editor**.



(c) Query: "What is the element that is vertically closest to the **"Activate fill tool"** and above it?". The element in the blue box is the reference one referring to "medium eraser" while the red one is the ground truth. The platform of the example is **Darktable**.

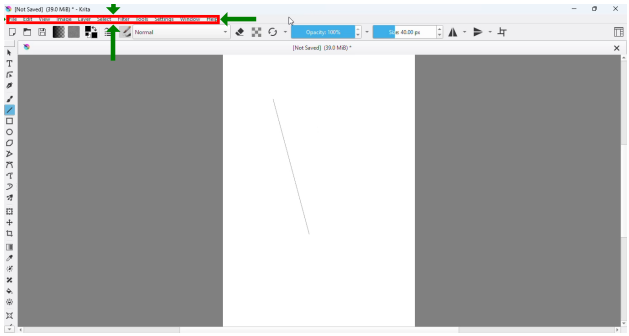


(d) Query: "What is the element that is horizontally closest to the **"Image Frame"** and to the right of it". The element in the blue box is the reference one referring to "medium eraser" while the red one is the ground truth. The platform of the example is **Scribus**.

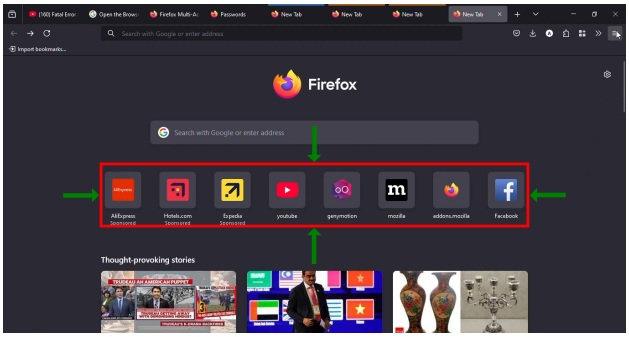
Figure 11: Element Grounding examples candidate for Spatial setting

insights into its capabilities.

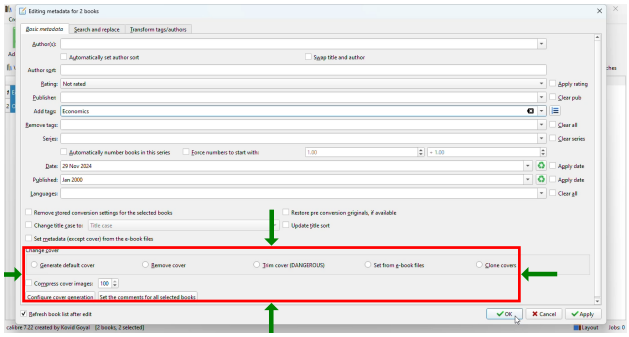
Furthermore, we do not explore **combined actions involving both mouse and keyboard inputs**, such as holding Control while dragging an item. This presents challenges for accurate evaluation, as such actions require precise coordination and understanding.



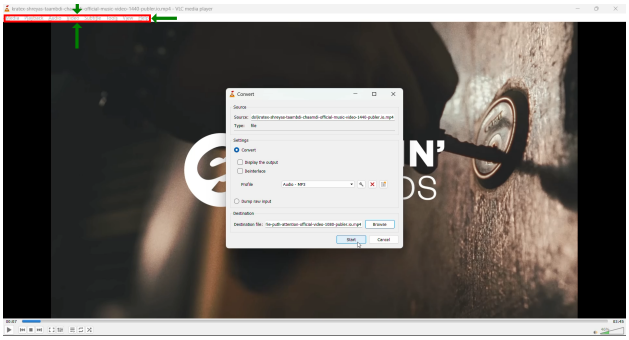
(a) Query: "`Menu Bar: this region contains the main menu options for the application, including file, edit, view, and more.`" The red is the ground truth. The platform of the example is **Krita**.



(b) Query: "`Shortcut Region: this region contains shortcuts to various actions and features such as bookmarks, history, and settings.`" The red is the ground truth. The platform of the example is **Mozilla Firefox**.

