Towards Fine-Grained Diagnosis of Computer Use Agents

Surgan Jandial

CMU-RI-TR-26-63

June 2026

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee

Fernando De la Torre Frade, Co-Chair
Andrea Bajcsy, Co-Chair
Louis-Philippe Morency
Yinong Oliver Wang

Submitted in partial fulfillment of the requirements for the Degree of

Master of Science in Robotics

To my parents, sister, partner and Ram.

Acknowledgements

I would first like to express my sincere gratitude to my advisors, Professor Fernando De La Torre Frade and Professor Andrea Bajcsy. Their guidance was invaluable in helping me identify meaningful research problems, refine my ideas, and develop a more rigorous and formal perspective from which to approach them. I am especially grateful for the times they challenged my assumptions and pushed back on my ideas, as these discussions strengthened both my research and my thinking. Above all, I deeply appreciate their patience, generosity, and willingness to accommodate my many requests throughout this journey.

I am especially thankful to Oliver Wang, who began as a mentor but whom I now regard as an elder brother – remarkably wise and always seems to have a thoughtful solution to every problem. I have rarely encountered someone who combines such exceptional ability with such genuine humility. Whatever he chooses to pursue, I have no doubt that he is destined for success. In our own slang, he is truly the GOAT of HSL.

My time in the lab was filled with conversations about everything and anything, constant jokes, stimulating discussions, and, occasionally, perhaps a little too much politics. These moments made the experience memorable, and I will always cherish the time I spent with everyone I met along the way: Runkai, Ashish, Jose, Saswat, Utkarsh, Aviral, Xiao, Yehonathan, Jianjin, Nicholas, Soham, Kennard, Xia, Jialu, Junkai, Willy, Srinath, Ritaban, Tanisha, Yixing, and Minhyek. Tan Nhat Bui deserves a special mention for his incomparable energy, and character.

This journey was made even more enjoyable by the company of an extraordinary group of peers, each of whom taught me something new: Karan, Pratik, Shantanu, Arsh, Richard, Avigyan, Srinivas, Sriram, Abhishek, and many others whose names I may have unintentionally missed.

I am forever indebted to my parents, my sister, and Winnie, who stood by me through thick and thin. They know a version of my Pittsburgh experience that no one else does, including the challenges and struggles that shaped these years in ways I

would never wish.

Finally, as a devout Hindu, I offer my deepest gratitude to the divine forces that guided and protected me at every step of this journey. I am who I am because of the blessings of Lord Rama and Lord Hanuman, and because of the teachings of Shri Premanand Maharaj. I pray that their wisdom and grace continue to guide me and inspire the world toward truth, compassion, and righteousness.

Abstract

Recent advances in Large Language Models and Vision-Language Models have enabled agents that interact with graphical user interfaces and web environments on behalf of users. Although these agents show promising capabilities, they remain unreliable in complex interactive settings. A key reason is that existing evaluations often emphasize high-level outcomes, such as task success or next-action accuracy, without diagnosing the underlying capabilities that cause agents to succeed or fail. As a result, important failure modes in perception, grounding, temporal reasoning, and planning can remain hidden behind aggregate metrics.

This thesis argues that building reliable GUI and web agents requires fine-grained diagnostic evaluation of core agentic capabilities. We study this problem through two complementary directions. First, we examine planning in web agents by evaluating whether Vision-Language Models understand the temporal structure of web interaction and can assess whether a sequence of actions achieves a given goal. Our analysis shows that current models struggle with fundamental planning prerequisites, including ordering webpage states, reasoning about state changes, and selecting effective plans. Second, we investigate grounding in GUI agents by testing whether models consistently identify the same UI element across diverse natural-language descriptions. Focusing on desktop environments, we introduce a sensitivity-based evaluation that reveals substantial variation in model predictions when the same target is referred to from different perspectives.

Together, these studies demonstrate that models can perform well under standard benchmarks while still lacking crucial fine-grained capabilities needed for reliable interaction. By decomposing agent evaluation into targeted diagnostic tests, this thesis provides a more precise understanding of current limitations in GUI and web agents. The proposed evaluations and benchmarks offer tools for identifying failure modes, guiding model development, and advancing toward more robust and trustworthy interactive AI systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in Large Language Models (LLMs) [51, 57] and Vision-Language Models (VLMs) [2, 46] have enabled a new class of autonomous agents that interact with digital environments on behalf of users. These agents are expected to understand user instructions, perceive graphical or web interfaces, reason about task requirements, and execute actions such as clicking, typing, selecting, or navigating across pages. In particular, GUI automation [44] and web agents [15, 72] have emerged as promising directions for building practical AI systems that can complete complex tasks in interactive environments.

Despite this promise, current agents remain far from reliable. Successful interaction with digital environments requires multiple interdependent capabilities. An agent must perceive the interface and identify relevant elements, ground instructions to concrete coordinates or UI components, reason about the current state, plan a sequence of actions, and execute those actions correctly. For web agents, prior work has characterized this process as requiring perception of webpage structure, planning over task requirements, and action execution toward the desired goal [14, 23]. However, failures in any of these components can cause the overall task to fail, and current VLM-based agents remain inadequate for reliable deployment [15, 22, 33, 72].

A central limitation of existing evaluation is that it often focuses on high-level outcomes. Prior web-agent benchmarks commonly assess whether a task is completed or whether an agent's predicted action is correct [8, 15, 49]. Similarly, GUI grounding benchmarks typically evaluate whether a model can predict the correct coordinate or bounding box for a given instruction [6, 26]. While these evaluations are important, they provide limited insight into why an agent succeeds or fails. A low task success rate may result from weak perception, incorrect grounding, poor temporal understanding, flawed planning, or brittle action execution. Aggregate metrics therefore obscure the

underlying failure modes that must be understood in order to build reliable agents.

This thesis argues that progress toward reliable GUI and web agents requires moving beyond high-level evaluation and toward fine-grained diagnosis of the underlying capabilities that enable agentic behavior. Rather than treating an agent as a black box and measuring only final success, this thesis studies specific failure modes in two core components of interactive agents: planning and grounding. Planning determines whether an agent can reason about how interface states evolve over time and identify a sequence of actions that satisfies a goal. Grounding determines whether an agent can map a user instruction to the correct UI element or coordinate. Together, these capabilities form a critical bridge between perception, reasoning, and action.

The first part of this thesis studies fine-grained planning capabilities in web agents. Existing works primarily focus on perception skills [33, 58], and do not directly assess the planning abilities required for reliable web interaction. We define planning as an agent's ability to devise a sequence of actions to successfully complete a given task. In the web setting, a capable planning model must understand the order in which webpage states should occur, recognize how actions change those states, and assess whether a proposed plan can achieve the desired goal. We therefore construct diagnostic evaluations that test temporal understanding and plan assessment directly. These evaluations reveal that current VLMs struggle with basic prerequisites for web planning: they often fail to infer the correct ordering of webpage states, rely on spurious input-order heuristics, degrade as action sequences become longer, and perform poorly when asked to assess plans in both real and synthetic web environments.

The second part of this thesis studies fine-grained grounding capabilities in GUI agents, with a particular focus on desktop environments. GUI grounding refers to predicting the exact coordinate or bounding box where an action, such as a click, should be performed. Recent works have trained large-scale GUI grounding models [13, 50, 61] and developed benchmarks for evaluating grounding performance [6, 26]. However, existing data and benchmarks are heavily skewed toward web and mobile environments [9, 22, 53], while desktop-specific GUI grounding remains comparatively underexplored [3, 18, 62]. This imbalance partly stems from the availability of HTML and XML in web and mobile settings, which makes it easier to extract UI element–coordinate pairs, whereas desktop environments often provide only screenshots and require manual annotation or proprietary VLMs.

Beyond the lack of desktop-focused evaluation, existing GUI grounding benchmarks often use a single instruction per UI element, typically written in a short, direct, or canonical form. For example, benchmarks may include brief instructions such as 'close

button" [6] or simplified references such as 'Click the Pictures icon" [26]. Although recent works incorporate more implicit referring expressions [34, 43, 63], they generally do not evaluate whether models can consistently identify the same UI element when it is described in multiple valid ways. In realistic settings, users may refer to the same element through visual traits, relative position, surrounding context, or functional role. Therefore, a model that truly understands a GUI should produce consistent predictions for the same element across diverse descriptions.

To evaluate this property, we introduce a sensitivity-based grounding evaluation that measures whether grounding models remain consistent across multiple references to the same UI element. We focus on desktop environments, which are challenging yet underrepresented, and design an automatic data generation and validation pipeline followed by human verification. This results in a GUI grounding sensitivity benchmark that exposes failure modes hidden by conventional single-reference evaluations. Our results show that current grounding models exhibit meaningful sensitivity to how an element is described: models that perform well on standard benchmarks can still produce inconsistent predictions when the same element is referred to from different perspectives. Motivated by these findings, we further develop a GUI grounding diagnosis agent that automatically generates diverse instructions, uses model feedback, and iteratively identifies failure cases.

Across these two studies, this thesis develops a common methodology: decomposing agent evaluation into targeted diagnostic tests. In the planning setting, we ask whether VLMs possess the temporal understanding and plan-assessment abilities required to act reliably on the web. In the grounding setting, we ask whether GUI grounding models robustly identify the same element across diverse user descriptions. Both studies show that models can appear competent under standard evaluations while still lacking important fine-grained capabilities. This suggests that improving agent reliability requires not only stronger models and larger training datasets, but also evaluations that expose the specific conditions under which models fail.

In summary, this thesis makes the following contributions:

- We propose a fine-grained diagnostic perspective for evaluating GUI and web agents, emphasizing that high-level task success alone is insufficient for understanding agent reliability.
- We introduce diagnostic evaluations for web-agent planning, focusing on temporal understanding and plan assessment as prerequisites for reliable web interaction.
- We introduce a sensitivity-based framework for GUI grounding, evaluating

whether grounding models remain consistent across diverse descriptions of the same UI element.

- We focus on the underrepresented desktop setting and develop a benchmark and diagnosis pipeline for identifying grounding model failure cases.

Chapter 2

Fine-Grained Diagnosis of Planning Capability

2.1 Introduction

Recent advancements in Vision-Language Models (VLMs) have shown great potential as web agents capable of autonomously executing user instructions, such as "Buy a flight ticket from A to B". To successfully accomplish these tasks, recent literatures [14, 23] have shown that VLMs must perform three key functions: perceive and comprehend the structure of a webpage and relevant elements (Perception), reason about task requirements and devise actionable plans (Planning), and execute appropriate actions to achieve the desired goal (Action). While VLMs have demonstrated promising potentials, their current performance remains inadequate for reliable web agents [15, 22, 33, 72].

Although limitations in model and algorithms pose a significant challenge, we argue that the lack of holistic evaluations is another key factor hindering progress. More precisely, prior works primarily assess the efficacy of VLM agents by judging the correctness of their actions and measuring task success rates, i.e., whether the goal is achieved [8, 15, 49]. While this is intuitive and aligned with the objective of task completion, end-to-end success rates cannot reveal the bottlenecks of agents or offer detailed diagnosis and insight into the specific Perception and Planning capabilities. Observing the gap, prior works [33, 58] prompt the evaluations of fine-grained VLM skills in web scenarios, such as website grounding and WebQA [33, 58]. However, they focus solely on perception skills, failing to address the underlying abilities required for effective planning as shown in Figure 2.1, thus motivating our central research

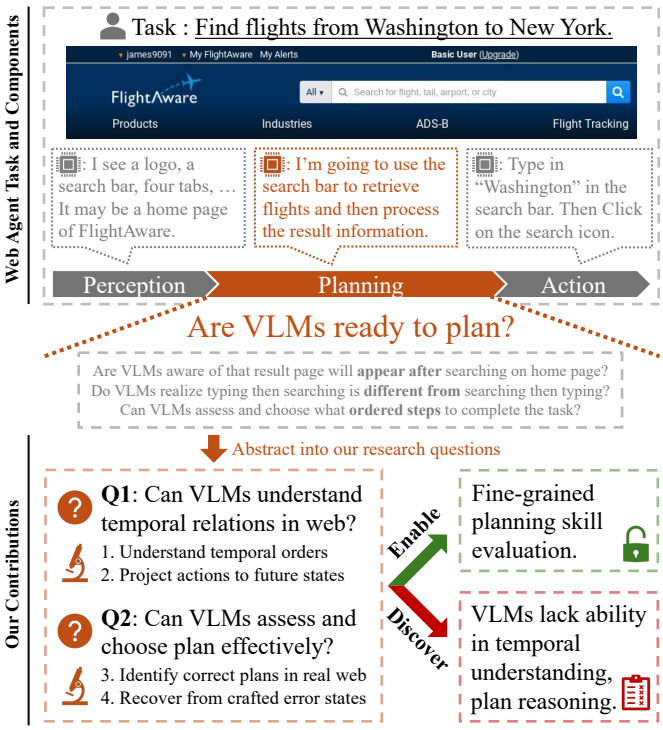


Figure 2.1: **VLMs and Web Planning.** For web agents to complete tasks, a set of critical components are involved. Among them, the fine-grained evaluation in the planning component is essential but overlooked. Thus, we design questions that investigate whether VLMs possess the nuanced planning skills, enabling the first web planning evaluation and concluding the lack of capabilities in VLMs to plan effectively and reliably.

question: ***Have VLMs acquired the necessary capabilities to plan effectively as reliable web agents?***

To concretize our study, we define planning as an agent's ability to devise a sequence of actions to successfully complete a given task. In the context of web, a performant planning model must possess three fundamental skills. First, it needs to understand "**order**", determining which webpage states should come earlier during task completion, essentially answering the question, "What needs to be done first, and what comes later?" Second, it must grasp "**change**", recognizing how the sequence of action(s) transforms the webpage states. Third, it must be able to incorporate the order and change to "**assess**" a plan that achieves the desired goal.

Hence, we design a series of unit tests to target the above prerequisite skills through two key dimensions. In the first dimension, **Temporal Understanding** (Figure 2.1 Q1), we conduct two tests in Section 2.4 to examine whether VLMs can infer which web-page state/screenshot comes earlier in the trajectory of a given task (e.g "find flights from Washington to New York"), and can predict future states following

actions. Our results highlight that VLMs struggle with basic temporal understanding capabilities and they tend to rely on spurious heuristics, such as the order in which inputs are presented, instead of the actual semantic logic. Moreover, as the number of action steps increases, VLMs exhibit declining performance in predicting future states.

In the second dimension, **Plan Assessment** (Figure 2.1 Q2), we examine whether VLMs can reason and choose the correct course of action to achieve a task, given a goal and a screenshot as the start state. Specifically, in Section 2.5, we test on real-world web environments and a synthetic web dataset curated for diversity and controllability. The results show overall poor performance in assessing plans, with significant struggles in handling edge cases, such as reverting erroneous actions (e.g clicking the wrong checkbox) in the synthetic dataset.

Overall, our findings reveal that VLMs are not yet equipped with adequate abilities to reliably plan as trustworthy web agents. We envision that our benchmark can pave the way for future research in thorough evaluations of VLMs and hopefully inspire ideas to advance effective and reliable web agents. In summary, our contributions are:

- We are the first to emphasize and study the fine-grained skills or prerequisites required by the VLMs to serve as reliable web agents.
- We repurpose existing datasets/environments to construct our test datasets and release for future web agent evaluations.
- We benchmark 19 state-of-the-art VLMs and conclude that current VLMs cannot be reliable web agents yet due to limitations in planning.

2.2 Related Work

VLMs as Web Agents. Vision Language Models (VLMs) exhibit great potential across a diverse range of challenges [12, 27, 39]. Of these, one such problem is "Web Navigation" [65, 72], where given a user instruction (e.g., "Buy a pen from eBay"), and the current state (e.g., webpage screenshot, HTML or meta-data), VLMs' objective is to predict the next best action(s) that could lead to the final goal. To succeed, VLMs must excel in three key areas: 1) **Perceiving** web pages with arbitrary formats, dense images and text, 2) **Planning and Reasoning** over user's intent, extracted webpage information, and potential steps to solve the task, and 3) Predicting the next **Action** and its location on the webpage. This led to diverse benchmarks to evaluate VLMs as shown in Figure 2.2.

VLM Agent Task Evaluations. Earlier benchmarks, such as MiniWob [55] and WebShop [66] typically evaluated web navigation in simple, simulated environments.

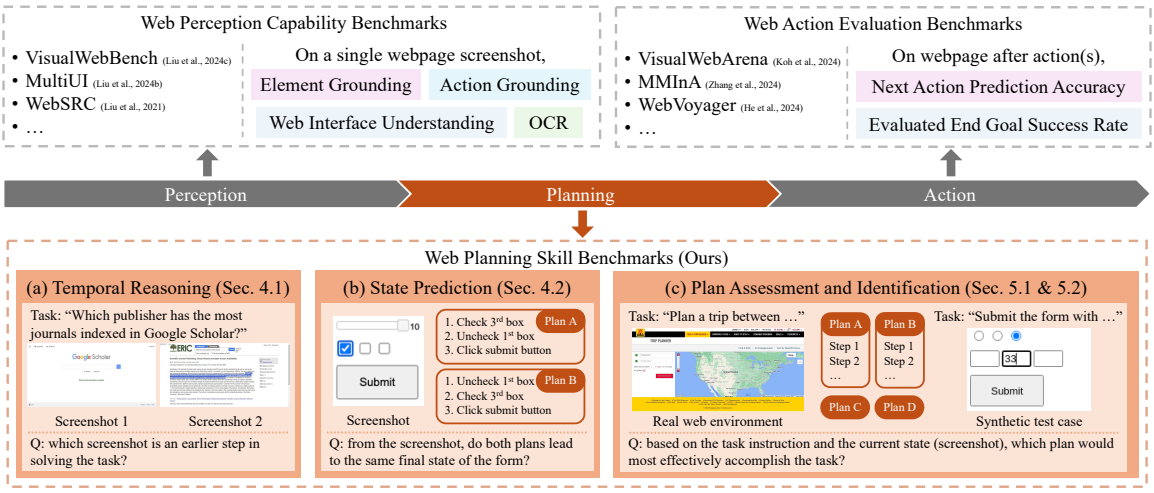


Figure 2.2: **Comparisons of benchmarks in different components.** While extensive study has focused on evaluating the required capabilities of the perception and action stage of web agents, the necessary skills for web planning are overlooked in existing literature and we contribute to cover this gap through four evaluations in this paper.

More recently, works like VisualWebArena [22], SeeAct [72], WebVoyager [15], and MMInA [70] introduced more challenging settings, testing agents on the real web. Building on this, [56] decomposed the agent's performance into grounding and planning by measuring the success rate on samples where VLM achieved perfect grounding. However, these benchmarks primarily assess task success, i.e., whether the final objective is achieved, in two main approaches: the first employs step-wise evaluation using human-annotated datasets, specifying the ground-truth action at each step [8, 72], whereas the second leverages contemporary VLMs (such as GPT-4o, Gemini) to determine if the agent successfully completed the task [15, 49]. Despite their intuitive nature, task success metrics have two key limitations: 1). **Reliability** – As noted in [15], there exist multiple valid trajectories to solve a task and enforcing the agent to match the predefined action for each step makes the evaluation overly restrictive. Moreover, relying on existing VLMs/LLMs to assess task completion introduces errors due to their limited capabilities in web environments [49]; 2). **Depth of Evaluation** – While task success rates measure VLMs' overall effectiveness, they lack fine-grained insights into their capabilities and defects.

VLM Web Understanding and Reasoning Evaluation. To circumvent the above issues in evaluation by success rates, recent works [5, 17, 31, 33] introduce benchmarks to assess VLMs' granular performance in web scenarios. Particularly, VisualWebBench [33] focuses on tasks such as Web captioning/QA, Action Grounding/Prediction, etc. Moreover, WebQuest [58] extends the above single web page

evaluations to web page sequences. They propose Multi-Screen QA, which requires gathering information across multiple webpages, and Trace QA, asking about the entire sequence viewed by the user when browsing the web. However, their questions focus on extracting information across multiple web pages instead of reasoning or action prediction. In summary, the above works focus on two categories: 1). Perception, where VLMs process the textual and visual content of web pages to answer direct or complex information extraction questions; 2). Single-Step Action, where VLMs determine the nature and the location of action on the user screen. This lack of focus on Planning motivated our fine-grained investigation into such capabilities. We devise novel questions, experiment formulations and examine 19 VLMs to derive our analysis. To our knowledge, this is the first work to analyze VLMs' nuanced planning capabilities in web environment.

2.3 Experiment Setup

Overview: In our experiments (Sections 2.4 and 2.5), we test VLMs on four groups of questions as discussed in Section 2.1 and Figure 2.2 and derive model responses by prompting the model to solve multiple-choice questions (MCQs) with randomly shuffled options and one correct choice labeled. In Section 2.4.1, we also evaluate the model in a non-MCQ QA setup for comparison. Moreover, for each experiment, we prompt VLMs with direct instructions (Instruct) and chain-of-thought (CoT) reasoning [24].

VLMs used: Our experiments evaluate 19 VLMs including general multi-modal and GUI-specific models. Considering the importance of practical-size web agents (Section A.10), for general VLMs, we choose Qwen2-VL (2B, 7B, 72B) [59], Llava-1.6 (7B, 13B, 34B) [30], Llava-One-Vision (0.5B, 7B) [25], Minicpm-V [67], Minicpm-o [41], InternVL2-MPO (8B) [60], DeepSeek-VL [36], GLM-4V [36], Idefics3 [20], and Phi-3.5 Vision [11], and for GUI-specific models, we consider Ferret-UI (2B, 8B) [68], UIX-Qwen2 [32], and CogAgent [16]. We utilize a maximum of 4 A4500 GPUs.