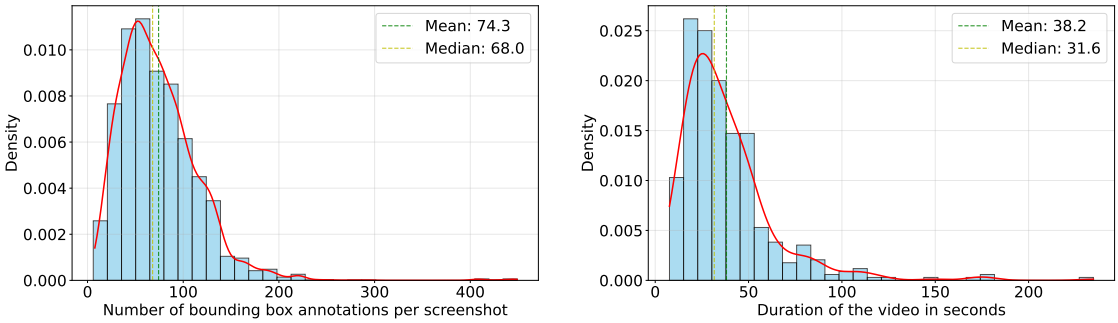


(c) Query: "`Cover Management Region: this region contains all the tools and options related to managing book covers, such as changing, removing, and generating covers.`" The red is the ground truth. The platform of the example is **Calibre**.



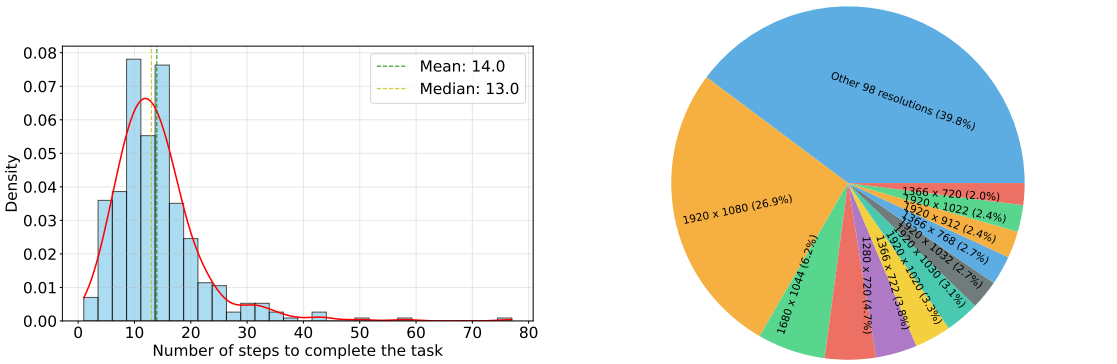
(d) Query: "`Top Menu Bar: this region contains the main menu options for the application, including Media, Playback, Audio, Video, and Help.`" The red is the ground truth. The platform of the example is **VLC Media Player**.

Figure 12: Layout Grounding examples candidate



(a) Distribution of bounding boxes

(b) Video duration distribution



(c) Action steps distribution

(d) Distribution of different screenshot resolutions

Figure 13: **Dataset Statistics.** (a) Distribution of the number of bounding boxes present per keyframe. (b) Distribution of video durations in UI-Vision. (c) Distribution of the number of actions required to complete the task.

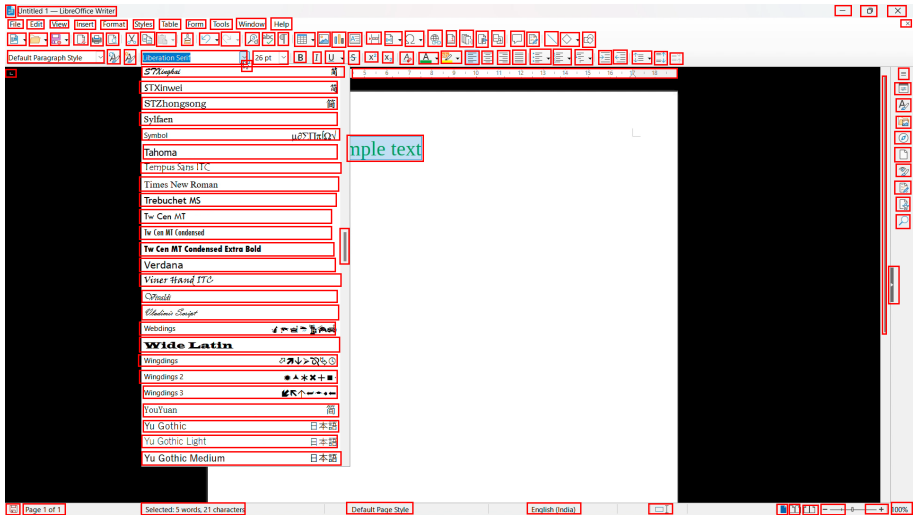


Figure 14: Dense and complete element coverage of bounding box annotations.