Response Evaluations: We parse the VLM output and measure accuracy against the ground truth of the questions as our metric. While we typically use deterministic regular expressions for parsing, some models did not follow the output format, for which we use Llama3-8B [10] for parsing (see Section A.5). As additional baselines, we also conduct a human study with at least 15 raters on 15 randomly sampled questions (with the same input and instruction for VLMs) per test. More details about human study are in Section A.6.

Model	Dataset1 - **MM-M2W**								Dataset2 - **GUIACT**								**Avg**
	MCQ				Non-MCQ				MCQ				Non-MCQ				
	Instruction		CoT		Instruction		CoT		Instruction		CoT		Instruction		CoT		
	Order1	Order2	Order1	Order2	Order1	Order2	Order1	Order2	Order1	Order2	Order1	Order2	Order1	Order2	Order1	Order2	
Qwen2-VL 2B	50.14	49.95	48.95	50.09	87.21	5.35	78.29	11.99	74.86	15.71	48.46	51.32	43.35	49.55	71.23	14.55	46.94
Qwen2-VL 7B	72.1	2.9	75.0	6.34	53.51	30.32	24.87	55.99	61.36	11.09	62.18	22.24	14.33	50.69	11.72	72.71	39.21
Qwen2-VL 72B	81.26	18.58	76.52	43.89	81.36	27.35	77.0	47.17	79.64	44.62	89.77	82.64	85.4	57.95	88.69	72.41	65.89
Minicpm	61.04	34.44	51.9	49.95	62.53	14.96	78.25	16.25	60.35	24.01	50.04	53.2	7.16	51.7	60.15	39.18	44.69
Phi 3.5	63.62	8.02	77.85	16.22	81.96	1.28	96.43	4.75	45.84	4.4	54.76	17.67	20.75	6.28	48.81	22.65	35.71
IDEFICS3	100.0	0.0	97.12	1.11	95.93	2.08	98.01	1.68	95.23	3.41	81.44	11.04	33.48	28.72	93.84	9.99	40.88
DeepSeek VL	100.0	0.0	68.83	27.63	100.0	0.0	98.41	8.9	100.0	0.0	83.79	14.16	97.97	0.25	98.31	1.3	31.41
Llava OV 0.5B	51.43	49.0	77.89	20.81	51.13	44.3	96.33	4.06	55.2	43.98	55.71	42.23	100.0	0.0	97.97	2.4	43.34
Llava OV 7B	72.34	17.59	79.28	23.38	25.66	48.66	85.62	12.38	83.84	7.78	80.93	20.01	0.03	77.81	55.79	16.78	44.24

Table 2.1: **Evaluation on Temporal Ordering Task.** We show the mean accuracy of various VLMs on two datasets across different configurations of the temporal ordering task. Pic1 and Pic2 indicate which the correct answer is. We observe most models fail to exceed the random guess baseline of 50%, with only Qwen2-VL 72B consistently surpassing it at 65.89%. The results highlight VLMs' reliance on spurious heuristics, i.e., the order of options, rather than genuine temporal understanding.

2.4 Temporal Understanding Analysis

Understanding temporal relations is a fundamental aspect and prerequisite for effective planning. For web agents, comprehending the sequential order of events (e.g., "adding to shopping cart before checking out") and reasoning about how a webpage's state changes through a series of interactions is critical for making informed plans and decisions. In this section, our objective is to evaluate the temporal understanding of VLMs through two hypotheses: 1). Temporal Ordering: *Can VLMs identify which webpage screenshot/state comes earlier in the user's trajectory for a given task (e.g., filling out form comes before submitting it)* 2). Future State Prediction: *Can VLMs determine whether two sequences of actions lead to same future states?*

2.4.1 Temporal Ordering

Motivation: As agents often process sequences of prior state screenshots to predict subsequent actions, understanding the order of webpage states during navigation is crucial to reason about the dynamics in web tasks. This process requires agents to inherently understand task trajectories (e.g., during online shopping) and infer the sequential ordering of screenshots to make decisions. To test this requirement, we assess whether VLMs can understand the temporal relationship of screenshots representing different steps in a task.

Experiment Setup: In this test, we provide VLM with a task instruction, two webpage states (or screenshots), and the objective to determine which screenshot represents an earlier step in the trajectory to solve the task. We use Multimodal Mind2Web (MM-M2W) [72] and the web-environment subset of GUIAct [4], both of which consist of a task and a ground-truth trajectory comprising T actions $\{a_t\}_{t=0}^{T}$ and corresponding webpage screenshots $\{i_t\}_{t=0}^{T}$. In these datasets, i_0 (or initial webpage state) is typically the homepage of a website and i_T (or final webpage state) is typically when task is completed. Therefore, for a clear and unambiguous test, we always use i_0, i_t for inputs, hypothesizing that presenting the initial and final webpage states is the easiest question to determine which comes earlier. For completeness, we include further discussion and ablation on randomly sampled i_j, i_k in Section A.2, which yields similar findings.

To prepare inputs for VLMs, we label the screenshot images with IDs "Picture 1" and "Picture 2", and the ordered input image tuple as (Picture 1, Picture 2). To mitigate the influence of prompt variations (specifically input orders), we create two broad configurations for each screenshot pair. In the first setting (referred to as Order1), the input image tuple is (start state i_0, end state i_T), and the expected answer is "Picture 1 comes earlier." (e.g., Figure 2.2(a)); in the second configuration (referred to as Order2), the input image tuple is (end state i_T, start state i_0), and the expected answer is "Picture 2 comes earlier." For each of the above two configurations, we prompt the model in MCQ and non-MCQ format as described in Section 2.3. Each data point thus generates four question types, enabling a comprehensive evaluation of the model's ability to reason about temporal relationships.

Result - Poor sense of temporal relation: We observe in Table 2.1 that the average performance (the rightmost column) of almost every model fails to surpass the random guess performance of **50%** while a human rater achieves a performance of **about 96.2**%. Particularly, only Qwen2-VL 72B, with a mean accuracy of **65.89**% across 16 configurations, exhibits a consistent performance above random guess but still remains significantly lower than human performance. Additionally, we observe no significant performance variations between MCQ and non-MCQ versions in Table 2.1.

Finding - Spurious preference for the first input image: Moreover, we observe that irrespective of the order of images, VLMs tend to prefer the answer "Picture 1 comes earlier in the trajectory" after logically assessing both images before making a decision. Specifically, the mean performance difference across all models, between cases where "Picture 1" is the correct answer and cases where "Picture 2" is the correct answer, is 46.64% for MCQ and 41.66% for non-MCQ. We further study this in

Section A.8 through some qualitative examples, and find that VLMs tend to rationalize their answer of "Picture 1" with untenable reasons. These examples highlight VLMs' reliance on spurious heuristics over genuine temporal understanding, increasing their failures in temporal ordering tests.

2.4.2 Future State Prediction

Motivation: A critical aspect of temporal understanding and planning is imagining webpage state changes and predicting outcomes from action sequences. Particularly, a model must reason about the final states resulting from different action sequences to figure out the most effective course of action for achieving the goal. Although previous works briefly discuss next-state prediction (see Section 2.2), understanding outcomes over a series of actions remains underexplored for VLMs. To address this, we propose a simple formulation and examine whether VLMs can predict if different action plans lead to the same or distinct final states.

Experiment Setup: This test requires the model to evaluate two proposed action sequences, provided with an initial state screenshot i_0, and determine whether the future state i_t achieved by these plans are identical. To construct this evaluation, we utilize the data from MiniWob++ environment [29] to generate user interface screenshots, and two candidate action plans per screenshot, representing common web interactions involving UI elements such as checkboxes, textboxes, sliders (see Figure 2.2(b) and more in Section A.1). To analyze the impact of trajectory length on planning capacity, we generate data samples with short-horizon (1-2 action steps) and long-horizon (3-4 action steps) plans. Although longer horizons (e.g., 6-8 steps) are possible, we limit our exploration to 1-4 steps, which already pose significant challenges for VLMs. For reliability, each data point is manually verified, including validating the logical coherence of action plans and the clarity of resulting states.

Result - Failure in predicting future states: We present our results in Table 2.2. While a random baseline would achieve approximately 50%, most models either perform below or near random guess in both short- and long-horizon scenarios. Notably, for the CoT setting in longer-horizon tasks, only (Qwen2 VL 72B) exceeds 75%. Meanwhile, human performance in these tasks is **97**%.

Finding - Easier to reason on shorter horizons: We see a trend where all VLMs, except CogAgent, perform better on short-horizon than long-horizon plans. While the accuracy gain varies, some models, such as Llava 1.6 - 34B, show improvements of 32% over their long-horizon results. This highlights the persistent limitations of

Model	Short Horizon		Long Horizon		Avg
	Instruct	CoT	Instruct	CoT	
Qwen2-VL 2B	55.68	53.35	50.42	49.36	52.2
Qwen2-VL 7B	66.69	76.33	67.45	48.38	64.71
Qwen2-VL 72B	98.29	97.35	76.88	67.23	84.94
Minicpm	51.13	62.55	50.0	35.57	49.81
Minicpm-o	57.17	68.43	57.82	48.83	58.06
Phi 3.5	67.52	66.96	45.11	47.14	56.68
IDEFICS3	34.67	79.57	50.07	50.49	53.7
DeepSeek VL	51.28	57.52	43.26	45.8	49.46
Llava OV 0.5B	50.78	44.28	49.28	44.01	47.09
Llava OV 7B	56.04	76.46	50.11	52.29	58.72
InternVL2-MPO	37.47	86.97	54.53	59.74	59.68
Glm	54.67	61.33	49.77	50.45	54.06
Llava 1.6 7B	65.27	64.18	40.61	48.63	54.67
Llava 1.6 13B	50.0	62.49	50.0	45.7	52.05
Llava 1.6 34B	80.03	76.39	46.17	53.78	64.09
Ferret-UI-Llama	46.76	49.99	66.41	49.85	53.25
Ferret-UI-Gemma	59.32	6.18	49.03	5.95	30.12
UIX-Qwen2	50.82	62.67	40.11	46.72	50.08
CogAgent	50.0	35.65	50.0	43.23	44.72

Table 2.2: **Evaluation on State Prediction Task in Section 2.4.2.** We show the accuracy of VLMs on the short- and long-horizon splits. While overall performance is suboptimal, models show improved accuracy with shorter horizons, underscoring their limitations in multi-step reasoning.

Model	Shuffle Perturb.		Semantic Perturb.		Avg
	Instruct	CoT	Instruct	CoT	
Qwen2-VL 2B	54.02	22.29	72.41	27.58	44.08
Qwen2-VL 7B	57.24	50.57	90.57	71.72	67.53
Qwen2-VL 72B	81.83	75.4	95.63	90.11	85.74
Minicpm	52.39	37.01	85.97	64.36	59.93
Minicpm-o	48.96	39.31	80.91	75.17	61.09
Phi 3.5	30.11	32.41	69.19	60.45	48.04
IDEFICS3	40.69	36.09	84.13	60.0	55.22
DeepSeek VL	24.13	23.67	26.89	28.73	25.85
Llava OV 0.5B	24.13	12.18	25.57	15.4	19.32
Llava OV 7B	55.17	47.12	89.88	69.65	65.46
InternVL2-MPO	45.05	22.75	90.11	68.73	56.66
Glm	30.34	35.86	70.8	67.81	51.2
Llava 1.6 7B	26.95	30.02	38.7	47.11	35.7
Llava 1.6 13B	28.34	24.88	59.44	45.85	39.62
Llava 1.6 34B	39.86	43.41	85.94	73.27	60.61
Ferret-UI-Llama	23.67	20.68	22.98	21.65	22.24
Ferret-UI-Gemma	11.95	22.06	9.88	20.04	15.98
UIX-Qwen2	57.7	29.65	67.35	43.21	49.48
CogAgent	23.41	23.9	26.25	26.72	25.07

Table 2.3: **Evaluation on Plan Selection in Real Web Environment. in Section 2.5.1** We show the accuracy of VLMs with two types of MCQ choices (Section 2.5.1). Many VLMs perform under 50%. Models perform better to distinguish semantically different options than options with shuffled orders.

Model	Instruct	CoT	Avg
Qwen2-VL 2B	37.83	35.47	36.65
Qwen2-VL 7B	49.56	47.84	48.7
Qwen2-VL 72B	88.75	76.25	82.5
Minicpm	33.77	33.82	33.8
Minicpm-o	47.76	37.35	42.56
Phi 3.5	32.85	26.18	29.52
IDEFICS3	46.2	55.87	51.04
DeepSeek VL	18.33	28.17	23.25
Llava OV 0.5B	19.45	21.65	20.55
Llava OV 7B	37.55	38.41	37.98
InternVL2-MPO	57.69	49.15	53.42
Glm	46.77	42.23	44.5
Llava 1.6 7B	29.33	40.25	34.79
Llava 1.6 13B	26.09	31.96	29.02
Llava 1.6 34B	40.35	29.72	35.04
Ferret-UI-Llama	20.83	22.61	21.72
Ferret-UI-Gemma	11.72	13.28	12.5
UIX-Qwen2	22.65	14.46	18.56
CogAgent	26.34	21.08	23.71

Table 2.4: **Evaluation on Plan Selection in Edge Cases in Section 2.5.2** We observe a similar trend to Table 2.3 here. Moreover, CoT consistently hurts the model performance compared to the direct instruction.

current VLMs and raises concerns about their reliability due to inconsistencies across different scenarios.

2.5 Plan Assessment Analysis

In this section, we investigate more complex reasoning capabilities: *Can VLMs select the right plan to achieve a specific goal under real-world and challenging synthetic edge-case scenarios?* While the previous temporal analysis focuses on understanding

the relationship between steps in a sequence, plan selection requires using this understanding to choose the optimal course of actions. This transition represents a critical step in evaluating whether VLMs can function as reliable web agents. Specifically, each test example consists of a goal description, a screenshot of current webpage state, and possible action plan candidates presented in an MCQ format. The model must identify the single correct plan to achieve the desired goal.

To analyze whether VLMs can reliably reason and select plans across diverse contexts, we conduct two tests. First, we evaluate real-world web use cases using tasks and scenarios from actual website data. Second, we leverage synthetic edge cases designed to challenge VLMs in controlled environments, offering a more comprehensive assessment of their reasoning abilities.

2.5.1 Real Web Environment Study

Experiment Setup: To test VLMs in real-world web environments, we evaluate whether VLMs can choose the correct plan from four MCQ options, given the current webpage screenshot and task instruction. Specifically, we curate the real-web dataset from the MM-M2W dataset [72], where each data point consists of a task instruction and a human-annotated trajectory to solve that task. To leverage this data, we first perform a few modifications. We begin by removing samples where the action sequence is longer than 10 steps to align with VLMs' reasoning limitations [40, 42]. Then, to make the dataset's provided action descriptions (e.g., "Target element: [button] Search, Action: CLICK") more natural to VLMs (e.g., "Click on the search button"), we paraphrase them with GPT-4o [45]. To ensure data quality, we manually inspect and refine each sample.

We construct the final test MCQ questions as follows: For each task instruction g, we sample the web page screenshot (i.e., state) i_t from its solution trajectory, where i_t is 4-5 action steps away from reaching the task's end state (or goal completion). These action steps will now serve as the ground-truth correct answer for MCQ. By limiting the reasoning to the next 4–5 steps, we simplify the task for a reasonable chance of success. Incorrect MCQ options are then populated using the following two strategies:
∘ **Shuffling Perturbation:** Given that a plan is a sequence of actions where the next action depends on the previous actions, this configuration creates incorrect options by violating the sequential flow of actions. Particularly, we create incorrect options by entirely or partially reversing the correct action sequence, e.g., for the task "Buy a Shampoo on Amazon" and the plan ["Visit Amazon", "Search Shampoo", "Add to

Cart", "Order"], an incorrect option would be ["Order", "Add to Cart", "Visit Amazon", "Search Shampoo"] (see Figure A.20c).

◦ **Semantic Perturbation:** Given an action plan for a task, this configuration generates incorrect options by inserting irrelevant actions from a different task into the current task's action plan. Particularly, for the task "Buy a Shampoo on Amazon" and correct action plan ["Visit Amazon", "Search Shampoo", "Add to Cart", "Order"], an incorrect option would be ["Visit Amazon", "Search flights to New York", "Add to Cart", "Order"] (see Figure A.20d). With manual validation, we obtain 435 high-quality samples per strategy, as detailed in **??**.

Result - Lack of genuine understanding of plans: The experiment results are presented in Table 2.3. Under the Shuffling Perturbation setting, while most VLMs outperform random guess performance (i.e., 25%), their accuracy remains significantly lower than that of human raters (i.e., 90%), with only 3 models exceeding 50%. In contrast, in the Semantic Perturbation setting, many models achieve higher accuracy over 50%, and every model surpasses its own performance under Shuffling Perturbation (due to intuitiveness, we skip human study for the generally easier Semantic Perturbation setting). This suggests that while VLMs can differentiate plans by assessing whether each of their action is contextually appropriate for the given task (shown in Section A.8), they struggle to compare plans based on the difference between the temporal ordering of actions, which is critical for planning. Consequently, we anticipate a need to explicitly model action sequences and state changes in VLMs.

2.5.2 Edge Case Test Study

Motivation: Real-world data provides valuable insights but often lacks the diversity of scenarios that agents may encounter during web browsing. For instance, datasets like MM-M2W include human-annotated trajectories that typically start from website homepages, have human labels for each action, as well as represent "perfect" trajectories with minimal navigation errors and where each webpage's state is likely an 'ideal state' for possible actions (e.g., to fill a form, we define an 'ideal state' is an empty form). However, in practice, edge cases such as random initializations (e.g., forms have pre-filled fields) or error states (e.g., checking the wrong checkbox) are common. While humans can recover from these, we explore whether VLMs can do the same. Thus, motivating our development of synthetic edge case data with controlled variations, focusing on recoverable errors.

Experiment Setup: We again leverage MiniWob++, focusing on UI elements

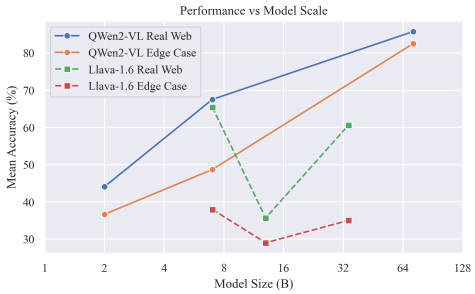


Figure 2.3: **Performance v/s Model Scales.** We plot average performance of Qwen2-VL and Llava-1.6 family over all settings in real web environment (Section 2.5.1) and edge case experiments (Section 2.5.2). We find that scaling does not automatically

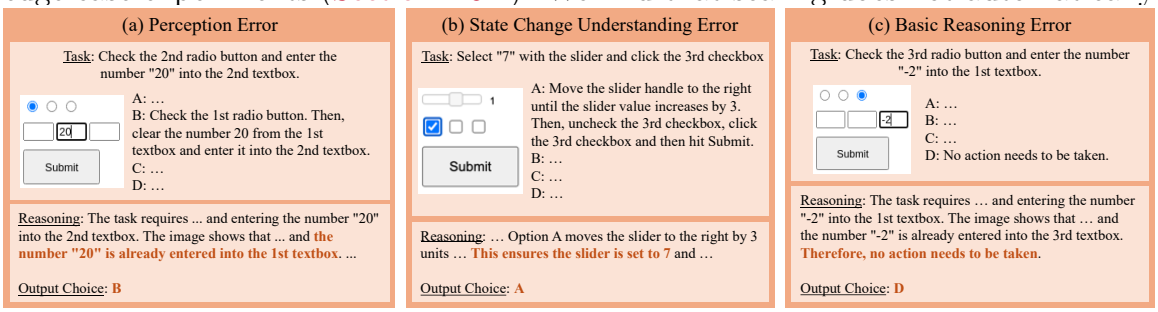


Figure 2.4: **Error case study.** We demonstrate three common error paradigms observed in our experiments.

including checkboxes, textboxes, sliders, radio-buttons, and buttons. For a given task (e.g., "Check the 1st and 2nd checkbox"), we algorithmically perform actions that modify the empty form state to introduce random initializations or errors (e.g., toggling the 3rd checkbox), and then generate four action plans, only one of which correctly solves the task (see Section A.1). To ensure high-quality data, we manually validate the correctness of the questions, and the existence of a single correct option.

Result - Worse performance on edge cases: From Table 2.4, we observe that most models perform significantly poor, with accuracy below 50% for both Instruct and CoT variants. Meanwhile, human raters achieve 99.2% accuracy for these tasks. The performance is even poorer for GUI-specific models (Ferret-UI, UIX-Qwen2, CogAgent), with a maximum accuracy of only 26%. Moreover, we observe that the highest-performing Qwen2VL 72B model suffers a 12% performance drop in its CoT variant. This reinforces concerns if models rely on spurious correlations rather than genuine understanding to arrive at correct answers.

2.6 Discussion and Analysis

In this section, we highlight the consistent trends observed across our experimental settings.

Do these skills lead to task success? We compare the task success of three models— Qwen2-VL-7B, LLaVA-1.6, and Phi-3.5— from the Mind2Web-Live benchmark [48], and their performance across our planning skills. As detailed in Section A.3, model's capabilities in our planning skills are correlated with the task success rate.