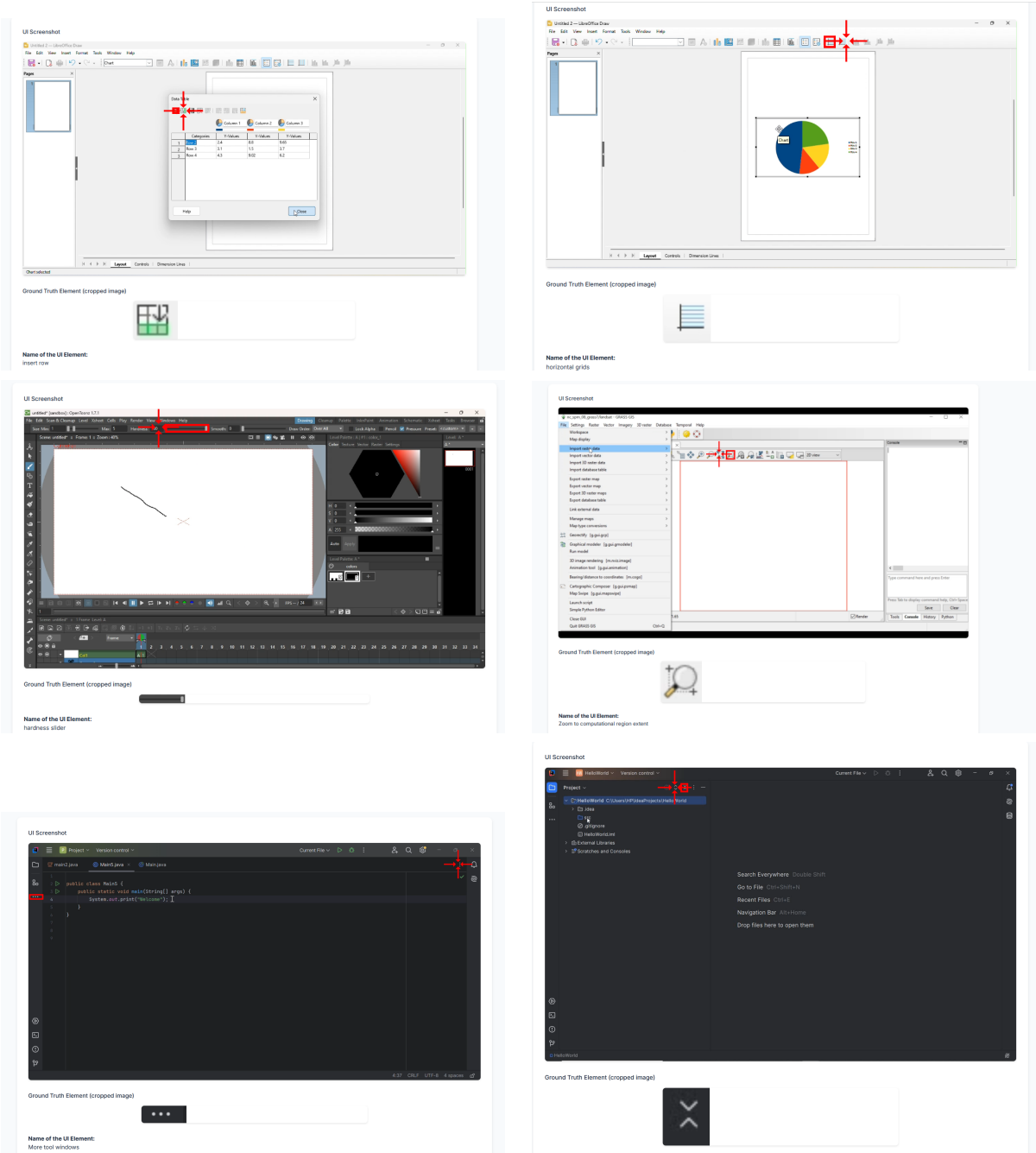


Figure 15: **Error cases of failure to identify the fine-grained differences.**

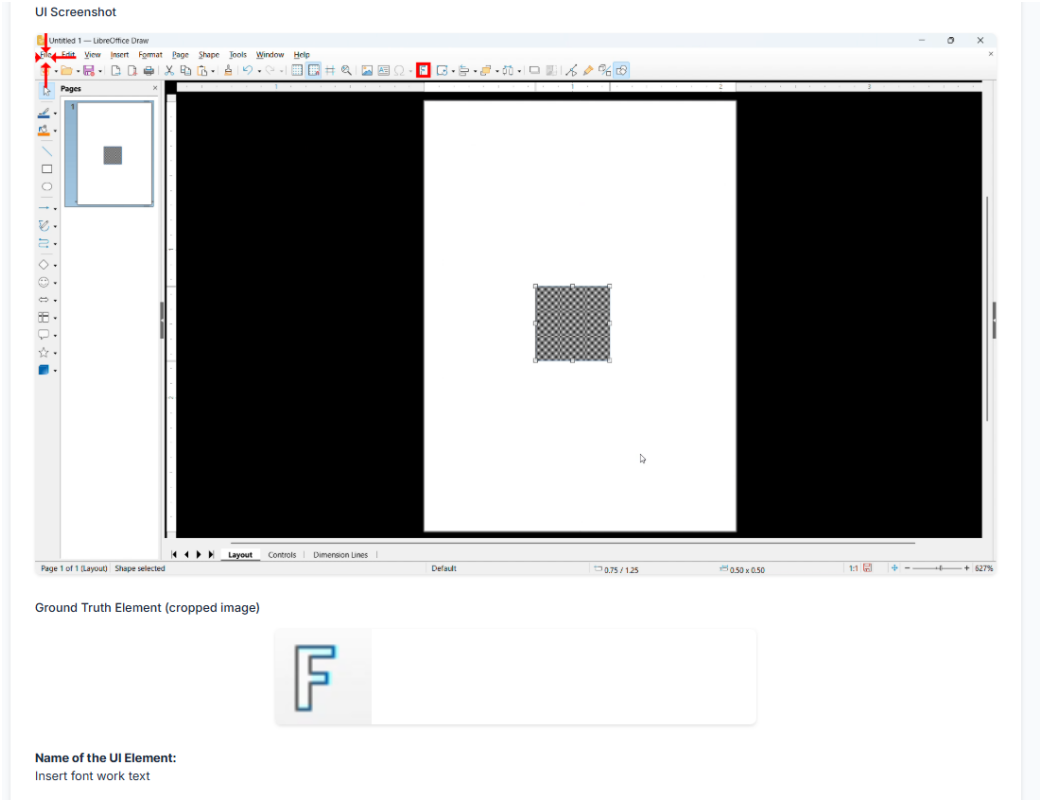