Is CoT reasoning always helpful? In Tables 2.1 to 2.4, we explore the impact of CoT reasoning on the performance in each experiment. Interestingly, we did not observe sufficient gains resulting from CoT. Specifically, while some models occasionally showed reduced performance due to CoT, others consistently struggled to adopt it effectively.

Are bigger models better than their smaller versions? From Figure 2.3, we find that while Qwen2-VL family improves in performance, Llava 1.6-34B is not significantly better than its smaller counterparts, contrary to the findings of [33] - scaling improves perception-related web tasks. Thus, it is crucial to investigate how factors beyond scaling—such as architecture and pre-training—impact VLMs' reasoning and planning abilities on the web.

Do GUI-specific models outperform general VLMs? Although previous studies show that GUI-specific models outperform generalist VLMs in GUI navigation and comprehension tasks [7, 52], they significantly underperform on our tests in Tables 2.2 to 2.4 (last three rows). Similar to [33], we found this behavior is likely due to over-fitting to the GUI settings, which affects their general reasoning and instruction-following capability. We support this using qualitative examples of GUI-Models in Section A.8, demonstrating their poor output quality.

Case Study: We present examples in Figure 2.4 to better understand the errors made by the model. This section focuses on the results of Qwen2-VL 72B in Section 2.5.2, with additional results provided in Section A.8. Our analysis reveals three most common errors: 1). Perception Errors (Figure 2.4a), where incorrect perception of details becomes bottleneck in reasoning, 2). State Change Understanding (Figure 2.4b), where the model fails to imagine future states, 3). Basic Reasoning (Figure 2.4c), where the model fails to make straightforward logical inferences.

2.7 Conclusion

In this work, we introduce the first evaluation frameworks aimed at understanding the fine-grained skills required for VLMs to plan in web environments. While prior research has largely focused on webpage perception and task success metrics, such evaluations overlooks the necessity in thoroughly assessing the planning capability

required for reliable web agents to plan. To address this gap, we carefully design critical questions encompassing planning skills such as temporal relations over state trajectory, temporal prediction of future states, and plan assessment for identifying effective plans. By carefully re-purposing existing data, generating synthetic data and validating by humans, we create four high-quality tasks and benchmark 19 state-of-the-art VLMs.

Our findings reveal that current VLMs are far from achieving human-like planning capabilities, and hardly above random guessing in many evaluations, indicating fundamental deficiencies in reasoning about web dynamics. Specifically, we observe that models struggle with temporal reasoning and plan assessment, highlighting their limitations in synthesizing sequential web information. Additionally, common strategies, such as CoT reasoning, increasing model scale, and training on GUI-specific datasets, do not consistently lead to improvements in fine-grained planning tests. These findings underscore the pressing need for more sophisticated training and evaluation paradigms.

Chapter 3

Fine Grained Diagnosis of Grounding Capability

3.1 Introduction

Graphical User Interface (GUI) automation [44] is an emerging application of Large Language Models (LLMs) [51, 57] and Vision Language Models (VLMs) [2, 46]. Given a UI and a user instruction, the goal is to perform the appropriate action(s) to fulfill the instruction. A key challenge in this setting is *GUI grounding*—predicting the exact coordinates or bounding box where an action (e.g., click) should occur. This task is particularly difficult because: (1) it demands high precision, as small coordinate errors can trigger incorrect interactions and lead to unrecoverable states; and (2) the variability in UI layouts and instructions across platforms (web, mobile, desktop) exacerbates the problem. To tackle these challenges, recent works generally train large-scale GUI grounding models [13, 50, 61], and develop diverse grounding benchmarks [6, 26].

Despite this, prior work remains heavily skewed toward web and mobile environments [9, 22, 53], limiting the development and analysis of desktop agents. This imbalance largely stems from the ready availability of HTML and XML, which allows easy extraction of UI element–coordinate pairs. In contrast, desktop environments like Windows typically provide only UI screenshots, making dataset creation dependent on manual annotation or proprietary VLMs—both hard to scale and prone to hallucinations. Consequently, desktop-specific GUI grounding benchmarks and datasets are scarce, with only a few recent works [3, 18, 62] explicitly targeting this domain.

Beyond these limitations, a crucial yet often overlooked aspect of GUI grounding

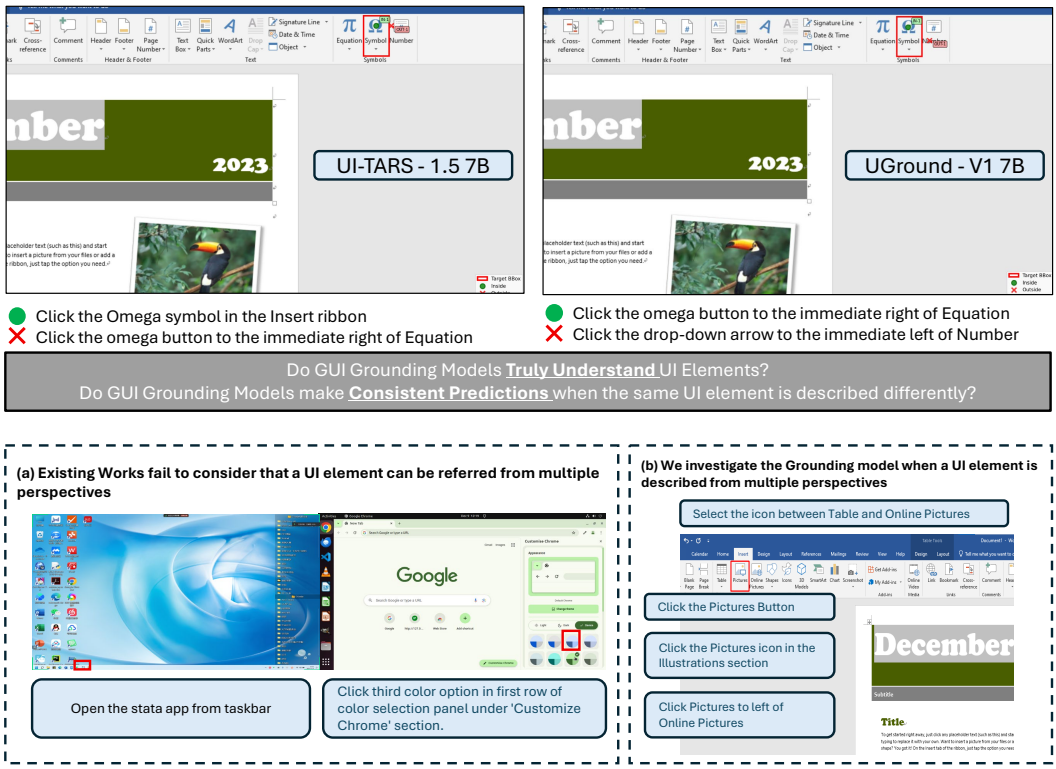


Figure 3.1: GUI grounding models struggle when UI elements are described from different perspectives (**Row 1**). Existing evaluations overlook this phenomenon by measuring accuracy on a single "best"/"common" reference (**Row 2**). Instead, we estimate the sensitivity (or consistency) of model predictions across different but valid references in our **GUI Grounding Sensitivity Benchmark**.

research is understanding the model's ability to handle diverse user instructions. Most prior works use either short (e.g., "close button") [6] or simplified instructions (e.g., "Click the Pictures icon") [26], which fail to capture the complexity of real-world interactions. Although recent efforts [34, 43, 63] incorporate more implicit expressions (e.g., "click the left icon"), they do not assess whether models truly comprehend the multifaceted ways users might describe an element. This is partly due to their focus on generating a single "best" instruction per element, which often results in the most common or obvious phrasing. In practice, a UI element can be referenced in multiple valid ways—by visual traits, relative position, or surrounding context. As such, judging grounding models by their ability to ground a UI element for a single reference yields an incomplete view. Instead, to assess whether GUI grounding models genuinely grasp element identification, we must ask: **Are grounding model predictions consistent across diverse descriptions of the same UI element?**

We focus on desktop environments – a challenging yet underrepresented setting – and

propose the first work to investigate the sensitivity of grounding models to multiple ways of referring to the same UI element. Since manually annotating multiple diverse instructions per UI element is impractical, we design an automatic pipeline for data generation and validation. This pipeline produces high-quality image-instruction sets, which are subsequently human-verified to construct our benchmark: **GUI Grounding Sensitivity Benchmark**.

We evaluate **12** GUI grounding models and report reasonable sensitivity in their predictions. Importantly, we find that although many recent models Qwen-2.5 [2], Jedi [63] outperform on popular GUI benchmarks they lag behind earlier models like UGround-V1 [13], OS-Atlas [61]) on our proposed benchmark. The aforementioned variation in sensitivity across models suggests that for a given GUI grounding model, certain instructions or contextual relationships are more challenging than others. This leads us to our next research question: developing a diagnostic pipeline that automatically generates diverse instructions, uses model feedback, and iteratively refines them to expose failure cases of the model. We call this system – **GUI Grounding Diagnosis Agent**. In summary:

- We propose the first framework for evaluating the **sensitivity** of GUI grounding models to diverse references of the same UI element.
- We focus on the underrepresented **desktop** setting and propose an automatic data generation pipeline, followed by human verification, to develop the **GUI Grounding Sensitivity Benchmark**.
- We present a **GUI Grounding Diagnosis Agent** to automatically extract failure cases of grounding models.

3.2 Related Work

Desktop GUI Agents: Vision Language Model (VLM) agents show strong potential in automating GUIs using natural language user instructions (e.g., "install apps", "close settings") and environment representations – text, images, or both [9, 15, 19, 22, 52, 53, 72]. While HTML, XML, or other text inputs are helpful, it is critical to improve GUI agents for image-only settings without access to such information (e.g desktop). Our work focuses on visual GUI grounding for desktop agents, which demands robust perception and GUI understanding. Prior works [6, 13, 50] tackle this by collecting large-scale image-position pair data, however their focus is

primarily mobile and web. UGround [13] collects 1.3M web-only GUI screenshot data, while GUICourse does 82K samples spanning web and mobile interfaces. OS-Atlas further curates 2.23M cross-platform screenshots (web, mobile, desktop), though their dataset remains web-heavy, with only 54K desktop-specific samples. Other approaches like UI-Tars [50], combine closed-source and open-source data; however, they still report the desktop samples contribute the least to the open-source split. Although a few works [18, 63, 64] introduce desktop-focused datasets covering common usage scenarios, there is still a pressing need for broader attention to this domain. Thus, steering our focus towards the limitations and sensitivities of Desktop Grounding models.

GUI Grounding Evaluation: While recent models show improvements on existing benchmarks [6, 26], the lack of rigorous desktop-specific testing in those evaluations limits our ability to accurately gauge their effectiveness in real desktop environments. In addition, these evaluations face two challenges: 1) they either test short or straightforward explicit/implicit instructions that reflect the most common ways to refer to elements[6, 26], 2) they only test the effect of image variations on GUI grounding [26, 69, 71]. Their fundamental limitation is that none consider the fact that there are often many ways to refer to a single UI element within a GUI screenshot. More precisely, for every GUI screenshot, a single UI element has multiple unique identifiers – visual, position, and so on. In other words, while GUI grounding models understand a few instructions (or common ways) to refer to a UI element, there is no guarantee that it will understand other ways to refer to the same element. Our analysis (Fig. 3.1) supports this observation, showing that current models frequently fail to identify a UI element when it is described from varied perspectives. To address this, we present the first effort to understand the sensitivity of grounding models to diverse referring expressions for the same UI element. Based on these insights, we introduce a novel agentic diagnostic pipeline that automatically uncovers instruction formulations where grounding models fail, enabling robust evaluation.

3.3 GUI Grounding Sensitivity

3.3.1 Background and Motivation

Given an image I and a UI element e, we can extract multiple visual and position identifiers:

$$\begin{aligned} e_v &= \{v_1, v_2, v_3, \ldots, v_n\} \\ e_p &= \{p_1, p_2, p_3, \ldots, p_m\} \end{aligned} \tag{3.1}$$

where e_v is the set of visual identifiers v_i and e_p is the set of position identifiers p_i. We define a unique identifier for e as any individual identifier (visual or position) or combination of v_i and p_i that matches only e—i.e., no other element can be described using it. Consequently, multiple identifiers can uniquely identify e:

$$\begin{aligned} e = \{ &{v_1}^u,\ {v_2}^u,\ \cdots,\ {p_1}^u,\ {p_2}^u,\ \cdots, \\ &(v_i,\ p_k)^u,\ \cdots,\ (v_j,\ p_k)^u,\ \cdots \} \end{aligned} \tag{3.2}$$

where v_i^u and p_i^u are individual unique identifiers, while $(v_i, p_k)^u$ denotes a combination unique identifier. Given that we aim to investigate the sensitivity of GUI grounding models against various ways of referring to e, our goal is to develop a data generation pipeline that reliably produces K distinct unique identifiers.

3.3.2 Data Generation

Image Selection: We select the desktop screenshots (particularly windows) from ScreenSpot [6] and OmniAct [21], yielding 173 images. Then, we filter out screenshots that were too easy to understand. Particularly, we prompt VLM[1] to skip images with too few UI elements, a static, or minimally interactive screen requiring no meaningful GUI reasoning.

UI Element Extraction: For each image I, we first extract the bounding box (b_i) and OCR text (t_i) of all elements E using OmniParser [37]:

$$E = \{e_i\}_{i=1}^N = (b_i, t_i)\}_{i=1}^N = \text{OmniParser}(I) \tag{3.3}$$

To allow for sensitivity in model predictions, we focus on UI elements that can be challenging to ground. We employ some heuristics to guide this selection - 1) visual similarity to other elements, 2) positional ambiguity – grouped closely with similar

[1] all our pipelines use GPT-4.1 as the VLM

elements, overlapping, and 3) functional overlap – elements appear to serve similar purposes. We then ask VLM to sample relatively challenging UI elements from E:

$$H = VLM(E) = \{h_i\}_{i=1}^{N} \tag{3.4}$$

where h_i are the confusion risk scores for each element. We then select the top-D elements $E_D \subseteq E$ with the highest scores.

Further, it is also crucial that our pipeline generates K accurate instructions for a UI element e within I. However, selecting e based solely on confusion risk scores can be unreliable, as VLMs often struggle to ground elements that are small or poorly visible (see Fig B.7). To address this, we incorporate two cues: the element's visibility score (v_i), which reflects how clearly it is visible on the screen (obtained by prompting the VLM), and its bounding box area (area(b_i)). We select the target element e from E_D as the one maximizing $v_i \times \text{area}(b_i)$.

Before proceeding, we manually inspect a random subset of images and observe that sometimes E_D is problematic. Specifically, several bounding boxes generated by OmniParser are incorrect—either empty, containing partial or multiple UI elements in a single box, or capturing non-interactive regions (like plain text). We prompt the VLM to filter out such cases, resulting in a set of **100** high-quality images.

Candidate Instruction Generation: Now, we ask the VLM to generate a diverse set of referring instructions for every (e, I) pair by incorporating multiple visual and positional identifiers (Eqn. 3.1). Particularly for positional cues, the VLM is instructed to describe e based on its immediate neighbors, broader local context, or the section it belongs to. While this approach yields many candidate instructions, we observe a notable number of hallucinations, consistent with findings in [13]. Thus, unlike prior work, which assumes a single instruction per element, our setting necessitates a robust verification pipeline, described next.

3.3.3 Data Validation

Overall Correctness Verification: Our first validation step is a simple check: we highlight the element e using a bounding box and ask the VLM to check whether the instruction correctly refers to e. Particularly, the VLM is instructed to assess whether: 1) the elements mentioned in the instruction are visible, 2) the described visual details are accurate, and, 3) the stated semantic relationships with surrounding elements are correct.

Position Correctness Verification: Manual inspection reveals that while the

previous step filtered some positional errors, it struggled with others. This is partly because VLMs tend to overgeneralize spatial terms like "next to" or "between," or lack precise interpretations for relations such as "left" or "above"—none of which were explicitly enforced earlier. To address this, we define strict spatial semantics (e.g., "left" refers to horizontal left; "next" means immediate adjacency). For instructions that pass the earlier check, we now perform explicit position verification by guiding the VLM with clear definitions of terms like "above," "below," "left of," and "right of," and filter out any spatially imprecise instructions.

Uniqueness Verification: In this step, we verify if the remaining instructions uniquely refer to the UI element e. That is, given an instruction i that refers to the UI element e, we ask the VLM to find (or count) if there exist other UI elements in image that can reasonably/substantially match the instruction.

3.3.4 Data Transformation

After manual inspection, we found that most generated instructions were acceptable, with the exception of a small subset that failed to uniquely identify e. Instructions containing visual cues (e.g., calendar icon) or specific positional references (e.g., first list item) were typically reliable, whereas those relying solely on relative position (e.g., icon between list and sidebar) often lacked uniqueness (see Fig B.8) . Rather than repeatedly attempting to fix such cases, we hypothesize that making such instructions more specific (by adding visual details) is a more effective strategy for ensuring correctness.

We begin by isolating the set of instructions that refer to e purely through its relative position to nearby elements. Next, we evaluate whether these positional relationships are sufficient to uniquely identify e. To do this, we crop the image around e to include its local context—that is, neighboring elements $\mathcal{N}(e)$ located within a distance threshold d_{th}:

$$\mathcal{N}(e) = \left\{ j \;\middle|\; \|\mathbf{c}_j - \mathbf{c}_e\|_2 \leq d_{\text{th}} \right\}$$

Here, $\mathbf{c}_j$ denotes the center of the element j. We use $\mathcal{N}(e)$ to compute the local crop I_{local}:

$$I_{\text{local}} = I\Big[\min_{j \in \mathcal{N}(e)} x_{1,j}, \quad \min_{j \in \mathcal{N}(e)} y_{1,j}, \quad \max_{j \in \mathcal{N}(e)} x_{2,j}, \quad \max_{j \in \mathcal{N}(e)} y_{2,j} \Big] \tag{3.5}$$

We then use I_{local} to prompt the VLM to assess whether the instruction uniquely identifies e (similar to Sec. 3.3). This yields a set of successful and failing instructions ($r_i^s,\ r_i^f$). We transform r_i^f into r_{tr} by augmenting it with visual identifiers ($\{v_1, v_2, v_3, \ldots, v_n\}$), resulting in the final output instructions r_o:

$$\begin{aligned} r_{tr} &= \Big\{ (v_1, r_1^f),\ (v_2, r_2^f),\ \ldots,\ (v_n, r_n^f) \Big\} \\ r_o &= \{r_s, r_{tr}\} \end{aligned} \tag{3.6}$$

Finally, we perform a manual review of the generated instructions, resulting in **1,712** validated instructions across all images — approximately 17 instructions per UI element. Refer to Fig B.1 for the visual illustration of the entire data generation pipeline.

3.3.5 Evaluation Metrics

In this section, we describe our evaluation measures to assess the proposed sensitivity of grounding models M.

Mean Distance (s_{mean}): Given image I, UI element e, and K different perspective instructions ($\{r_k\}_{k=1}^K$), M predicts K output coordinates $\{c_k\}_{k=1}^K$. We compute their centroid (c_{centroid}), and then define the sensitivity to understand e as the mean distance:

$$d_{mean} = \frac{1}{K} \sum_{k=1}^{K} \frac{|c_k - c_{\text{centroid}}|}{diag(b)} \tag{3.7}$$

where $diag(b)$ is the diagonal of the bounding box b enclosing e. Finally, we average d_{mean} across input images ($I \in \mathcal{I}$) to obtain s_{mean}:

$$s_{mean} = \frac{1}{|\mathcal{I}|} \sum_{I \in \mathcal{I}} d_{mean}^I \tag{3.8}$$

Worst Distance (s_{worst}): We define the worst-case sensitivity as the maximum distance of any predicted coordinate c_k from c_{centroid}:

$$d_{worst} = \max_k \frac{|c_k - c_{\text{centroid}}|}{diag(b)} \tag{3.9}$$

Then, we average d_{worst} across input images ($I \in \mathcal{I}$) to obtain s_{worst}:

$$s_{worst} = \frac{1}{|\mathcal{I}|} \sum_{I \in \mathcal{I}} d_{worst}^I \tag{3.10}$$

Instruction Out-of-Box Rate (s_{oob}): This metric measures the number of instructions where the output coordinate c does not lie inside the bounding box b containing e:

$$s_{oob} = \frac{1}{\sum_{I \in \mathcal{I}} K_I} \sum_{I \in \mathcal{I}} \sum_{k=1}^{K_I} 1\{c_k \notin b_k\} \tag{3.11}$$

3.3.6 Experiments

Models Used: We evaluate **12** models: OpenAI Operator [47], UI-Tars-1.5-7B [50], UGround-V1-2B/7B [13], OS-Atlas-7B [61], ShowUI-2B [28], Jedi-3B/7B [63], Qwen-2.5-VL-7B/32B [2], InfiGUI-R1-3B [35], UI-R1-E-3B [38].

Performance on GUI Grounding Sensitivity Benchmark: We report our results in Table 3.1 (Base Column), which also includes model performance on ScreenSpot(SS)-Pro [26], a widely used GUI grounding benchmark. Our findings indicate that the Operator is the least sensitive model, while the Jedi family has the highest sensitivity. Among open-source models, UI Tars 1.5 7B and UGround-V1-7B perform best, likely due to large-scale training and curated datasets. We observe that sensitivity generally improves with model size – UGround-V1-7B and Qwen-2.5-VL 32B outperform their smaller versions.

Interestingly, reinforcement learning-trained models [1, 54] like InfiGUI-R1-3B and UI-R1-E 3B are competitive with 7B-scale models and sometimes surpass OS-Atlas-7B, Qwen-2.5-VL 7B.