Figure 16: **Error cases of Lacking Domain Knowledges.**

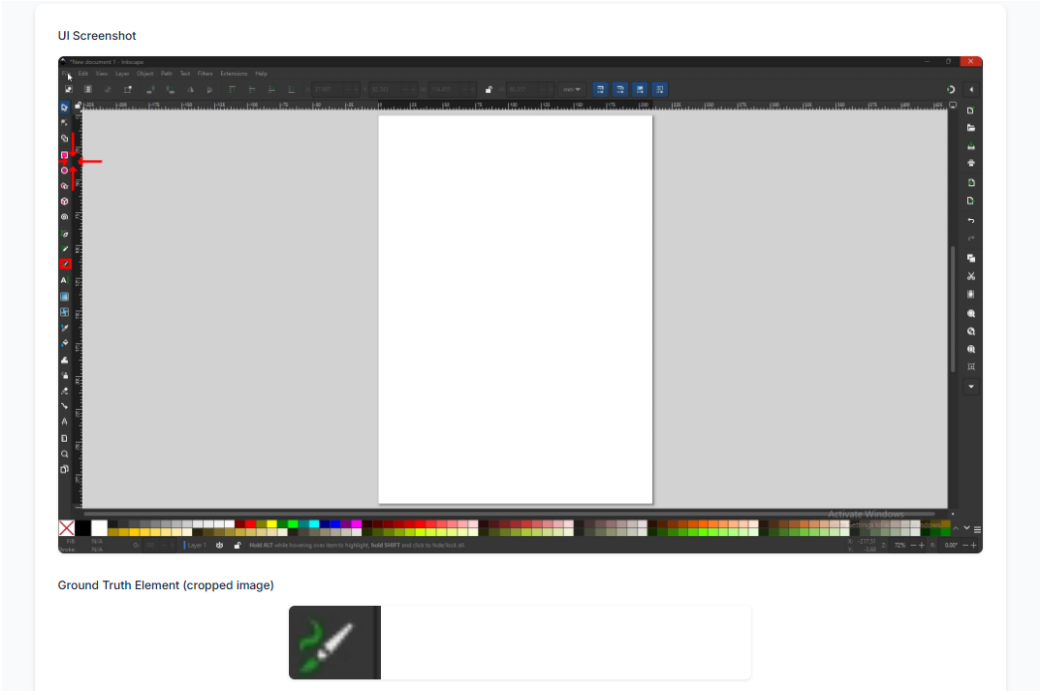
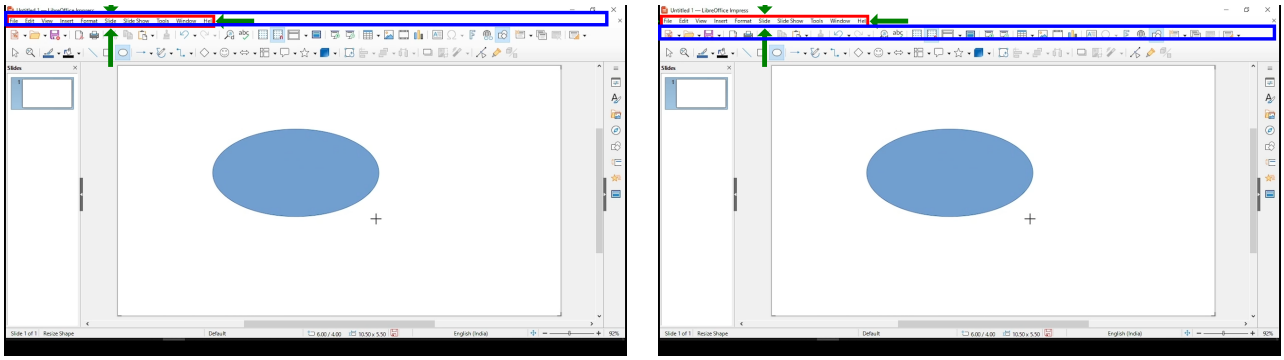
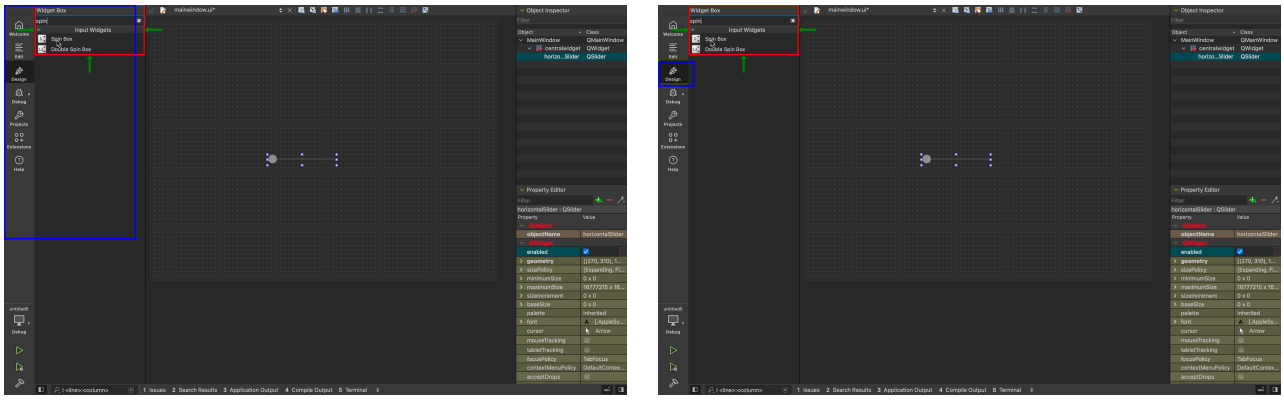