Finally, consistent with our hypothesis, we find that strong performance on existing GUI benchmarks does not imply that models better understand UI elements from multiple perspectives. For example, although UI Tars 1.5 7B outperforms Operator on SS-Pro, it lags behind in our results. Similar trends are noted for other pairs like UGround-V1-7B vs. Qwen-2.5-VL 32B and OS-Atlas-7B vs. Qwen-2.5-VL 7B, and so on.

Performance over different Image Inputs: Here, we analyze the sensitivity of grounding models by varying common ways in which UI elements appear within GUI screenshots. For the first setup, **Local Crop**, we use Eq.3.5 to zoom-in and generate a crop around the UI element. In the second setup, **Window-in-Background**, we simulate realistic scenarios where applications appear in smaller windows over desktop backgrounds by overlaying the original screenshot onto a background image (see Fig. B.4). Results in Table 3.1 show no clear benefit of local cropping on sensitivity. This is because, although cropping may improve visibility of target element, it does not necessarily help the model understand the semantic relationship between elements,

Model	Base			Local Crop			Window-in-Background			Language			SS-Pro
	$s_{\text{mean}} \downarrow$	$s_{\text{worst}} \downarrow$	$s_{\text{oob}} \downarrow$	$s_{\text{mean}} \downarrow$	$s_{\text{worst}} \downarrow$	$s_{\text{oob}} \downarrow$	$s_{\text{mean}} \downarrow$	$s_{\text{worst}} \downarrow$	$s_{\text{oob}} \downarrow$	$s_{\text{mean}} \downarrow$	$s_{\text{worst}} \downarrow$	$s_{\text{oob}} \downarrow$	Acc. $\uparrow$
Operator	0.3217	1.3342	**13.52**	**0.2364**	0.8817	**14.02**	1.0200	3.5524	**33.06**	0.1632	0.5346	2.96	36.6
UI-TARS-1.5-7B	**0.3021**	1.2723	20.61	0.2406	**0.8683**	33.24	**0.9063**	**2.5977**	38.94	0.1017	0.4773	2.50	**61.6**
Qwen2.5-VL-32B	0.5059	1.6707	28.44	0.3790	1.1481	36.36	1.4885	3.8425	54.82	0.1316	0.3727	2.73	47.6
Qwen2.5-VL-7B	0.4548	1.5358	41.20	0.3613	1.0161	46.76	1.7426	4.3578	73.12	0.1101	0.3324	4.82	27.6
UI-R1-E-3B	0.4207	1.4895	26.75	0.2503	0.7685	42.76	1.0909	3.4827	46.10	0.1043	0.2837	2.28	33.5
InfiGUI-R1-3B	0.3854	1.3938	26.52	0.2536	0.8966	41.53	1.2301	3.6199	44.00	0.1023	0.4426	2.42	35.7
UGround-V1-7B	0.3176	**1.1512**	22.55	0.2392	0.8616	24.82	1.0558	3.9273	39.92	**0.0718**	0.3212	**1.28**	31.1
UGround-V1-2B	0.6218	1.9334	39.54	0.4074	1.1863	38.38	1.4080	4.5210	54.90	0.0813	0.3618	3.07	25.0
OS-Atlas-7B	0.4641	1.6698	29.96	0.2747	0.8819	27.57	1.3227	3.8103	54.09	0.0967	**0.2029**	1.32	18.9
ShowUI-2B	0.6423	2.1535	47.20	0.3630	1.0758	45.79	1.1918	3.7318	71.15	0.2156	0.8340	6.56	7.7
Jedi-7B-1080p	1.0426	2.3546	77.39	0.4955	1.2417	79.21	1.2737	2.8370	87.94	0.7152	2.0161	7.58	39.5
Jedi-3B-1080p	0.9158	2.0822	76.20	0.5062	1.0818	74.47	1.1964	2.6510	81.67	0.7365	1.9309	13.97	36.1

Table 3.1: **Main Results**. (a). Sensitivity scores ($s_{mean}, s_{worst}, s_{oob}$) of **12** grounding models (**Base** Column) reveal that strong performance on existing benchmarks (**SS-Pro** Column) does not imply models truly understand a UI element. (b) Enhancing target element visibility via zoom-in (**Local Crop**) does not improve sensitivity, while increasing image complexity (**Windows in Background**) worsens it. (c) Models perform well under linguistic perturbations (**Language**), suggesting that sensitivity mainly stems from inherent limitations in understanding UI elements. Results for SS-Pro are taken from existing papers.

disambiguate similar-looking components, or overcome inherent biases and limitations. Finally, we observe that increased visual complexity of the Window-in-Background worsens sensitivity.

Performance over language variations: Here, we explore grounding sensitivity to linguistic variations, including spelling errors, synonym replacements, and paraphrasing. To ensure that performance changes are attributed solely to these variations, we apply perturbations to instructions that most clearly and directly describe the UI element – cases where the model is expected to perform confidently. Results in Tab. 3.1 (Language) show that most models, with the exception of the Jedi family, exhibit robustness to linguistic variations. Moreover, models exhibit substantially lower sensitivity compared to results in Table 1 (Base). For instance, Uground-V1-7B has $s_{mean} = 0.3176$ in Tab. 3.1 (Base) vs. 0.07 in Language; similarly, OS-Atlas-7B has 0.46 vs. 0.09, and Qwen2.5-VL-32B has 0.50 vs. 0.13.

3.4 Automated Grounding Diagnosis

Motivation: Results from the previous section highlight that state-of-the-art GUI grounding models exhibit sensitivity when a UI element is described from multiple perspectives. That is, some instructions or spatial relations are more difficult for the model to interpret than others. This naturally motivates our exploration towards an agentic pipeline that can iteratively explore instructions, understand model's behavior

on those instructions, and automatically identify cases that lead to grounding failures. In this section, we present our pipeline, called **Automatic Grounding Diagnosis Agent**, which comprises three key components.

Step I - Seed Instruction Generation: To initiate our agentic loop, we first obtain K seed instructions $(R_1, R_2, \ldots, R_K)$ for a UI element e using Sections 2.2–2.3. We then predict output coordinates o_c^k for each R_k using the grounding model M, and record whether o_c^k lies within the bounding box b containing e.

Step II - Diagnosis Plan Generation: Next, we use the instructions where M succeeds (i.e. ${o_c}^k$ is inside b) to devise a diagnosis plan that could fail M. We begin by asking the VLM to analyze previous successful instructions, and come up with reasons r that made those instructions easy to ground:

> ...
> - Reason why instruction did not introduce ambiguity.
> - Highlight specific cues/phrasing that made disambiguation easy.
>
> ...

We then use r and ask VLM to generate a plan P which results in instructions that can potentially fail M. To ensure VLM generates P that is grounded in realistic challenges faced by M, we provide a list of heuristics to consider in the generated strategy:

> Your instruction Generation Strategy may include:
> - **Spatial phrasing traps** with nearby elements
> - **Visual lookalike confusion** based on color, size, iconography
> - **Functional overlaps** with elements that do similar things
> - **OCR proximity issues** for ambiguity with nearby text

Note that even though the aforementioned step involves two logical parts, we find that incorporating them in a single prompt is sufficient to achieve the goal.

Step III - Diagnostic Instruction Generation: We now require A to generate diagnostic instructions $(R_1^d, R_2^d, \cdots, R_N^d)$ that are different and appropriately incorporate P.

```
{
 "improvement_strategy": "how you used the
Plan to craft harder instructions",

 "generated_instructions": [
   {
    "instruction": "user instruction #1",
    "reason": "brief explanation"
   },
   ...
 ]
}
```

Since every R_k^d may not fail the model (i.e. ${o_c}^k$ is outside b), we filter those that achieve the intended goal. Then, we loop through Steps II, III for a maximum of $T = 10$ iterations or until $S = 5$ desired instructions are obtained.

Step IV - Instruction Selection: We find that some of the S generated instructions may contain incorrect details. However, our goal is not to collect S failing cases, but to identify the most accurate and effective one. Furthermore, since each instruction may include a different detail of the UI element, a detail missed or misrepresented in one may be correctly captured in another. This motivates a joint evaluation: instead of assessing correctness in isolation, we verify every instruction in the context of other instructions. Specifically, we input the image and complete instruction set into the VLM, allowing it to consider details from all instructions to inform its decisions. In cases where no instruction is deemed correct, we record no failing instruction for that image.

Next, multiple instructions may pass the previous step. In the final step, we select the best one—defined as the instruction with the fewest identifiers that could be confused with other elements. To do this, we ask the VLM to describe all UI elements that loosely match each instruction and choose the one with the fewest matches. For example, given "click the photos icon in the top left," a photo icon in the bottom right would still be counted as a loose match.

```
{
 "number": [number],
 "elements": [
  {
   "description": "element description",
   "clarity_rating": "(1-5), how clearly does
   instruction refer to the element"
  },
  ...
 ]
}
```

3.4.1 Experiments

In this section, we apply our Diagnostic Agent on state-of-the-art GUI grounding models. Due to OpenAI costs, we select 50 images and two models—UGround-V1-7B and ShowUI-2B.

Performance of Diagnostic Agent: Given an input image, the diagnostic agent aims to produce a single instruction that causes the grounding model to fail. There are two possible outcomes: (1) the agent fails to find such an instruction—either by hitting the iteration limit or being filtered out during Instruction Selection; or (2) it succeeds in generating an instruction that triggers a grounding failure, however, instruction may be correct (i.e., it accurately refers to a UI element) or incorrect. Importantly, we do not expect the agent to succeed for every image. The goal is to uncover potential failure cases of the model, with an emphasis on the accuracy of these failures. Specifically, we define **Success Rate (SR)** as the percentage of correct instructions among all failure-inducing instructions. To ensure evaluation reliability, we manually annotate the generated instructions when computing SR, avoiding potential inaccuracies from VLM-based evaluation. Our results in 3.2 (a) show that our Diagnostic Agent achieves a high success rate (upto 84%) in generating valid instructions that fail the grounding model. We include example failure cases generated by the agent in Fig. 3.3.

Transfer to other GUI models: In this experiment, we test if the correct failing instructions for previous models potentially fail other GUI grounding models. Our metric here is Instruction failure rate, i.e, how many failed a given model. The results in Tab 3.2 demonstrate that failing instructions for one model can also lead to failure for other models.

Impact of different components: In this section, we ablate our core components – Diagnosis Plan Generation and Instruction Selection component. We use UGround-V1-7B as the reference model, and manually annotated Success Rate (SR) for each

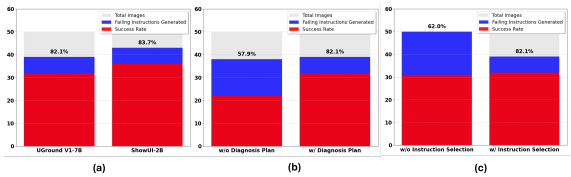
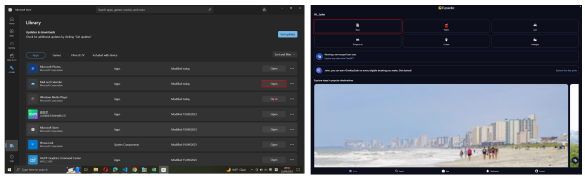


Figure 3.2: **Evaluation** of our Diagnostic agent. (a) **High Success Rate** of diagnostic instructions that fail UGround-V1-7B and ShowUI-2B, (b). **Diagnosis Plan Generation** improves the precision of generated instructions (c). **Instruction Selection** effectively filters out incorrect failing instructions.



(a). Use the Open button directly above Windows Media Player

(b). Pick the option in the upper row, furthest from Cars.

Figure 3.3: **Failing Instructions** generated for UGround-V1-7B. ▭ : Target UI element, • : Incorrect model prediction. Zoom in for clarity.

study.

From Fig. 3.2b, we observe that total number of failure-inducing instructions is comparable with and without the Diagnosis Plan Generation step. However, the accuracy of the generated instructions is significantly higher when this step is included, indicating the benefit of structured planning in instruction quality. Fig. 3.2c further shows that omitting the Instruction Selection step increases the number of failing instructions, but many are incorrect, resulting in a sharp drop in precision. With Instruction Selection, the agent achieves a higher accuracy—82.1% vs. 62.0%—demonstrating its effectiveness to filter out noisy instructions.

3.5 Conclusion

In this work, we focus on GUI grounding and highlight key limitations of existing approaches and benchmarks. Current methods largely overlook desktop environments, focus mainly on image variation, and evaluate grounding models using only a single "best" instruction per UI element. However, users may refer to UI elements through diverse visual and positional cues. Therefore, grounding models should produce consistent outputs across such variations, as single-instruction accuracy provides an incomplete view of model performance.

To address this, we design an automated data synthesis pipeline that generates

Model	UGround-V1-7B	ShowUI-2B
Operator	**12.50%**	**16.67%**
UI-TARS-1.5-7B	34.38%	25.71%
UGround-V1-7B	<u>100.00%</u>	44.44%
UGround-V1-2B	62.50%	69.44%
Qwen2.5-VL-32B-Instruct	43.75%	38.89%
Qwen2.5-VL-7B-Instruct	62.50%	66.67%
InfiGUI-R1-3B	34.38%	50.00%
UI-R1-E	37.50%	55.56%
OS-Atlas-Base-7B	43.75%	40.00%
ShowUI-2B	62.50%	<u>100.00%</u>
Jedi-3B-1080p	59.38%	77.78%
Jedi-7B-1080p	75.00%	83.33%

Table 3.2: **Instruction failure rate(↓)** of GUI grounding models on failing instructions generated for UGround-V1-7B and ShowUI-2B.

multiple instructions per UI element and release the GUI Grounding Sensitivity Benchmark to assess sensitivity to diverse descriptions. Our findings show that current models are indeed sensitive to instruction variation, and strong performance on existing benchmarks does not imply true UI understanding. We also observe robustness to language variation, while increasing image complexity worsens sensitivity.

Finally, recognizing that grounding models lack a comprehensive understanding of UI elements, we propose extracting the relations or descriptions they struggle with. We introduce the GUI Grounding Diagnosis Agent that automatically generates diverse instructions, incorporates model feedback, and iteratively refines them to expose failure cases. This diagnostic tool provides a new direction for evaluating and improving grounding models.

Chapter 4

Conclusion

This thesis investigated how computer-use agents can be evaluated beyond aggregate measures of task success. Although end-to-end benchmarks are useful for measuring whether an agent ultimately completes a task, they often provide limited insight into the capabilities that caused success or failure. A single unsuccessful trajectory may arise from incorrect perception, imprecise grounding, weak temporal understanding, flawed planning, or an inability to recover from an earlier mistake. Conversely, an agent may succeed on a benchmark through shortcuts or favorable task structure while still lacking the capabilities required for reliable deployment. The central argument of this thesis is therefore that progress toward robust computer-use agents requires fine-grained diagnostic evaluation of the underlying abilities that connect perception, reasoning, and action.

The first part of this thesis examined planning in web environments. We evaluated whether Vision-Language Models possess several prerequisites for effective planning: understanding the temporal order of webpage states, predicting how actions change an interface, and assessing whether a proposed sequence of actions will achieve a goal. Across these evaluations, current models showed substantial weaknesses. They frequently relied on spurious cues such as input order, degraded as action horizons increased, and struggled to distinguish plans that differed primarily in the ordering of otherwise relevant actions. They also performed poorly in controlled edge cases that required reasoning about non-ideal initial states or recovering from prior errors. Moreover, chain-of-thought prompting, increasing model scale, and leveraging GUI-specific models did not consistently resolve these limitations. These results suggest that strong visual recognition or next-action prediction does not necessarily imply an internal understanding of interface dynamics.

The second part of this thesis examined the grounding in desktop graphical user

interfaces. Standard grounding benchmarks usually associate each target element with a single instruction, although users can describe the same element through its appearance, function, relative position, or surrounding context. To address this limitation, we introduced a sensitivity-based evaluation that measures whether a grounding model produces consistent predictions across multiple valid descriptions of the same element. The resulting benchmark revealed that strong performance on conventional grounding benchmarks does not guarantee robust understanding of UI elements. Models often changed their predictions when the referring expression changed, despite the target remaining fixed. Increasing visual complexity further worsened this sensitivity. We additionally introduced a GUI Grounding Diagnosis Agent that uses model feedback to iteratively generate instructions that expose model-specific failure cases, demonstrating how diagnostic evaluation can move from passive measurement toward active failure discovery.

The results of this thesis indicate that reliable computer-use agents will not emerge from optimizing a single benchmark score alone. They require models that understand how interfaces change over time, ground language consistently to interface elements, recover when their expectations are violated, and expose uncertainty when they cannot act safely. Fine-grained diagnosis provides a practical path toward these goals by revealing failures that aggregate metrics conceal and by turning those failures into concrete targets for model improvement. As computer-use agents become more capable and are trusted with increasingly consequential tasks, evaluation must evolve from asking only whether an agent succeeded to asking what it understood, why it failed, and whether it can be trusted under variation.

Chapter 5

Future Work

5.1 Closing the Loop Between Diagnosis and Training

A natural next step is to use the failures identified by diagnostic evaluations as direct training signals. The planning evaluations in this thesis could support curricula that explicitly teach temporal ordering, action-conditioned state transitions, and recovery from erroneous states. Similarly, grounding failures discovered across diverse referring expressions could be converted into hard negative examples or consistency objectives that encourage a model to map semantically equivalent instructions to the same UI element. The GUI Grounding Diagnosis Agent provides an initial mechanism for automatically finding such examples, but future systems could form a continuous loop in which a diagnostic agent generates challenging cases, a target model is updated on those cases, and the evaluator verifies whether the improvement generalizes beyond the generated examples. Reinforcement-learning approaches, including group-relative optimization objectives, may also be useful for training instruction generators to discover valid, diverse, and increasingly informative failure cases.

5.2 Unified Diagnosis Across the Agent Pipeline

This thesis studied planning and grounding as separate capabilities in order to isolate their failure modes. In an end-to-end agent, however, these components interact closely. A grounding error can corrupt the state used for subsequent planning, while a planning error can cause an otherwise correct grounding prediction to be applied to the wrong action. Future work should therefore develop unified diagnostic frameworks

that trace how errors propagate across perception, grounding, memory, planning, and execution. Controlled interventions could replace one component with ground-truth information while leaving the rest of the agent unchanged, making it possible to estimate the causal contribution of each capability to task success. Such evaluations would provide a more precise bridge between unit-level benchmarks and end-to-end performance.

5.3 Interactive, Long-Horizon, and Recovery-Centered Evaluation

The tests developed in this thesis intentionally isolate specific capabilities, often over relatively short horizons or controlled screenshots. Future evaluations should extend these ideas to interactive environments in which agents must maintain state over longer trajectories, handle delayed consequences, and adapt when the interface changes unexpectedly. In particular, recovery should be treated as a first-class capability. Real agents will inevitably misclick, encounter pop-ups, lose context, or reach states that differ from the expected trajectory. Benchmarks should therefore measure not only whether an agent avoids errors, but also whether it detects them, revises its internal state, and returns to a valid plan. This direction is especially important for tasks containing irreversible or costly actions, where an agent must reason about risk before acting.

5.4 State Representation and Predictive Models of Interfaces

The planning results suggest that current VLMs often lack a stable representation of how interfaces evolve under actions. One promising direction is to equip agents with explicit state representations or learned world models of graphical environments. Rather than reasoning only from the current screenshot and a textual action history, an agent could maintain a structured belief over UI elements, their attributes, and the effects of previous actions. Training objectives based on action-conditioned future-state prediction, state consistency, and counterfactual rollouts may improve both planning and error recovery. These representations could also connect planning and grounding by associating linguistic references with persistent interface entities rather than isolated pixel coordinates.

5.5 Broader Generalization and Human Variation

The grounding study focuses on desktop environments, but substantial variation remains across operating systems, applications, languages, accessibility settings, screen resolutions, and user populations. Future benchmarks should include multilingual instructions, assistive technologies, personalized terminology, and interfaces with uncommon layouts or visual themes. The same principle applies to planning: agents should be tested across websites and applications whose structure, interaction conventions, and state transitions differ from those seen during training. Evaluating such distribution shifts would help determine whether a model has learned general principles of computer interaction or merely adapted to recurring interface patterns.

5.6 Calibration, Verification, and Safe Delegation

Finally, reliable agents should recognize when their grounding or planning is uncertain. Future work should pair diagnostic accuracy with measures of calibration, abstention, and selective prediction. An agent that can identify ambiguous references, request clarification, verify the effect of an action, or defer a high-risk decision may be substantially safer than one that always acts with high confidence. Diagnostic evaluations could be used to define capability-specific confidence thresholds and to determine when human oversight is necessary. This would shift the goal from building agents that always produce an action to building agents that act only when they have sufficient evidence that the action is correct.

Appendix A

Additional material for Chapter 1

A.1 Creating Synthetic Data with MiniWob++

In this section, we describe more details and examples for the data we create for Section 2.4.2, Section 2.5.2 using the MiniWob++ environment.

Edge Case Test: As shown in Figure A.2, given a task (e.g "Check abs checkbox"), we first obtain a clean or fresh form which has not been interacted with. Next, we algorithmically perform a series of actions, where we randomly interact with a subset of form elements (e.g clicking one/two checkboxes). to generate our edge-case screenshots. This process results in an edge case screenshot where some incorrect elements may be clicked, or certain fields may have different pre-filled values, and VLMs have to figure out their way to navigate (or recover) through this altered state and solve the task. We perform the above across three MiniWob++ environments, each featuring a form with a distinct combination of elements - (1) checkboxes only, (2) a slider and a checkbox, (3) a textbox and a radiobutton (See Figure A.1). Finally, these environments correspond to three specific tasks for our Edge Case test, namely Task 1, Task 2, and Task 3, respectively (See Figure A.3).

Future State Consistency: Now, we leverage the form screenshots obtained above and algorithmically create three tasks (Task 1, Task 2, Task 3) to examine if VLMs can determine whether two sequences of action lead to the same future states (see Figure A.4 for long-horizon action plan scenarios and see Figure A.5 for shorter ones).

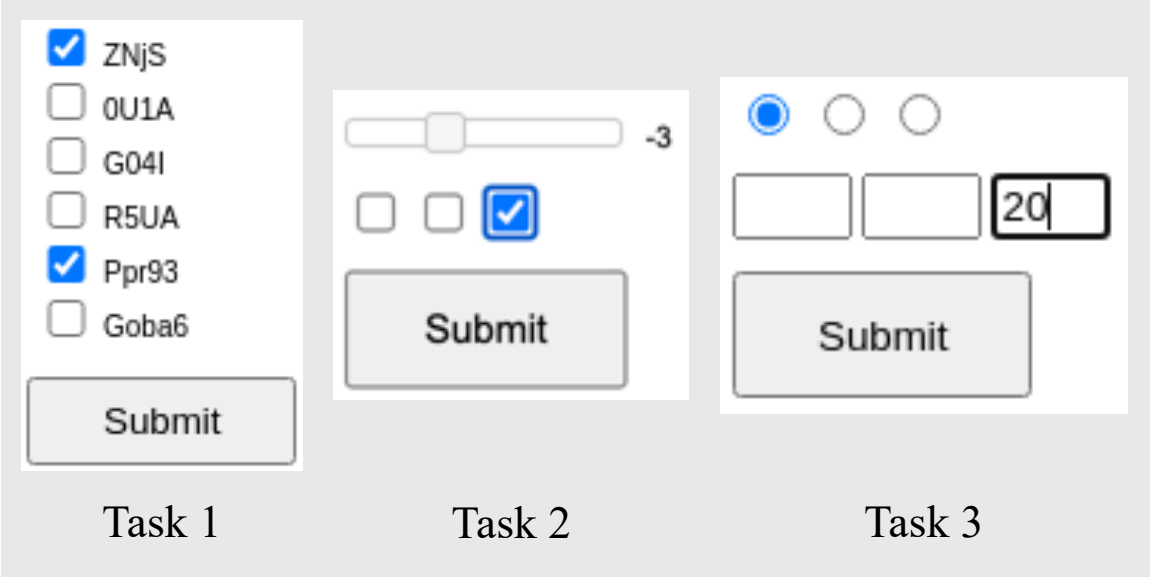


Figure A.1: **Example Synthetic Data from MiniWob++.** We leverage the MiniWob++ environment to generate three groups of tasks: 1). a list of checkboxes, 2). a slide with three checkboxes, and 3). three radio-buttons with three text boxes. All tasks contain a "Submit" button.

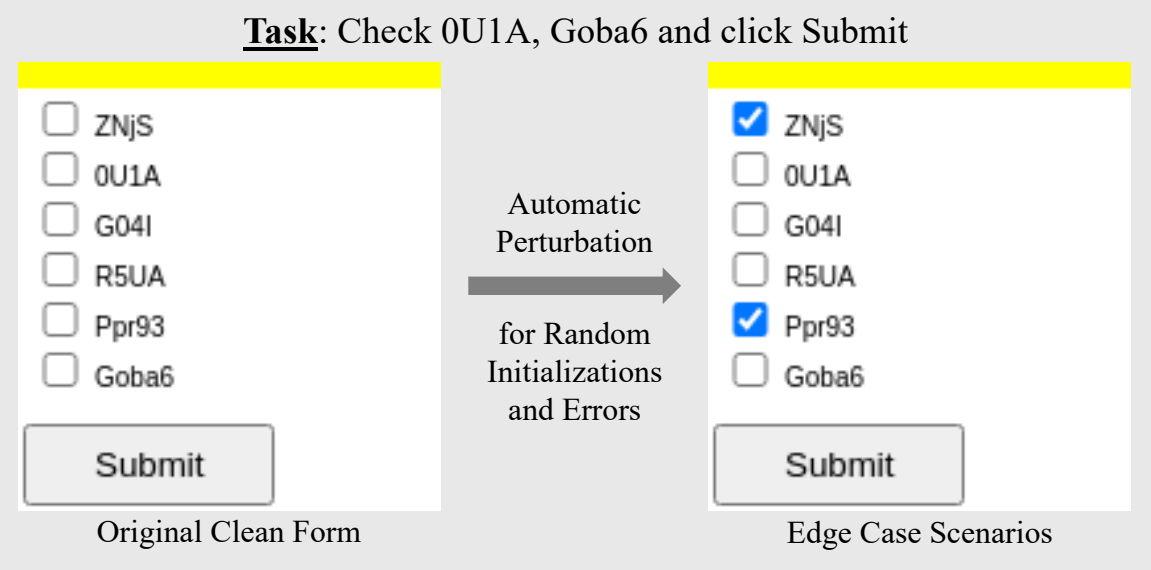
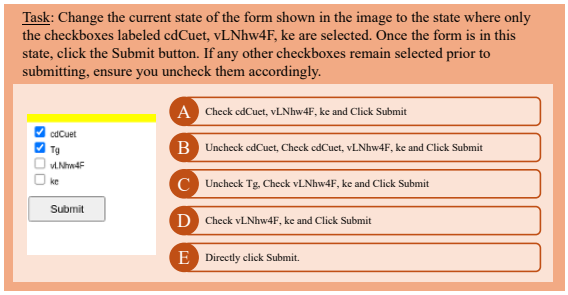
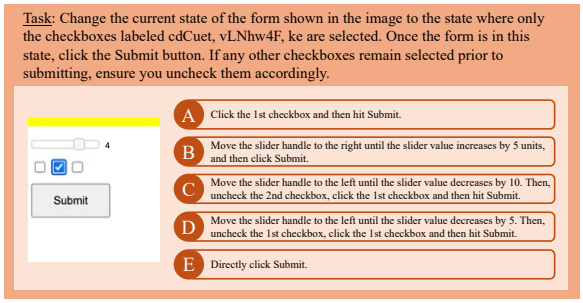


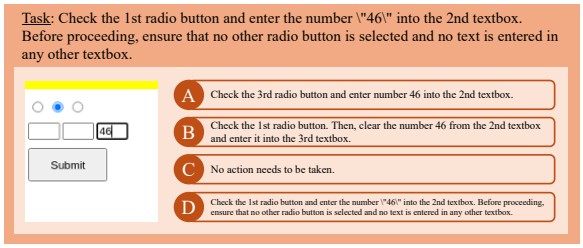
Figure A.2: Illustrating one of our algorithmic interactions to perturb the original clean form state.



(a) Task 1



(b) Task 2



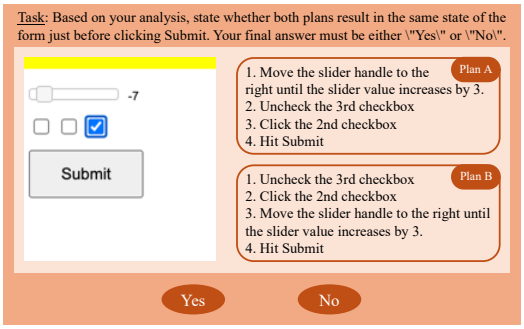
(c) Task 3

Figure A.3: Task examples for Edge Case Test Study

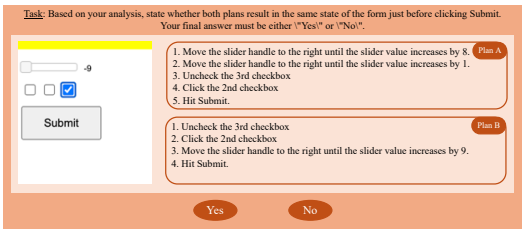
A.2 Choice of States in the Temporal Ordering Setup

As discussed earlier, our design in the temporal ordering experiment focuses on transitions between the initial state i_0 and the final state i_t to reduce ambiguity. We now present a discussion for our choice along with the ablation experiments after selecting random states i_j, i_k.

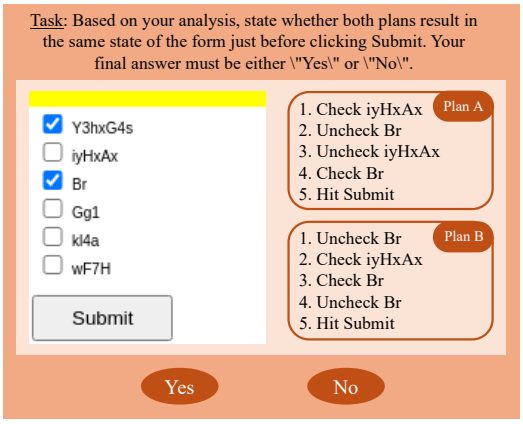
Discussion. Our original choice addresses challenges of noise associated with the random sampling of intermediate webpage states. For example, in tasks like *"Buy Louis Vuitton on Amazon,"* users scroll through search results dynamically, producing intermediate screenshots that reflect varying, non-sequential segments of the page. These variations, which do not follow a consistent semantic order, complicate efforts



(a) Task 1

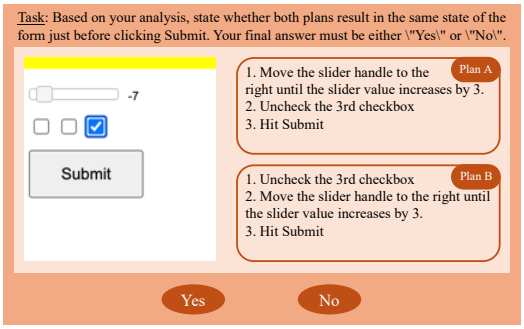


(b) Task 2

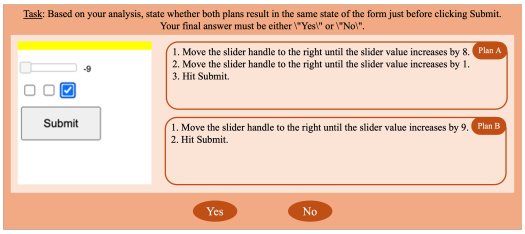


(c) Task 3

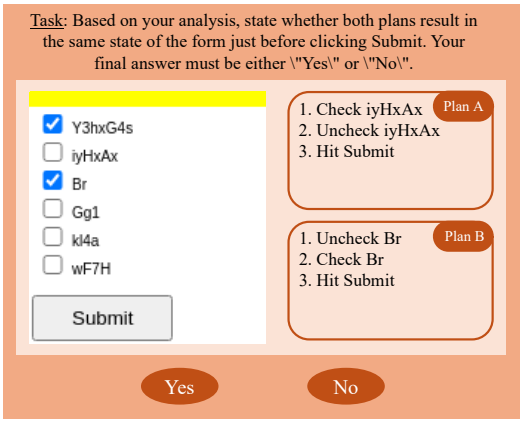
Figure A.4: Task examples for Future State Prediction study – Long Horizon i.e plans have (3-4 action steps before submitting the form)



(a) Task 1



(b) Task 2



(c) Task 3

Figure A.5: Task examples for Future State Prediction study – Short Horizon i.e plans have (1-2 action steps before submitting the form)

to determine which state occurs earlier. Similarly, in tasks such as *"Add toilet paper and laundry detergent to Amazon cart,"* the order of intermediate screenshots can plausibly vary depending on user behavior, making definitive temporal labels infeasible. Thus, creating a reliable evaluation data in the random way necessitates the need for manually inspecting and filtering the ambiguous intermediate states and the entire generated data, which is out of scope for our work.

Ablation Results. For the sake of completeness, we now include evaluation results with random states i_j, i_k. From Table A.1, we can observe that for both the datasets MM-M2W, GUIAct, the performance of chosen VLMs (MiniCPM, Qwen2-VL-7B) is poor and not very substantiallyy better than the random guess performance of 50%. Although the potential for the aforementioned issues makes it harder to draw trends from these numbers, we can still observe the VLMs to struggle similarly for the intermediate states.

Model	**MM-M2W**		**GUIAct**	
	Instruct	CoT	Instruct	CoT
Minicpm	46.28	48.26	51.65%	59.35%
Qwen2-VL 7b	44.94	44.72	54.28%	54.64%

Table A.1: Comparison of performance on MM-M2W and GUIAct datasets using random intermediate states.

A.3 Correlation Between Planning Skill Performance and Task Success Rate

To investigate the relationship between planning skill performance and overall task success, we analyze the results of three models—Qwen2-VL-7B, LLaVA-1.6, and Phi-3.5—on the Mind2Web-Live benchmark [48]. We evaluate these models across three core planning skills: *Future State Prediction*, *Plan Selection (Real World)*, and *Edge Case Detection*, along with their task success metrics, including average step success rate, completion rate, and full task success rate. The results are summarized in Table A.2.

As shown in Table A.2, the average planning skill scores for LLaVA-1.6, Phi-3.5, and Qwen2-VL-7B are 41.72, 44.75, and 60.31, respectively. These scores correlate with each model's task success rates: LLaVA-1.6 achieves a full task success rate of 4.8%, while Phi-3.5 and Qwen2-VL-7B achieve 13.3% and 19.3%, respectively.

Model	Future State Prediction	Plan Selection (Real World)	Edge Case Detection	Planning Avg	Avg Step SR	Completion Rate	Full Task SR
LLaVA-1.6	35.70	54.67	34.79	41.72	32.0	30.3	4.8
Phi-3.5	56.68	48.04	29.52	44.75	44.0	34.0	13.3
Qwen2-VL-7B	64.71	67.53	48.70	60.31	45.3	40.2	19.3

Table A.2: Comparison of planning skill evaluation and task success metrics across models on the Mind2Web-Live benchmark.

The ranking of models by average planning ability closely mirrors their ranking in task-level performance metrics, suggesting a strong relationship between planning skill and downstream task effectiveness. This finding reinforces the value of skill-level evaluations in understanding and predicting model performance in real-world web environments.

A.4 Comparison with Proprietary Models

Given resource constraints, we were unable to run the complete set of experiments on large-scale proprietary models. To demonstrate the value of our experimental setup, we conducted smaller-scale evaluations using **GPT-4o-mini** across two representative tasks: *Temporal Ordering* and *Plan Selection in Real Web Environments*.

For the Temporal Ordering task, we select the GUIAct data and sample a subset of 500 questions from the full evaluation set, whereas for Plan Assessment, we evaluate on the Shuffle Perturbation setting. The results are shown in Table A.3.

Phase	Instruct	CoT
Order1	40.6	41.6
Order2	57.8	59.6

Table A.3: Temporal Ordering task performance (%) on GPT-4o-mini.

Prompt Type	Accuracy (%)
Instruct	49.66
CoT	48.97

Table A.4: Plan Selection task performance (%) on GPT-4o-mini under the Shuffle Perturbation setting.

Our results with GPT-4o-mini reveal two key findings: (1) temporal prediction accuracy is only marginally better than random guessing (50%), and (2) performance

on the plan selection task is weaker than that of many open-source models (see Table 2.3). These observations suggest that proprietary models do not necessarily outperform smaller open models on fine-grained web planning evaluations, highlighting the value of our task-specific benchmarking approach.

A.5 Prompts

In this section, we include the following prompts: **Temporal Ordering:** For simplicity, we only include the prompts for Non-MCQ settings. Note that we use the same prompt for the MCQ setting, differing only in the output format. Figure A.6 shows the Instruction prompt and Figure A.7 shows the CoT prompt.
Future State Prediction: Figure A.8 shows the Instruction prompt and Figure A.9 shows the CoT prompt.
Plan Assessment : Real Web Environment Figure A.10 shows the Instruction prompt and Figure A.11 shows the CoT prompt. We add action history to the prompt in Figure A.12 to evaluate the impact of the presence of history in Section A.7.
Plan Assessment : Edge Case Scenario Figure A.13 shows the Instruction prompt and Figure A.14 shows the CoT prompt.
Llama as a Judge Prompt Figure A.16, Figure A.15, Figure A.17 includes all the LLama prompts we use throughout our paper.

A.6 Human Study Details

In this section, we describe the process we followed to obtain human performance for the tests mentioned in this paper.
Number of samples. We randomly sample 15 questions for Temporal Ordering, Plan Selection (Real Web Scenarios), and Plan Selection (Edge Cases). For Future State Prediction, which has Long and Short Horizon configurations, we sample 20 total questions to balance volunteer effort and question count. As noted in the main paper, of the two configurations for Plan Selection (Real Web)—Shuffling Perturbation and Semantic Perturbation—we conduct human study only for the harder Shuffling Perturbation due to constraints.
Instructions given to raters. The tests are presented to humans in the same format as the VLMs in our study, i.e., MCQs with 4/5 options. To ensure a fair comparison, we present the same instructions to both VLMs and human raters. Accordingly, we use the prompts outlined in Figure A.13, Figure A.8, Figure A.10, and Figure A.6.

Temporal Ordering (No-MCQ) (Instruction Prompting)

You are given a task along with two image screenshots representing steps in a trajectory that either solves the given question or helps find the relevant information required to answer it. Your goal is to determine which screenshot likely represents an earlier step in the trajectory.

An **earlier step in the trajectory** refers to a step that must logically or procedurally occur before the other. Consider dependencies, prerequisites, or actions that logically precede others. For example, if one step involves entering information and another involves confirming it, entering the information must logically come earlier. Use this principle to analyze the order of the screenshots.

Use the task to understand the context, and carefully examine each screenshot for any clues that suggest their order in the sequence. Consider all visible text, UI elements, and page structure to infer which screenshot likely comes earlier in the process.

Important Notes:
1. **Picture Labelling**: Refer to the input image screenshots as [Picture 1, Picture 2]. These labels are purely for identification and do not indicate any logical order or sequence in the trajectory. Your objective is to determine which picture represents the earlier step in the trajectory based on the given question description and careful analysis of the pictures.

2. **Relevance to the Task**: Both screenshots are directly related to either solving the question or finding relevant information to answer it. Do not assume that either image is irrelevant.

3. **Expert Trajectories**: These screenshots are taken from expert human trajectories that were executed to successfully solve the question. Trust that the screenshots reflect valid steps.

4. **Arbitrary Order**: The screenshots are presented in no particular order. Their arrangement does not indicate which step comes earlier in the trajectory.

Task:
{task}

Question:
Based on the two given images and the question, determine which image (or screenshot) represents an earlier step in the trajectory.

Output format:
Clearly state which picture represents the **earlier step** in the trajectory, "Picture 1" or "Picture 2".

Only output your answer choice ("Picture 1" or "Picture 2"). Do not include any reasoning, explanation, or additional text.

Figure A.6: Prompt for Temporal Ordering (No-MCQ)

Temporal Ordering (No-MCQ) (Chain-of-Thought Prompting)

You are given a task along with two image screenshots representing steps in a trajectory that either solves the given question or helps find the relevant information required to answer it. Your goal is to determine which screenshot likely represents an earlier step in the trajectory.

An **earlier step in the trajectory** refers to a step that must logically or procedurally occur before the other. Consider dependencies, prerequisites, or actions that logically precede others. For example, if one step involves entering information and another involves confirming it, entering the information must logically come earlier. Use this principle to analyze the order of the screenshots.

Use the task to understand the context, and carefully examine each screenshot for any clues that suggest their order in the sequence. Consider all visible text, UI elements, and page structure to infer which screenshot likely comes earlier in the process.

Important Notes:
1. **Picture Labelling**: Refer to the input image screenshots as [Picture 1, Picture 2]. These labels are purely for identification and do not indicate any logical order or sequence in the trajectory. Your objective is to determine which picture represents the earlier step in the trajectory based on the given question description and careful analysis of the pictures.

2. **Relevance to the Task**: Both screenshots are directly related to either solving the question or finding relevant information to answer it. Do not assume that either image is irrelevant.

3. **Expert Trajectories**: These screenshots are taken from expert human trajectories that were executed to successfully solve the question. Trust that the screenshots reflect valid steps.

4. **Arbitrary Order**: The screenshots are presented in no particular order. Their arrangement does not indicate which step comes earlier in the trajectory.

Task:
{task}

Question:
Based on the two given images and the task, determine which image (or screenshot) represents an earlier step in the trajectory.

Output format:

REASONING: Provide a concise step-by-step reasoning using the steps outlined above.

OUTPUT: Output your answer choice ("Picture 1" or "Picture 2"). Do not include any additional text here.

Figure A.7: Chain-of-Thought Prompt for Temporal Ordering (No-MCQ)

Future State Prediction (Instruction Prompting)

You are given an image showing the initial state of a form. This form requires the user
to perform a series of actions before clicking the Submit button. Submitting the form
navigates the user to a new window or page.

You are provided with two action plans (Plan A and Plan B) that describe the sequence of
actions to interact with the form. Each plan is executed independently and in isolation,
starting from the exact same initial state shown in the image. The two plans are not
executed one after the other, and neither plan influences the result of the other.

Your task is to determine if the state of the form just before clicking Submit will be
the same after following either plan.

Action Plans:
{plan_choices}

Question:
Based on the given image and the provided action plans, do both plans lead to the same
state of the form just before clicking Submit when executed independently and in isolation,
each starting from the same initial state of the form (i.e., shown in the image)?

Output format:
Only output "Yes" or "No".