Figure 17: **Error cases of Small Items within High resolution.**



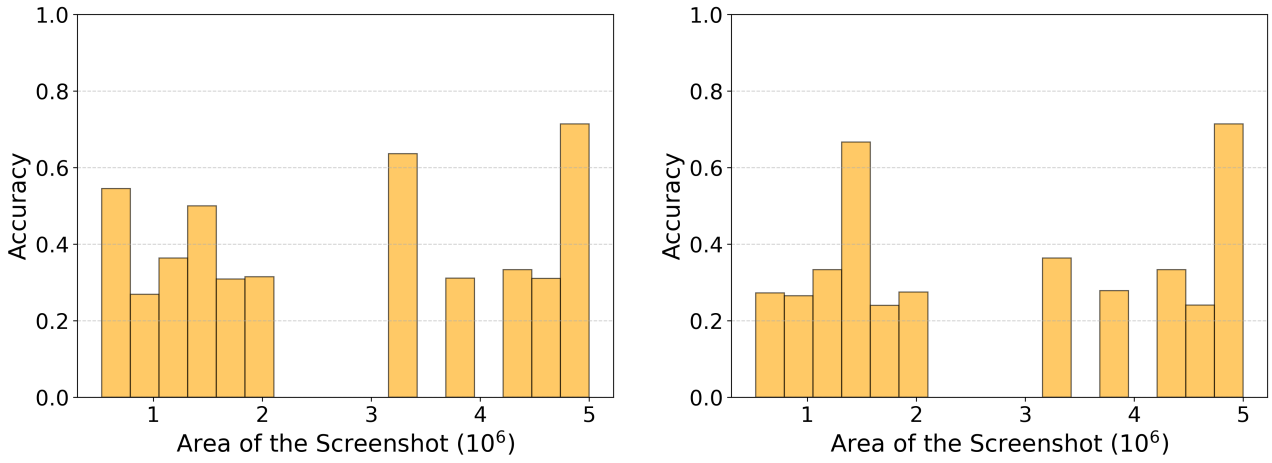
(a) Case from Gemini-1.5-pro (b) Case from OSAtlas-7B

Figure 18: Layout Grounding error case. Query: "`Menu Bar: this region contains the main menu options for the application, including file, edit, view, and more.`" The red is the ground truth. The platform of the example is **Krita**.



(a) Case from Gemini-1.5-pro (b) Case from OSAtlas-7B

Figure 19: Layout Grounding error case. Query: "`Menu Bar: this region contains the main menu options for the application, including file, edit, view, and more.`" The red is the ground truth. The platform of the example is **Krita**.



(a) **UI-Tars-72B** (b) **UGround-v1-72B**

Figure 20: **Analysis on Element Grounding accuracy on top two performing models in terms of the area of screenshot (i.e. resolution of screenshot).**