Do not include any reasoning, explanation, or additional text.

Figure A.8: Prompt for Future State Prediction

Future State Prediction (Chain-of-Thought Prompting)

You are given an image showing the initial state of a form. This form requires the user
to perform a series of actions before clicking the Submit button. Submitting the form
navigates the user to a new window or page.

You are provided with two action plans (Plan A and Plan B) that describe the sequence of
actions to interact with the form. Each plan is executed independently and in isolation,
starting from the exact same initial state shown in the image. The two plans are not
executed one after the other, and neither plan influences the result of the other.

Your task is to determine if the state of the form just before clicking Submit will be
the same after following either plan.

Action Plans:
{plan_choices}

Question:
Based on the given image and the provided action plans, do both plans lead to the same
state of the form just before clicking Submit when executed independently and in isolation,
each starting from the same initial state of the form (i.e., shown in the image)?

Follow the steps provided next to arrive at your answer:

1. **Image Analysis**: Describe the relevant elements in the image, and carefully review the initial state of the form.
2. **Plan Evaluation**: Separately evaluate both the action plans step-by-step, and carefully determine how each step changes the state of the form.
3. **State Comparison**: Compare the final states of the form just before clicking Submit after executing Plan A and Plan B in isolation, each starting from the same initial state. Look for differences in the values and configurations of the form elements.
4. **Final Answer**: Based on your analysis, state whether both plans result in the same state of the form just before clicking Submit. Your final answer must be either "Yes" or "No".

Output format:
REASONING: Be concise and to the point.

OUTPUT: "Yes" or "No". Do not include any additional text here.

Real Web Environment Study (Instruction Prompting)

You are given an image and a task that needs to be performed based on that image. The image represents what a user is currently seeing on their screen while attempting to solve the task. You are also provided with four choices of action plans (each represented as a set of actions) that outline potential steps to accomplish the task.

Your goal is to carefully analyze the image and evaluate the four action plans to decide which one most effectively helps the user complete the task.

Important Notes:
1. The current screenshot represents the user's current state or screen. As the user executes an action (e.g., clicking a button) from any plan, the screen or state may change. You must imagine these transformations and assess the feasibility of the remaining actions in each plan.

2. Some action plans may include invalid actions that cannot be executed based on the current or subsequent screens. It is your responsibility to identify and exclude such plans from being valid answer candidates.

3. The task is guaranteed to succeed within the next {plan_len} steps. If an action plan cannot complete the task within its steps, it is not a valid answer candidate.

4. Assume all plans are related to the task. However, some plans may include errors, inefficiencies, or invalid actions. Your responsibility is to evaluate each plan thoroughly to determine its effectiveness.

Task:
{task}

Plans (each represented as a series of actions):
{plan_text}

Question:
Based on the given image (the webpage screenshot), and the task,
which plan (A, B, C, D) would most effectively accomplish the task?

Output format:
Only output your choice of A, B, C, or D.

Do not include any reasoning, explanation, or additional text. Only provide your choice of plan (either A, B, C, D) in the response.

Figure A.10: Prompt for Real Web Environment Study

Real Web Environment Study (Chain-of-Thought Prompting)

You are given an image and a task that needs to be performed based on that image. The image represents what a user is currently seeing on their screen while attempting to solve the task. You are also provided with four choices of action plans (each represented as a set of actions) that outline potential steps to accomplish the task.

Your goal is to carefully analyze the image and evaluate the four action plans to decide which one most effectively helps the user complete the task.

Important Notes:
1. The current screenshot represents the user's current state or screen. As the user executes an action (e.g., clicking a button) from any plan, the screen or state may change. You must imagine these transformations and assess the feasibility of the remaining actions in each plan.

2. Some action plans may include invalid actions that cannot be executed based on the current or subsequent screens. It is your responsibility to identify and exclude such plans from being valid answer candidates.

3. The task is guaranteed to succeed within the next {plan_len} steps. If an action plan cannot complete the task within its steps, it is not a valid answer candidate.

4. Assume all plans are related to the task. However, some plans may include errors, inefficiencies, or invalid actions. It is your responsibility to evaluate them thoroughly.

Task:
{task}

Plans (each represented as a series of actions):
{plan_text}

Question:
Based on the given image (the webpage screenshot), and the task, which plan (A, B, C, D) would most effectively accomplish the task?

Output format:
Follow the format below for your response.

REASONING: Provide a concise step-by-step reasoning using the steps outlined above.

OUTPUT: Output your answer choice ('A', 'B', 'C', or 'D').

Figure A.11: Chain-of-Thought Prompt for Real Web Environment Study

Real Web Environment Study (Instruction Prompting) with Action History / Previous Action Input

You are given an image and a task that needs to be performed based on that image. The image represents what a user is currently seeing on their screen while attempting to solve the task. Along with this, you are given the exact set of previous actions the user has taken to reach the current state depicted in the image.

In addition, you are provided with four choices of action plans (each represented as a set of actions) that outline potential steps to accomplish the task. Your goal is to carefully analyze the image, review the previous actions, and evaluate the four action plans to determine which one most effectively helps the user complete the task.

Important Notes:
1. The current screenshot represents the user's current state or screen. As the user executes an action (e.g., clicking a button) from any plan, the screen or state may change. You must imagine these transformations and assess the feasibility of the remaining actions in each plan.

2. Some action plans may include invalid actions that cannot be executed based on the current or subsequent screens. It is your responsibility to identify and exclude such plans from being valid answer candidates.

3. The task is guaranteed to succeed within the next {plan_len} steps. If an action plan cannot complete the task within its steps, it is not a valid answer candidate.

4. Assume all plans are related to the task. However, some plans may include errors, inefficiencies, or invalid actions. Your responsibility is to evaluate each plan thoroughly to determine its effectiveness.

Task:
{task}

Previous actions:
{previous_actions_text}

Plans (each represented as a series of actions):
{plan_text}

Question:
Based on the given image (the webpage screenshot), the previous actions, and the task, which plan (A, B, C, D) would most effectively accomplish the task?

Output format:
Only output your choice of A, B, C, or D.

Edge Case Test Study (Instruction Prompting)

You are given an image and a task that needs to be performed based on that image. Additionally, you are given several possible action plans, each describing potential next steps to achieve the task. Your goal is to select the most effective plan from the options provided. Carefully analyze the image, and then evaluate each option to decide which one best completes the task.

Task:
{task}

Possible Action Plans:
{plan_text}

Question:
Based on the given image, and the given task, which action plan ({option_text}) would most effectively accomplish the task?

Output format:
Only output your choice of {option_text}.

Do not include any reasoning, explanation, or additional text.

Figure A.13: Chain-of-Thought Prompt for Real Web Environment Study with Action History / Previous Action Input

Edge Case Test Study (Chain-of-Thought Prompting)

You are given an image and a task that needs to be performed based on that image. Additionally, you are given several possible action plans, each describing potential next steps to achieve the task. Your goal is to select the most effective plan from the options provided. Carefully analyze the image, and think step by step to evaluate each option to decide which one best completes the task.

Task:
{task}

Possible Action Plans:
{plan_text}

Question:
Based on the given image, and the given task, please follow these steps:

1. **Image and Task Analysis:** Describe relevant elements in the image and clarify the requirements of the task.
2. **Option Evaluation:** Go through each action plan ({option_text}) one by one, and describe why it would or would not help achieve the task. Mention any advantages or limitations of each option based on the image.
3. **Final Choice:** Based on your analysis, select the option ({option_text}) that best accomplishes the task.

Output Instructions:
Your output should follow the below format:

REASONING: Be concise and to the point.

OUTPUT CHOICE: An uppercase letter

Figure A.14: Chain-of-Thought Prompt for Edge Case Test Study

Gathering Responses. Our raters included individuals from diverse backgrounds, including students across various degrees (Bachelor's, Master's, PhD) and fields (CS, ECE, Mathematics, Robotics, Nursing, Pharmacy, etc.), as well as Postdoctoral Researchers, Research Associates, and Technical Industry Professionals. Responses were collected via Google Forms. Depending on their availability, each rater participated in 1–4 studies, with 2 being the most common. This ensured a minimum of 15 responses per study.

Results. The performances of human raters are **96.2%** for Temporal Ordering, **97%**

Llama as a judge for Temporal Ordering

You are acting as a judge. The statement below describes in text which image comes
earlier in a sequence. Parse it carefully and decide which picture comes earlier.
Respond strictly with either **"Picture 1"** or **"Picture 2"** and no additional text:

Statement:
{output}

Figure A.15: Prompt for Llama as a judge for Temporal Ordering

Llama as a judge for Future State Prediction

You are provided with a response to the following question:

"Do both plans (Plan A and Plan B) result in the same final state of the form (i.e., just before clicking Submit)?"

Your task is to analyze the response and determine the answer based on the following criteria:

1. Output "yes" if the response explicitly states that both Plan A and Plan B achieve the same final state.
2. Output "no" if the response explicitly states that Plan A and Plan B do not achieve the same final state.
3. Output "unclear" if the response does not clearly state whether Plan A and Plan B achieve the same final state.

Respond with only the answer: "yes", "no", or "unclear". Provide no additional text.

****Response:****
{response}

****Answer:****

Figure A.16: Prompt for Llama as a judge for Future State Prediction

Llama as a judge for Plan Assessment (both Real-World and Edge Case)

You are acting as a judge. The statement below describes in text which Action Plan Option best accomplishes the task. Parse it carefully.

Your task is to analyze the statement and determine the answer based on the following criteria:

1. Output "A" if the statement states "Plan A" or "Option A" or anything related.
2. Output "B" if the statement states "Plan B" or "Option B" or anything related.
3. Output "C" if the statement states "Plan C" or "Option C" or anything related.
4. Output "D" if the statement states "Plan D" or "Option D" or anything related.
5. Output "unclear" if the statement does not state which Action or Option best accomplishes the task.

Respond with only the answer: "A", "B", "C", "D", or "unclear". Provide no additional text.

Statement: {response}

****Answer:****

Figure A.17: Prompt for Llama as a judge for Plan Assessment

for Future State Prediction, **90%** for Plan Selection – Real Web Scenarios, and **99.2%** for Plan Selection – Edge Case Test. Qualitative feedback from raters revealed that most errors occurred due to rater fatigue of questions, rushing, or missing thorough reading of the task instruction and options.

A.7 Impact of Action History in Future Plan Assessment

As described in Section 2.5.1, we use the MM-M2W dataset [72], where each data point includes a task instruction and a human-annotated trajectory (a sequence of states/screenshots and actions at each state) to complete the task. We first select samples with action trajectories of length ≤ 10. To create a test question for each task instruction g, we sample a web page screenshot (or state) i_t from its solution trajectory, where i_t is 4-5 action steps (a_p) away from the task's end state (or goal completion). Thus, it can be understood that for sequences with lengths ≥ 5, there will be i_t steps leading from the start state (i_0, typically the homepage) to i_t. We refer to these as "previous actions." or "action history". In this ablation experiment, we investigate whether VLM performance in plan assessment improves when provided with "previous actions" as input, specifically testing if VLMs can better assess plans when given the history of actions. To construct prompts for this study, we adapt those used in Section 2.5.1, e.g, Figure A.12 is derived from Figure A.10.

Our results in Table A.5 reveal a trend similar to the main paper table without action history. Notably, only 2/19 models achieve an average performance of $\geq$50% under the Shuffling Perturbation setting, while models perform better in the Semantic Perturbation setting. In summary, we conclude that augmenting action history offers no significant benefit in improving VLMs' plan assessment capabilities.

A.8 Additional Results and Details

In this section, we visualize outputs and include tables that we could not include in the main paper due to space constraints. Specifically, we present more results for: (1) VLMs' spurious preferences in Temporal Ordering task (Figure A.18), (2) Unreasonable quality of GUI-specific models (Figure A.19), (3) Samples with erroneous responses for Future State Prediction (Figure A.21) and Real-Web Plan Assessment Analysis (Figure A.20), (4) Comprehensive tables detailing results for each individual

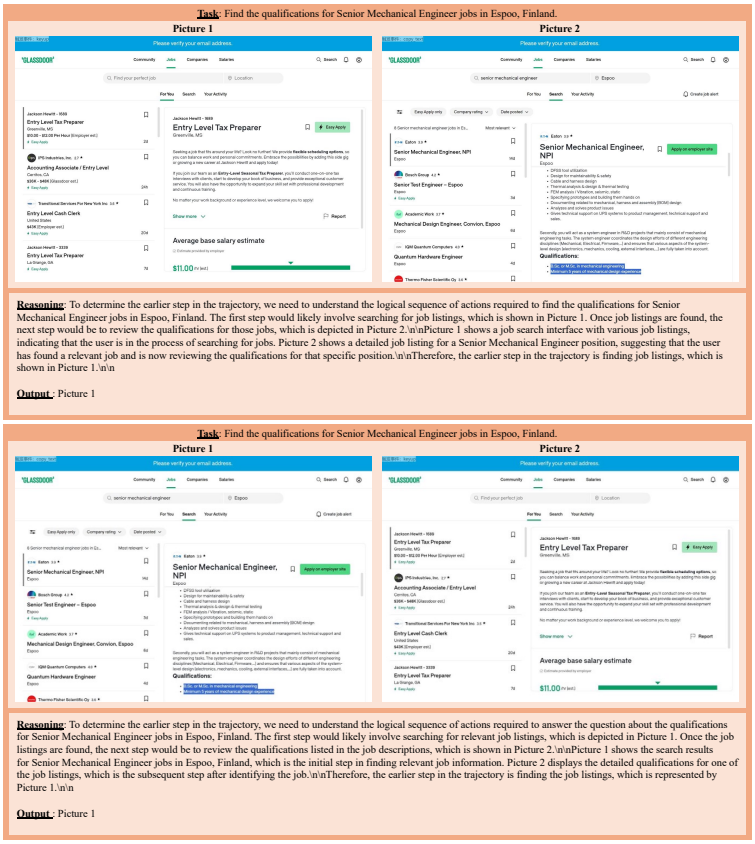
Main	Shuffle Perturb.			Semantic Perturb.		
	INS	COT	**Avg**	INS	COT	**Avg**
Qwen2-VL 2B	41.88	23.01	32.45	52.45	28.67	40.56
Qwen2-VL 7B	56.22	48.3	52.26	87.92	71.69	79.81
Qwen2-VL 72B	79.24	69.05	74.14	95.09	90.94	93.02
Minicpm	55.47	36.22	45.84	83.01	63.01	73.01
Minicpm-o	49.05	39.24	44.14	73.0	70.56	71.78
Phi 3.5	27.92	33.2	30.56	56.6	55.09	55.84
IDEFICS3	43.39	33.58	38.48	80.75	55.47	68.11
DeepSeek VL	28.67	20.0	24.34	38.87	25.66	32.27
Llava OV 0.5B	24.52	8.3	16.41	27.02	8.3	17.66
Llava OV 7B	56.6	41.5	49.05	89.05	75.09	82.07
InternVL2-MPO	46.41	30.56	38.48	90.56	71.32	80.94
Glm	28.3	39.62	33.96	64.52	59.62	62.07
Llava 1.6 - 7B	26.41	30.18	28.30	38.86	46.59	42.73
Llava 1.6 - 13B	30.94	29.43	30.19	52.45	48.3	50.38
Llava 1.6 - 34B	41.50	41.50	41.50	87.16	70.45	78.81
Ferret-UI-Llama	23.1	26.03	24.57	26.03	23.77	24.90
Ferret-UI-Gemma	12.07	21.13	16.60	9.88	21.21	15.55
UIX-Llava	48.3	33.2	40.75	67.54	41.88	54.71
CogAgent	23.86	22.34	23.10	23.77	17.73	20.75

Table A.5: Evaluations of 19 VLMs on Plan Selection in Real-Web Environment. In this particular experiment, we augment VLM input with the history of its previous actions. While we anticipated an improvement in performance after using information of previous actions, we found that VLMs reaped no substantial benefit from the aforementioned.

task—Task 1, Task 2, and Task 3 of the Edge Case Test (see Table A.9), as well as Task 1, Task 2, and Task 3 of the Future State Consistency test (see Table A.7, Table A.8), (5) Dataset Statistics summarizing number of test examples for every configuration of our paper (see Table A.6).

A.9 Would VLMs perform perfectly if Perception error did not exist?

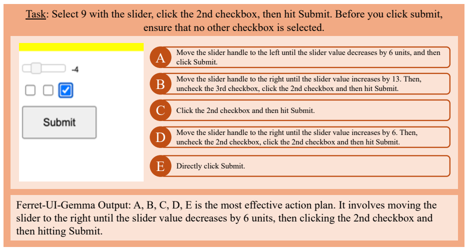
In the error case study for the edge case test (Figure 2.4), we observe that "Perception Error," where VLMs misinterpret certain visual details in the image, emerges as a bottleneck for plan identification. While addressing this issue is beyond the scope of our work, we explore how VLM performance changes when provided with accurate ground truth captions describing the form. Specifically, we aim to



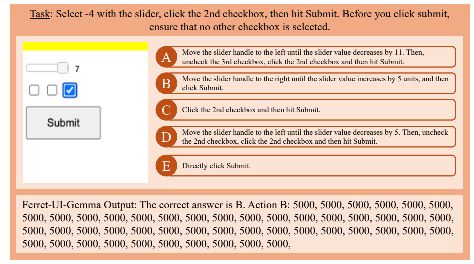
(a) Example 1

(b) Example 2

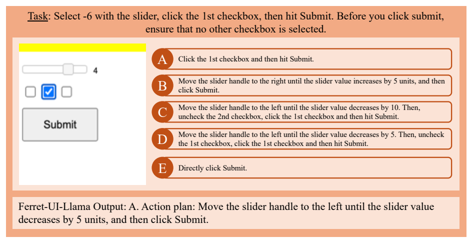
Figure A.18: In the main paper, we find that VLMs are biased towards answering 'Picture 1' for the temporal ordering task. Here, we show qualitative results how VLMs tend to rationalize their answer of "Picture 1" with untenable reasons



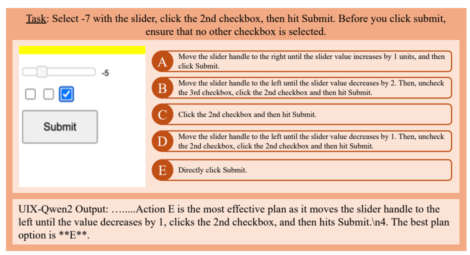
(a) GUI Model Outputs – Irrelevant/Gibberish



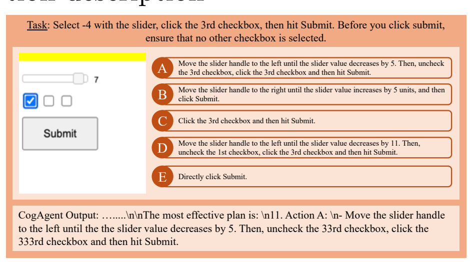
(b) GUI Model Outputs – Irrelevant/Gibberish



(c) GUI Model Outputs – Wrong option description



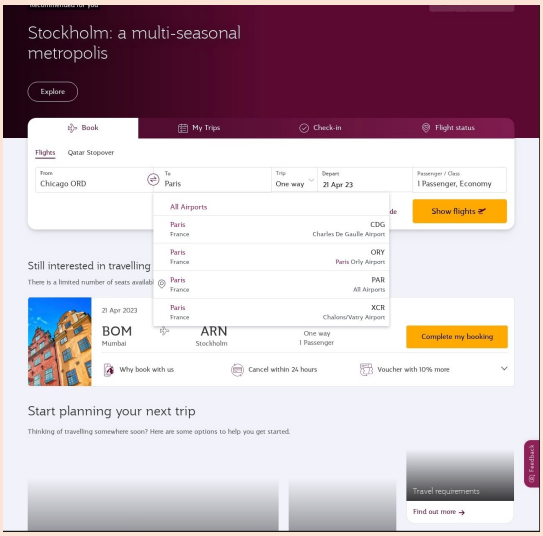
(d) GUI Model Outputs – Wrong option description



(e) GUI Model Outputs – Wrong option description

Figure A.19: In the main paper, we discuss how GUI models have worse performances than general VLMs. We now demonstrate qualitative outputs eliciting deterioration in GUI models' general reasoning and instruction following capabilities. More specifically, fig (a), (b) represent cases where model starts producing gibberish content and does not choose an option from the given choices. Fig (c), (d), (e) has examples where model did choose an option but gave an incorrect description of what the option actually is in the question.

Task:Show me the list of one-way flights today (April 17) from Chicago to Paris



Plan A

1. Click on the Show flights button to view the available flight options.",
2. Click on the Continue button to proceed with the booking process.",
3. Select 17 April 2023 from the available grid cell for the departure date.",
4. Click on the Paris France CDG Charles De Gaulle Airport menu item to select the departure location.",
5. Choose the One way option for the type of trip."

Plan B

1. Click on the Add to Cart link to include the item in your shopping cart.",
2. Click on the Buy a copy link to proceed with purchasing options.",
3. Click on the Gorillaz link to explore more about Gorillaz.",
4. Click on the Gorillaz Gorillaz Artist link to view the artist's page."

Plan C

1. Click on the Show flights button to view the available flight options.",
2. Click on the Continue button to proceed with the booking process.",
3. Select 17 April 2023 from the available grid cell for the departure date.",
4. Choose the One way option for the type of trip.",
5. Click on the Paris France CDG Charles De Gaulle Airport menu item to select the departure location."

Plan D

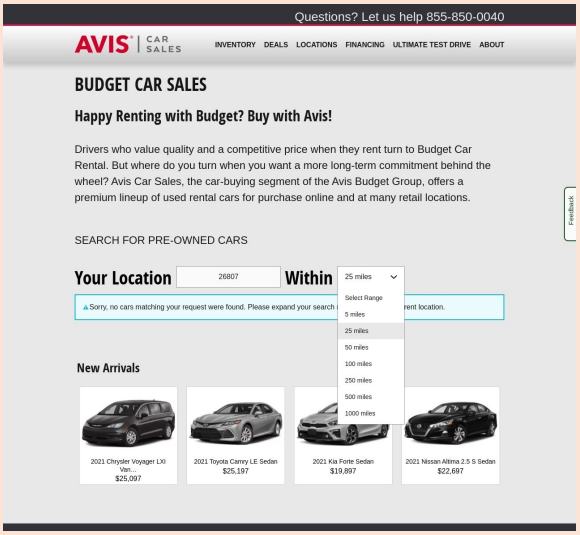
1. Click on the Continue button to proceed with the booking process.",
2. Select 17 April 2023 from the available grid cell for the departure date.",
3. Choose the One way option for the type of trip.",
4. Click on the Paris France CDG Charles De Gaulle Airport menu item to select the departure location.",
5. Click on the Show flights button to view the available flight options."

Reasoning: "REASONING: \n1. The task requires showing a list of one-way flights today (April 17) from Chicago to Paris.\n2. The screenshot shows a booking form with fields for departure and destination airports, trip type, and departure date.\n3. Plan A starts by clicking the \"Show flights\" button, which is not the correct first step since the departure date needs to be set first.\n4. Plan B selects the destination airport first, which is incorrect since the departure date needs to be set first.\n5. Plan C starts by clicking the \"Show flights\" button, which is incorrect as the departure date needs to be set first.\n6. Plan D starts by clicking the \"Continue\" button, which is not the correct first step since the departure date needs to be set first.\n7. None of the plans correctly set the departure date as the first step, which is necessary to complete the task.\n\nOUTPUT: None of the plans (A, B, C, D) effectively accomplish the task.\n"

Output Choice: None of the plans (A, B, C, D) effectively accomplish the task.\n

(a) Shuffling Perturbation: Example 1

Task: Find a Buick for sale within 250 miles from 26807.



Plan A

1. Click on the image to view a larger picture of the vehicle.
2. Select the Buick checkbox to filter the vehicle options.
3. Click on the information icon for more details.
4. Click on the View Vehicles button to see available vehicles.
5. Click on the 250 miles from the available options.

Plan B

1. Click on the 250 miles from the available options.
2. Click on the View Vehicles button to see available vehicles.
3. Click on the information icon for more details.
4. Select the Buick checkbox to filter the vehicle options.
5. Click on the image to view a larger picture of the vehicle.

Plan C

1. Click on the image to view a larger picture of the vehicle.
2. Select the Buick checkbox to filter the vehicle options.
3. Click on the information icon for more details.
4. Click on the 250 miles from the available options.
5. Click on the View Vehicles button to see available vehicles.

Plan D

1. Select the Buick checkbox to filter the vehicle options.
2. Click on the information icon for more details.
3. Click on the View Vehicles button to see available vehicles.
4. Click on the 250 miles from the available options.
5. Click on the image to view a larger picture of the vehicle.

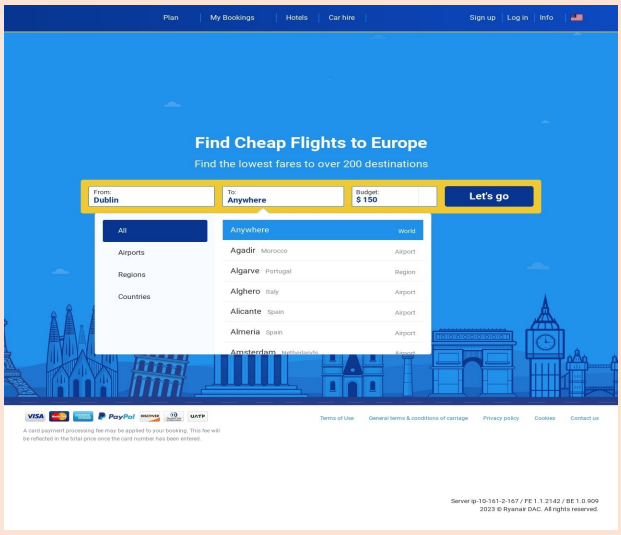
Reasoning: \n1. The task is to find a Buick for sale within 250 miles from 26807.\n2. The screenshot shows a search interface with options to enter a location and select a range.\n3. Plan A starts with an irrelevant action (viewing a larger picture of the vehicle) and does not include selecting the Buick checkbox.\n4. Plan B also starts with an irrelevant action (viewing a larger picture of the vehicle) and does not include selecting the Buick checkbox.\n5. Plan C starts with an irrelevant action (viewing a larger picture of the vehicle) and does not include selecting the Buick checkbox.\n6. Plan D includes selecting the Buick checkbox, viewing vehicle details, and selecting the 250-mile range, which are all relevant actions to complete the task.

Output: D\n

(b) Shuffling Perturbation: Example 2

Figure A.20

Task: Use the FareFinder to find flights for $150 or less from Dublin to anywhere.



Plan A

1. Click the Let's go button to continue with the booking process.
2. Select the price indicator showing $150 for further details.
3. Click on the 'Anywhere' icon to proceed with the selection.
4. Click on the 'To' section to explore destination options.

Plan B

1. Select the price indicator showing $150 for further details.
2. Click the Let's go button to continue with the booking process.
3. Click on the 'Anywhere' icon to proceed with the selection.
4. Click on the 'To' section to explore destination options.

Plan C

1. Click the Let's go button to continue with the booking process.
2. Select the price indicator showing $150 for further details.
3. Click on the 'To' section to explore destination options.
4. Click on the 'Anywhere' icon to proceed with the selection.

Plan D

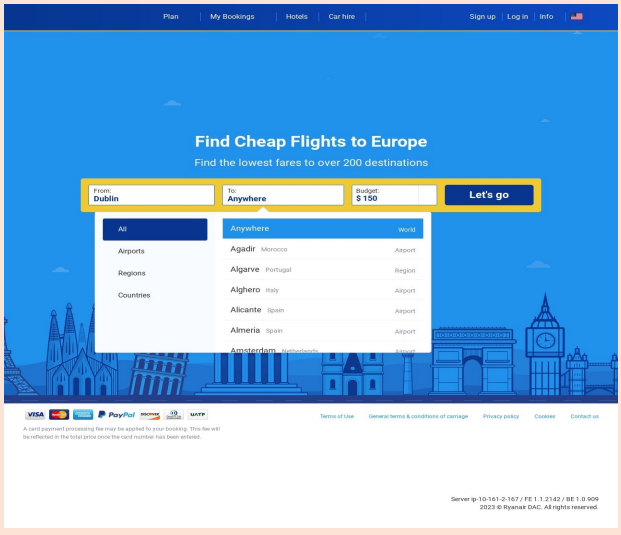
1. Click on the 'To' section to explore destination options.
2. Click on the 'Anywhere' icon to proceed with the selection.
3. Select the price indicator showing $150 for further details.
4. Click the Let's go button to continue with the booking process.

Reasoning:\n1. The task is to find flights for $150 or less from Dublin to anywhere using the FareFinder.\n2. The screenshot shows the FareFinder interface with fields for \"From,\" \"To,\" and \"Budget,\" and a \"Let's go\" button.\n3. Plan A starts by clicking the \"Let's go\" button, which is not the correct first step since the budget needs to be set first.\n4. Plan B starts by selecting the price indicator showing $150, which is the correct first step, followed by clicking the \"Let's go\" button, which is the correct second step.\n5. Plan C starts by clicking the \"Let's go\" button, which is incorrect as the budget needs to be set first.\n6. Plan D starts by clicking on the \"To\" section, which is incorrect as the budget needs to be set first.

Output: B\n

(c) Shuffling Perturbation: Example 3

Task: Use the FareFinder to find flights for $150 or less from Dublin to anywhere.



Plan A

1. Click the Let's go button to continue with the booking process.
2. Select the price indicator showing $150 for further details.
3. Click on the HAPPY HOUR BURGER link for the event on Thursday, April 13, 2023.
4. Click on the 'To' section to explore destination options.

Plan B

1. Click on the Flight status link to check the details.
2. Click the Let's go button to continue with the booking process.
3. Click on the 'Anywhere' icon to proceed with the selection.
4. Click on the 'To' section to explore destination options.

Plan C

1. Click the Let's go button to continue with the booking process.
2. Select the price indicator showing $150 for further details.
3. In the ZIP text box, type \"11101\" as the postal code.
4. Click on the 'Anywhere' icon to proceed with the selection.

Plan D

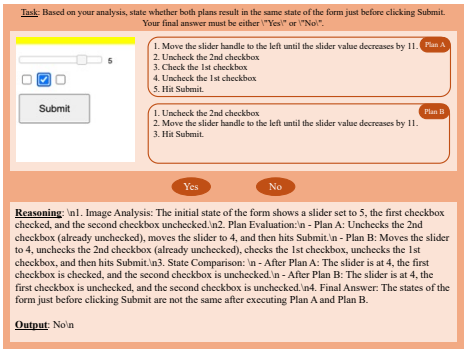
1. Click on the 'To' section to explore destination options.
2. Click on the 'Anywhere' icon to proceed with the selection.
3. Select the price indicator showing $150 for further details.
4. Click the Let's go button to continue with the booking process.

Reasoning: \n1. The task is to find flights for $150 or less from Dublin to anywhere using the FareFinder.\n2. The previous actions show that the user has already entered Dublin as the departure location and is currently selecting the destination.\n3. In the screenshot, the user has selected \"Anywhere\" as the destination and set the budget to $150.\n4. Plan A involves checking flight status and selecting a price indicator, which is not relevant to the task.\n5. Plan B involves typing an unrelated location and clicking the Let's go button, which is not relevant to the task.\n6. Plan C involves typing a postal code, which is not relevant to the task.\n7. Plan D involves selecting \"Anywhere\" as the destination and then selecting the price indicator showing $150, which aligns with the task.\n8. Plan D is the only plan that directly addresses the task and can be completed within the given 4 steps.

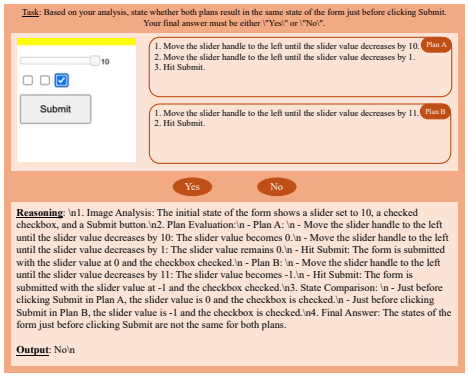
Output: D\n

(d) Semantic Perturbation: Example 4

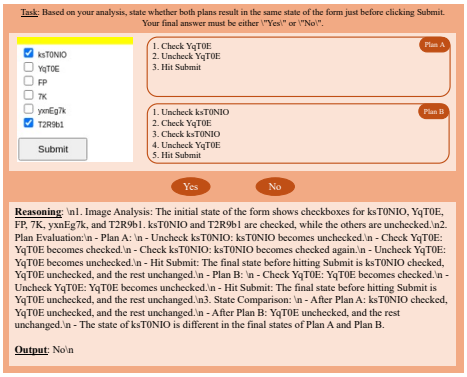
Figure A.20: Qualitative results for Plan Assessment - Real web scenario. The above figure shows outputs from Qwen2-VL 72B. Example 1, 2, 3 includes results for Shuffling Perturbation whereas Example 4 is the result for Semantic Perturbation.



(a) Example 1



(b) Example 2



(c) Example 3

Figure A.21: Qualitative results for Future State Prediction. The above figure shows outputs from Qwen2-VL 72B.

Test Name	Configuration	Number of Examples
Temporal Ordering - MM-M2W	Total	**2018**
	Order1	1009
	Order2	1009
Temporal Ordering - GUIAct	Total	**4742**
	Order1	2371
	Order2	2371
Future State Prediction - Long Horizon	Total	**3838**
	Task 1	960
	Task 2	1816
	Task 3	1062
Future State Prediction - Short Horizon	Total	**3358**
	Task 1	960
	Task 2	1336
	Task 3	1062
Plan Selection - Edge Case Scenario	Total	**1530**
	Task 1	531
	Task 2	500
	Task 3	499
Plan Selection - Real Web Environment	Total	**435**

Table A.6: Dataset Statistics: Number of examples for each test and configuration.

assess whether VLMs can reason about plans even after getting an accurate and complete textual description of the form screenshot. For simplicity, we conduct this test using Task 1 from the Edge Case Test Study and evaluate five VLMs. We compare performance across three configurations – Image Only, Caption Only, and Image + Captions. Our results in Table A.10 reveal two key findings: (1) Although adding ground truth captions improves performance over the Image-Only configuration, the selected models still fall significantly short of reasonable performance, and (2) while ground truth captions provide slight improvement, the Image + Caption configuration is either worse than Caption-Only or offers no noticeable advantage. Thus, highlighting critical limitations in the reasoning and planning capabilities of contemporary VLMs.

A.10 Additional Related Work: Importance of Small Models for Web Planning

While several recent studies have employed proprietary models for web agents, our work emphasizes the use of open-source models, many of which—despite being small—have been widely adopted in the web planning literature. For instance, prior

works [4, 7, 48] utilize models such as Qwen-VL and Minicpm, while others leverage models like Phi-3.5v, Ferret-UI, UIX (LLaVA-based), and CogAgent for GUI-specific tasks [7, 16, 65]. Unlike models that solely generate actions, many of these approaches train models to produce thoughts or reasoning steps prior to action execution, thereby implicitly requiring planning capabilities. Notably, Qwen2-VL models have been explicitly trained with planning and reasoning datasets to support such behavior.

Given the high costs, proprietary constraints, and privacy concerns associated with large-scale commercial models, the broader research community is increasingly gravitating toward smaller, open-access alternatives. Our study aligns with this direction by providing timely insights into the intrinsic planning capabilities of vision-language models (VLMs) that are already seeing practical use in web-based tasks.

Although resource limitations prevented us from conducting the full experiment suite on proprietary models, we conducted additional trials using GPT-4o-mini to demonstrate the broader potential of our approach. In this work, we focus on two core tasks—Temporal Ordering and Plan Assessment for real-world web environments. For temporal prediction, we utilize the GUIAct dataset and evaluate a sample of 500 questions from the full set. For Plan Assessment, we assess model behavior under the Shuffle Perturbation setting.

Main	Task 1		Task 2		Task 3		Avg.
	INS	COT	INS	COT	INS	COT	
Qwen2-VL 2B	73.43	63.64	43.98	49.32	49.62	47.08	54.51
Qwen2-VL 7B	99.27	90.48	50.81	81.27	50.00	57.25	71.51
Qwen2-VL 72B	97.50	97.91	97.66	98.37	99.71	95.76	97.82
Minicpm	52.39	66.67	50.99	64.93	50.00	56.04	56.84
Minicpm-o	70.10	82.25	40.76	67.47	60.64	55.57	62.80
Phi 3.5	87.50	81.77	61.19	62.42	53.86	56.68	67.24
IDEFICS3	53.22	96.56	0.5	87.34	50.28	54.80	57.12
DeepSeek VL	58.85	79.89	44.35	53.02	50.65	39.64	54.40
Llava OV 0.5B	50.00	43.95	50.00	40.78	52.35	48.11	47.53
Llava OV 7B	55.83	87.18	52.14	76.47	60.16	65.72	66.25
InternVL2-MPO	62.91	96.77	20.96	92.04	28.53	72.10	62.22
Glm	50.31	68.64	50.23	57.07	63.46	58.28	58.00
Llava 1.6 - 7B	92.08	82.29	58.91	78.08	44.82	32.16	64.72
Llava 1.6 - 13B	50.00	76.14	50.00	63.46	50.00	47.87	56.25
Llava 1.6 - 34B	91.14	91.87	83.03	81.94	65.91	55.36	78.21
Ferret-UI-Llama	50.00	50.10	50.00	49.88	40.29	50.00	48.38
Ferret-UI-Gemma	69.00	5.72	44.73	2.75	64.23	10.07	32.75
UIX-Qwen2	60.52	70.20	59.94	63.00	32.01	54.80	56.75
CogAgent	50.00	30.93	50.00	32.06	50.00	43.97	42.83

Table A.7: Task-wise results for Future State Prediction in short horizon action scenarios

Main	Task 1		Task 2		Task 3		Avg.
	INS	COT	INS	COT	INS	COT	
Qwen2-VL 2B	52.60	51.56	48.95	48.12	49.71	48.39	49.89
Qwen2-VL 7B	84.27	55.75	68.07	38.5	50.0	50.90	57.91
Qwen2-VL 72B	100	57.70	45.81	55.67	84.83	88.32	55.55
Minicpm	50.0	48.00	50.00	2.29	50.00	56.42	42.78
Minicpm-o	50.41	54.08	86.89	40.27	36.15	52.14	53.32
Phi 3.5	74.06	54.27	11.56	34.14	49.71	53.01	46.13
IDEFICS3	50.83	51.97	49.39	34.63	50.00	64.87	50.28
DeepSeek VL	50.00	51.56	30.34	45.26	49.43	40.58	44.53
Llava OV 0.5B	50.00	42.81	50.00	40.91	47.83	48.30	46.64
Llava OV 7B	50.00	51.66	51.10	44.49	49.24	60.73	51.20
InternVL2-MPO	75.62	65.52	39.20	46.21	48.77	67.48	57.13
Glm	50.62	55.20	50.00	43.85	48.68	52.30	50.11
Llava 1.6 - 7B	58.22	56.04	22.19	41.18	41.43	48.68	44.62
Llava 1.6 - 13B	50.00	55.41	50.00	34.58	50.0	47.12	47.85
Llava 1.6 - 34B	72.08	65.729	22.46	48.84	43.97	46.79	49.98
Ferret-UI-Llama	100	49.79	50.0	49.77	49.23	50.00	58.13
Ferret-UI-Gemma	52.36	4.89	44.80	6.55	49.92	6.40	27.49
UIX-Qwen2	51.14	51.04	50.00	44.76	19.20	44.35	43.41
CogAgent	50.00	48.54	50.00	38.21	50.00	42.93	46.61

Table A.8: Task-wise results for Future State Prediction in long horizon action scenarios

Main	Task 1		Task 2		Task 3	
	INS	COT	INS	COT	INS	COT
Qwen2-VL 2B	27.11	28.92	26.66	32.20	59.71	45.29
Qwen2-VL 7B	13.74	11.86	67.20	62.40	67.73	69.27
Qwen2-VL 72B	95.29	88.51	83.40	81.40	87.57	58.83
Minicpm	1.6	10.54	42.60	25.00	57.11	65.93
Minicpm-o	26.36	20.15	49.60	32.60	67.33	59.31
Phi 3.5	21.84	12.05	26.00	16.00	50.70	50.50
IDEFICS3	27.11	43.69	59.20	64.00	52.30	59.91
DeepSeek VL	0.9	12.24	13.80	38.60	40.28	33.66
Llava OV 0.5B	14.87	20.52	28.60	22.20	14.87	22.24
Llava OV 7B	19.39	13.93	65.20	53.60	28.05	47.69
InternVL2-MPO	45.38	24.14	75.60	66.20	52.10	57.11
Glm	25.80	28.86	54.00	42.60	60.52	55.22
Llava 1.6 - 7B	26.55	31.45	33.40	41.60	28.05	47.69
Llava 1.6 - 13B	26.36	32.01	24.40	33.60	27.51	30.26
Llava 1.6 - 34B	14.90	11.48	35.60	29.20	70.54	48.49
Ferret-UI-Llama	28.06	23.6	18.40	23.60	16.03	20.64
Ferret-UI-Gemma	10.73	12.42	8.00	12.80	16.43	14.62
UIX-Qwen2	0.3	10.73	37.00	8.00	30.66	24.64
CogAgent	16.57	22.22	41.40	21.80	21.04	19.23

Table A.9: Task-wise results for Plan Selection – Edge case Test

	Img Only			Img + Cap			Cap Only		
	INS	CoT	Avg	INS	CoT	Avg	INS	CoT	Avg
Minicpm-o	26.36	20.15	23.26	33.90	20.53	27.21	34.46	20.90	27.68
Qwen2-VL-7B	13.74	11.86	12.81	15.25	3.58	9.42	13.94	9.04	11.49
Phi 3.5	21.84	12.05	16.95	32.77	25.61	29.19	32.96	27.87	30.41
Glm	25.80	28.86	27.33	23.35	22.03	22.69	22.22	24.29	23.26
UIX-Qwen2	0.3	10.73	5.56	4.52	9.04	6.78	3.77	9.04	6.40

Table A.10: Performance comparison under different configurations (Img Only, Img + Cap, and Cap Only) across INS, CoT, and Avg metrics.

Appendix B

Additional material for Chapter 2

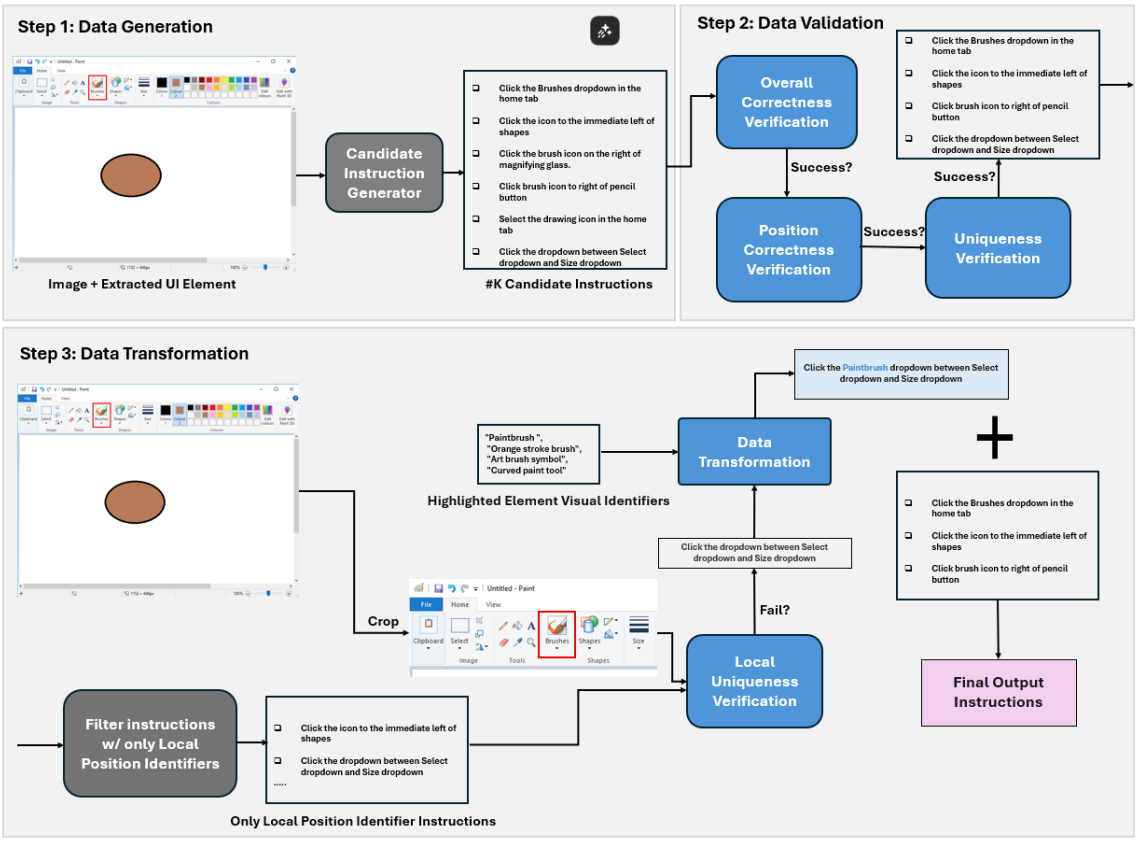
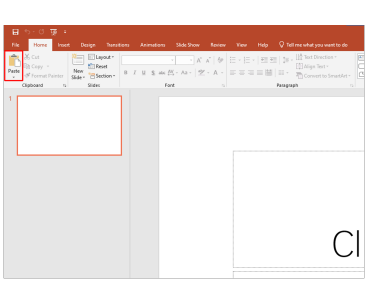


Figure B.1: Data synthesis pipeline for generating multiple instructions per UI element. For more details, refer to Sec 3.2 for Step 1, Sec 3.3 for Step 2, Sec 3.4 for Step 3.

B.1 Additional Visualizations

In this section, we present additional visual illustrations that were omitted from the main paper due to space constraints: 1) Fig B.1 is our data generation pipeline in Sec 3, 2) Fig B.2 includes dataset examples for our GUI Grounding Sensitivity Benchmark, 3) Fig B.3 includes qualitative outputs of state-of-the-art GUI grounding models on our benchmark, 4) Fig B.4 is the visualization for Local Crop View and Windows-In-Background View.



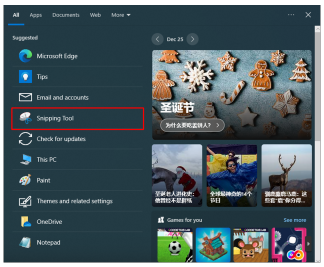
"Click the Paste icon in the Clipboard section“

"Use the Paste option in the upper left corner",

"Hit the downward arrow under the clipboard image in Home"

"Click the clipboard icon in the Home tab section"

"Select the large button above the word Paste in the ribbon"



"Select the option in the Suggested section just after Email and accounts"

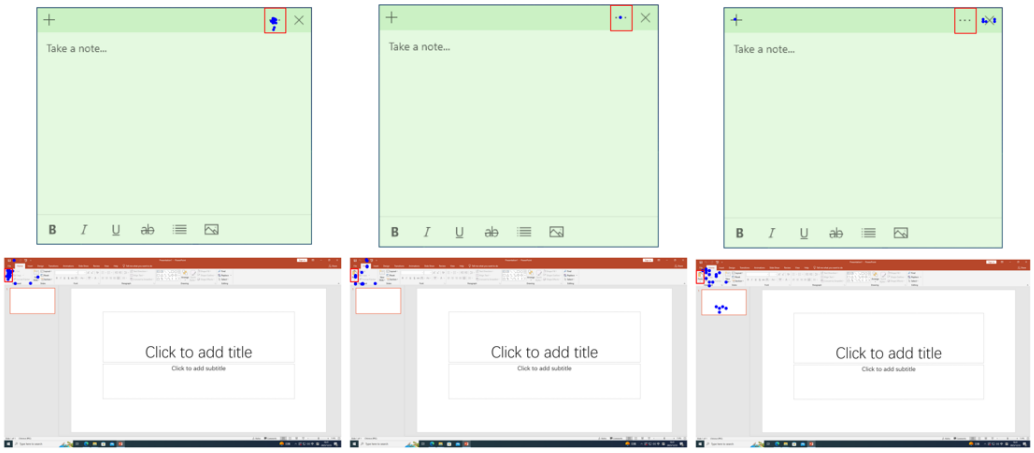
"Open the white oval feature between the envelope and the circular arrow"

"Launch the fourth item in the Suggested list“

"Click the scissors icon in the Suggested section"

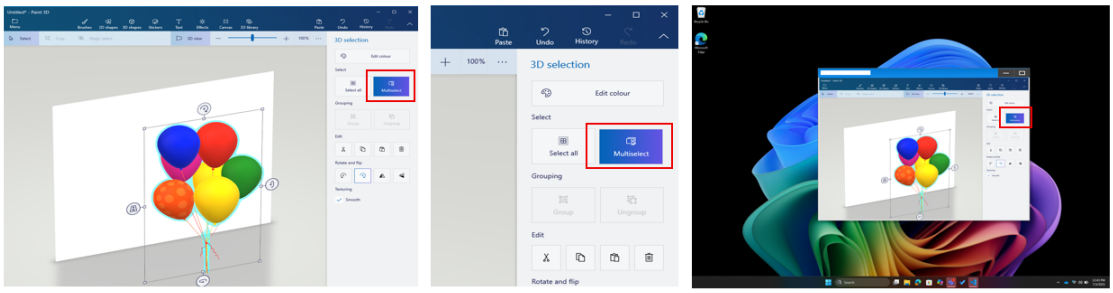
"Select the tool between Email and accounts and Check for updates",

Figure B.2: Dataset examples from the GUI Grounding Sensitivity Benchmark. For illustration, we display only 5 instructions per UI elements.



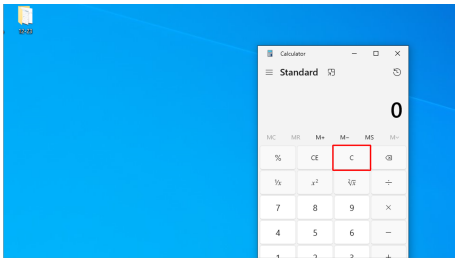
(a). Operator **(b).** UGround-V1-7B **(c)** ShowUI-2B

Figure B.3: Qualitative visualizations of grounding model outputs on diverse instructions referring to the same element show that the predicted coordinates often vary within a region, appear in separate areas, or fall outside the target bounding box.

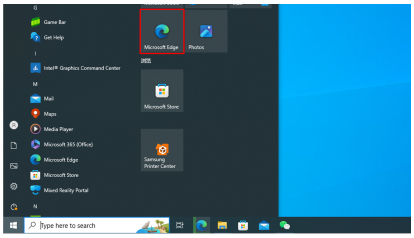


(a) Original Image (b) Local Crop (c) Window-in-Background

Figure B.4: Diverse ways in which UI elements can appear



Reason: 'C' and 'CE' (ID 14) are very similar both visually and functionally. References to 'clear' are highly ambiguous.



Reason: There is also a Microsoft Edge entry in the left app list (ID 11) and a Microsoft Edge shortcut in the taskbar (ID 20). All share the same icon and label, and a generic instruction ('open Edge') could refer to any. The tile and list versions are especially similar.

Figure B.5: Qualitative visualization of the VLM's inferred reasoning for why a UI element is confusing to ground (See Eqn 4 in the main paper).

B.2 GUI Grounding Performance over granularity of instructions

Results in Sec 3.3.6 demonstrate that current models struggle in consistently predicting outputs over multiple ways to describe UI elements. In this section, we understand the performance as a variation of instruction specificity. More particularly, we consider three levels of specificity: 1) High – most direct and unambiguous combination of visual and position indicators, 2) Medium – relatively less specific visual indicators and position indicators, 3) Low – no visual indicator and only relies on position indicator. Note that "no position indicator and only visual indicator" is not plausible because images often have more than one UI element of same visual appearance, and it would not necessarily produce a correct instruction. We report the Instruction Out-of-Box rate (s_{oob}) in Fig. B.6 and observe a clear trend: model performance degrades (higher s_{oob}) as instruction specificity decreases, highlighting their sensitivity and supporting our hypothesis.

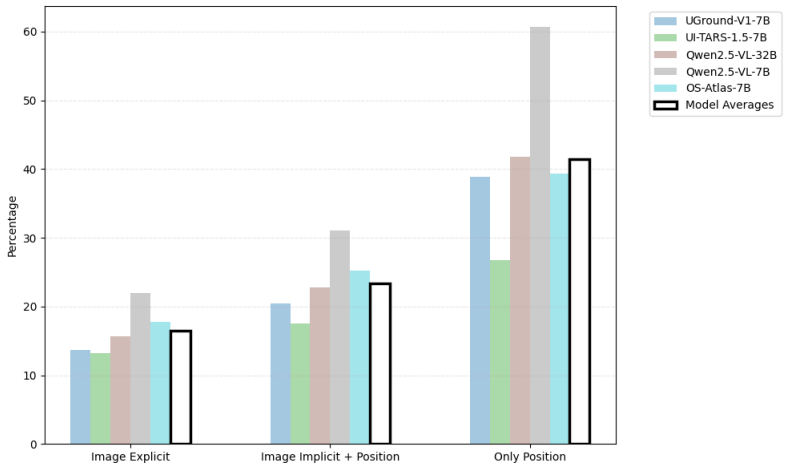


Figure B.6: Grounding Performance over granularity of Instructions

B.3 Unreliability to generate descriptions of small UI elements

As discussed in Section 3.2, considering both the visibility score and the bounding box of UI elements is crucial to ensure that proprietary VLMs (such as GPT-4.1) can accurately interpret the elements and generate multiple valid, high-quality instructions. In Fig B.7, we present empirical results supporting our observation that instruction generation is often unreliable for small UI elements.

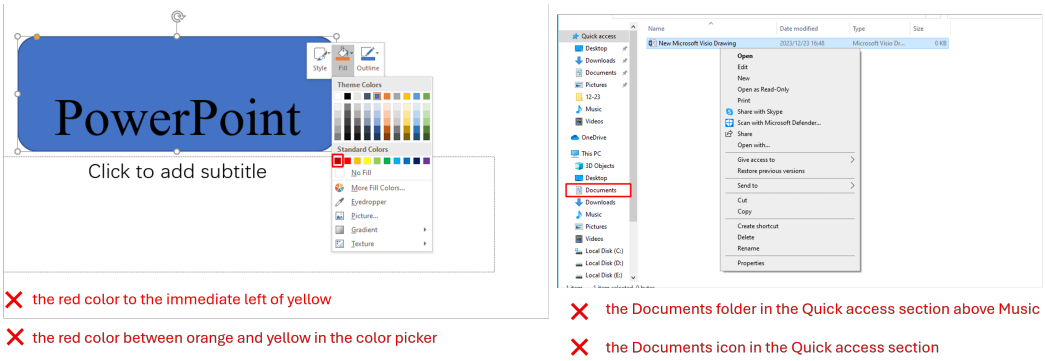
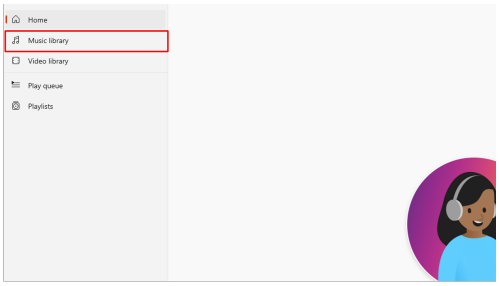


Figure B.7: Proprietary VLMs like GPT-4.1 generate incorrect referring instructions for small elements (shown in red bounding box).

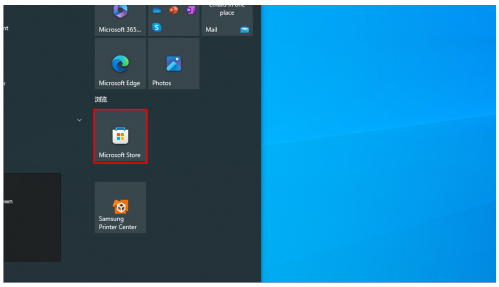
B.4 Limitations of Data Validation

As discussed in Section 3.4, even after the data validation step, some instructions may still ambiguously refer to multiple UI elements within the image, i.e, failing to uniquely identify the target element e. We now present qualitative examples in Fig. B.8 to better understand on what kind of samples does the validation process fails.



Choose the library entry in the sidebar below Home

Reason: There are two library icons – Music and Video and both are below Home



Open the app tile lies between Photos and Samsung Printer Center

Reason: There are two apps – Microsoft Edge and Microsoft Store

Figure B.8: Examples where Data Validation step (Sec 3.3) fails to filter out instructions that do not uniquely identify the target element.

Bibliography

[1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024. URL https://arxiv.org/abs/2402.14740.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

[3] Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Jang, and Zack Hui. Windows agent arena: Evaluating multi-modal os agents at scale, 2024. URL https://arxiv.org/abs/2409.08264.

[4] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Guicourse: From general vision language models to versatile gui agents, 2024.

[5] Xingyu Chen, Zihan Zhao, Lu Chen, JiaBao Ji, Danyang Zhang, Ao Luo, Yuxuan Xiong, and Kai Yu. WebSRC: A dataset for web-based structural reading comprehension. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4173–4185, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.343. URL https://aclanthology.org/2021.emnlp-main.343/.

[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, 2024. URL https://arxiv.org/abs/2401.10935.

[7] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.505.

[8] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=kiYqbO3wqw.

[9] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023. URL https://arxiv.org/abs/2306.06070.

[10] Aaron Grattafiori et. al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[11] Marah Abdin et. al. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

[12] Akash Ghosh, Arkadeep Acharya, Sriparna Saha, Vinija Jain, and Aman Chadha. Exploring the frontier of vision-language models: A survey of current methodologies and future directions, 2024. URL https://arxiv.org/abs/2404.07214.

[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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kxnoqaisCT.

[14] Yu Gu, Boyuan Zheng, Boyu Gou, Kai Zhang, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your llm secretly a world model of the internet? model-based planning for web agents, 2024. URL https://arxiv.org/abs/2411.06559.

[15] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

[16] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents, 2024. URL https://arxiv.org/abs/2312.08914.

[17] 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.

[18] Zheng Hui, Yinheng Li, Dan zhao, Tianyi Chen, Colby Banbury, and Kazuhito Koishida. Winclick: Gui grounding with multimodal large language models, 2025. URL https://arxiv.org/abs/2503.04730.

[19] Surgan Jandial, Yinong Oliver Wang, Andrea Bajcsy, and Fernando De la Torre. On the fine-grained planning abilities of VLM web agents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 25347–25380, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1382. URL https://aclanthology.org/2025.findings-emnlp.1382/.

[20] Yuanhan Jiang, Yiyang Gu, Hervé Jégou, and Adam Lerer. Building and better understanding vision-language models: insights and future directions, 2024.

[21] Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web, 2024. URL https://arxiv.org/abs/2402.17553.

[22] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks, 2024. URL https://arxiv.org/abs/2401.13649.

[23] Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents, 2024. URL https://arxiv.org/abs/2407.01476.

[24] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

[25] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[26] 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, 2025. URL https://arxiv.org/abs/2504.07981.

[27] Lin Li, Guikun Chen, Hanrong Shi, Jun Xiao, and Long Chen. A survey on multimodal benchmarks: In the era of large ai models, 2024. URL https://arxiv.org/abs/2409.18142.

[28] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent, 2024. URL https://arxiv.org/abs/2411.17465.

[29] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1802.08802.

[30] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

[31] Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. Harnessing webpage uis for text-rich visual understanding, 2024. URL https://arxiv.org/abs/2410.13824.

[32] Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. Harnessing webpage uis for text-rich visual understanding, 2024. URL https://arxiv.org/abs/2410.13824.

[33] Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding?, 2024.

[34] Xinyi Liu, Xiaoyi Zhang, Ziyun Zhang, and Yan Lu. Ui-e2i-synth: Advancing gui grounding with large-scale instruction synthesis, 2025. URL https://arxiv.org/abs/2504.11257.

[35] Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners, 2025. URL https://arxiv.org/abs/2504.14239.

[36] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. Deepseek-vl: Towards real-world vision-language understanding, 2024.

[37] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent, 2024. URL https://arxiv.org/abs/2408.00203.

[38] Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning, 2025. URL https://arxiv.org/abs/2503.21620.

[39] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai, 2024. URL https://arxiv.org/abs/2405.14093.

[40] Mikołaj Małkiński, Szymon Pawlonka, and Jacek Mańdziuk. Reasoning limitations of multimodal large language models. a case study of bongard problems, 2024. URL https://arxiv.org/abs/2411.01173.

[41] MiniCPM-o Team. MiniCPM-o 2.6: A GPT-4o-Level MLLM for Vision, Speech, and Multimodal Live Streaming on Your Phone, 2025. Accessed: January 31, 2025.

[42] Aishik Nagar, Shantanu Jaiswal, and Cheston Tan. Zero-shot visual reasoning by vision-language models: Benchmarking and analysis, 2024. URL https://arxiv.org/abs/2409.00106.

[43] Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M. Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction, 2025. URL https://arxiv.org/abs/2503.15661.

[44] Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. Gui agents: A survey, 2024. URL https://arxiv.org/abs/2412.13501.

[45] OpenAI. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

[46] OpenAI. Gpt-4.1: Improved coding, instruction-following, and long-context abilities. API release blog post, OpenAI, April 2025.

[47] OpenAI. Operator: A computer-using ai agent. arXiv preprint arXiv:2501.10114, 2025. Research preview of the "Operator" AI agent; released January 23, 2025.

[48] Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Awadallah. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents, 2025. URL https://arxiv.org/abs/2502.11357.

[49] Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents, 2024.

[50] 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.

[51] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi

Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[52] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control, 2023. URL https://arxiv.org/abs/2307.10088.

[53] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2025. URL https://arxiv.org/abs/2405.14573.

[54] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[55] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/shi17a.html.

[56] Segev Shlomov, Aviad Sela, Ido Levy, Liane Galanti, Roy Abitbol, et al. From grounding to planning: Benchmarking bottlenecks in web agents. *arXiv preprint arXiv:2409.01927*, 2024.

[57] Meta Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[58] Maria Wang, Srinivas Sunkara, Gilles Baechler, Jason Lin, Yun Zhu, Fedir Zubach, Lei Shu, and Jindong Chen. Webquest: A benchmark for multimodal qa on web page sequences, 2024. URL https://arxiv.org/abs/2409.13711.

[59] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and

Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[60] Weiyun Wang, Zhe Chen, Wenhai Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu, Xizhou Zhu, Lewei Lu, Yu Qiao, and Jifeng Dai. Enhancing the reasoning ability of multimodal large language models via mixed preference optimization, 11 2024.

[61] 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.

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

[63] Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis, 2025. URL https://arxiv.org/abs/2505.13227.

[64] Yibin Xu, Liang Yang, Hao Chen, Hua Wang, Zhi Chen, and Yaohua Tang. Deskvision: Large scale desktop region captioning for advanced gui agents, 2025. URL https://arxiv.org/abs/2503.11170.

[65] 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, 2025. URL https://arxiv.org/abs/2412.04454.

[66] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *ArXiv*, preprint.

[67] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

[68] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding withnbsp;multimodal llms. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXIV*, page 240–255, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73038-2. doi: 10.1007/978-3-031-73039-9_14. URL https://doi.org/10.1007/978-3-031-73039-9_14.

[69] Yanzhe Zhang, Tao Yu, and Diyi Yang. Attacking vision-language computer agents via pop-ups, 2025. URL https://arxiv.org/abs/2411.02391.

[70] Ziniu Zhang, Shulin Tian, Liangyu Chen, and Ziwei Liu. Mmina: Benchmarking multihop multimodal internet agents, 2024.

[71] Haoren Zhao, Tianyi Chen, and Zhen Wang. On the robustness of gui grounding models against image attacks, 2025. URL https://arxiv.org/abs/2504.04716.

[72] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024. URL https://arxiv.org/abs/2401.01614.