Navi-*plus*: Managing Ambiguous GUI Navigation Tasks with Follow-up Questions

Anonymous ACL submission

Abstract

Graphical user interfaces (GUI) automation agents are emerging as powerful tools, enabling humans to accomplish increasingly complex tasks on smart devices. However, users often inadvertently omit key information when conveying tasks, which hinders agent performance in the current agent paradigm that does not support immediate user intervention. To address this issue, we introduce a **Self-Supplement GUI Navigation** task that incorporates interactive information completion capabilities within GUI agents. We developed the **Navi-*plus*** dataset with GUI follow-up question-answer pairs, alongside a **Dual-Stream Trajectory Evaluation** method to benchmark this new capability. Our results show that agents equipped with the ability to ask GUI follow-up questions can interact with human users and recover their performance when faced with ambiguous user tasks.

1 Introduction

Graphical User Interface (GUI) becomes the foundational approach of modern human-computer interaction with increasing numbers of screens filling people's lives. To augment human capabilities and mitigate mental burdens in operating digital devices, GUI automation agents have arisen in recent years. Following the advancements of (Multimodal) Large Language Models (LLMs or MLLMs), extensive efforts were invested in constructing these autonomous agents through large-scale continual pre-training (Cheng et al., 2024, Chai et al., 2024, Lin et al., 2024), grounding-augmented supervised fine-tuning (Li et al., 2024a, Sun et al., 2024), and utilization of Chain-of-Action-Thought (CoAT) in navigation(Zhang et al., 2024, Liu et al., 2025).

However, the previous paradigm of GUI navigation agents is limited to receiving full human instruction and performing actions serially, diminishing human control halfway through agent processing. Thus, a practical "elephant in the room" question arises: ***If some important information is missing from the human instruction (i.e., product specs or important dates), how can the agent continue the task that it is expected to finish?***

Figure 1: Overview of our proposed Self-Supplement GUI Navigation task. Agent proactively asks for missing information when the task is ambiguous.

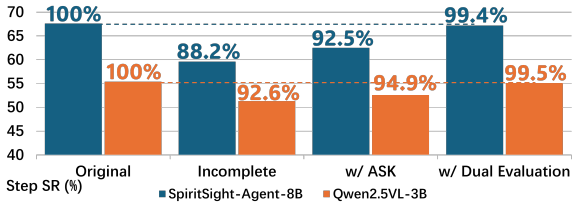


Figure 2: Comparison of agents' Step SR scores on original tasks and generated ambiguous tasks of AndroidControl-Navi*plus* data. Completing the task with ASK action can recover agents' performance.

This issue has motivated us to review the formulation of the current GUI navigation task. In this paper, we propose a novel task called **Self-Supplement GUI Navigation**, which endows GUI agents with a new ability to handle the ambiguity of human instruction. The core idea is to add an "ASK" action in the agents' action space, enabling it to engage in intermediate natural language inter-

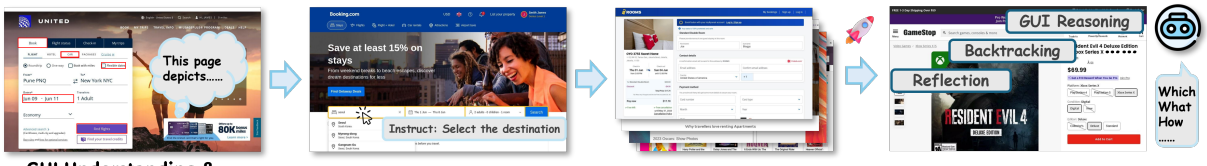


Figure 3: GUI tasks have developed from GUI environment understanding, through single-step instruction execution and multi-step navigation, and are evolving into more proactive and helpful assistants in the digital world.

actions with human users by proposing follow-up questions (Figure 1).

We first design a data annotation pipeline to construct Navi*plus* dataset from existing trajectory datasets using open-source LLMs(MLLMs). GUI navigation trajectories with ambiguous task descriptions are intentionally and controllably generated, along with corresponding GUI follow-up question-answering (QA) pairs.

We then include fair and comprehensive evaluation metrics to benchmark GUI agents' capability to complete the GUI navigation task when the task description is ambiguous. A **Dual-Stream Trajectory Evaluation** method is proposed to separately compute metrics for the operational actions (e.g., Step Success Rate) and for the additional ASK action, allowing direct model comparison.

Our experiments show that the missing information in task description can significantly harm GUI agents task success rate, but with the completion of ASK action, agents can restore up to 99.4% of performance (Figure 2). Moreover, we find that modern MLLM-based GUI agents can seamlessly learn to propose follow-up questions and achieve satisfactory performance, with a timing accuracy of up to 0.947 and a content similarity of up to 0.832. Through extensive experiments, we further demonstrate that both model scale and dataset size affect performance on the proposed self-supplement GUI navigation task.

2 Related Work

2.1 GUI Navigation Agents and Datasets

GUI navigation agents (GUI agents in short) are data-driven, end-to-end and pure-vision-based agent models (Qin et al., 2025) that comprehend human instructions, perceive the virtual environment, and automate operations in the UI world. Simply providing the task with one sentence, the agent that functioned by a single model captures the screenshots and directly output the action to be conducted with one inference. Standing on the shoulder of vision-language foundation models, GUI agents further improve their capability in (1) GUI environment understanding, (2) operational elements grounding, and (3) navigation task planning. See Appendix E for detailed discussion.

In previous GUI navigation datasets, however, task instructions are deliberately designed to be unambiguous to ensure a unique action path. In this paper, we continue to advance the development of GUI navigation agents by addressing the unavoidable problem of ambiguous user task presentations, thereby enhancing the agents' capabilities and flexibility in real-world application scenarios. We aim for a proactive GUI assistant experience. (Figure 3).

2.2 ASK Augmentation for Conversational Agents

Conversational AI agents emulate human conversations by understanding intentions and interacting with the provided environment to complete tasks or answer questions. The application of (multimodal) LLMs as conversational agents in various fields has garnered considerable attention, as they demonstrate remarkable performance in tasks such as decision-making (e.g. FILM (Min et al., 2021), ReAct (Yao et al., 2022)), tool usage (e.g. Toolformer (Schick et al., 2023), ToRA (Gou et al., 2023)), real-world interaction (e.g. DEPS (Wang et al., 2023), LABOR (Chu et al., 2024)), and multi-agent collaboration (e.g. CoMM (Chen et al., 2024a), L2MAC (Holt et al., 2024)).

The capability of actively putting up clarifying questions by conversational agents have been explored in the field of embodied robots (Nguyen and III, 2019; Ramrakhya et al., 2025), information retrieval (Chi et al., 2024), and text-to-SQL generation (Wu et al., 2024a). However, these previous works enhance model capability by integrating user assistance and ask human to provide explicit next-step guidance, which are not identical to GUI

agent's convention that only keep the agent responsible for decision making and execution.

Our work proposes an interactive agent task that refines user intentions with ASK to aid GUI navigation, exploring the proactive information completing capability of conversational agents in GUI scenarios.

2.3 Conversational Web Navigation

Conversational web navigation is a novel task recently presented by WebLINX (Lù et al., 2024), wherein humans provide task descriptions to agents section by section, with each containing two or three actions. MT-Mind2Web (Deng et al., 2024b) proposed synthesizing conversational navigation data with existing GUI trajectory datasets by breaking down the full task description into lower-level instructions.

Some concurrent works, like AutoGLM (Liu et al., 2024b) and CogAgent-V2 (Hong et al., 2024), demonstrated an interesting ability to judge the sensibility of the next action and remind users to double-check the model-generated action. They also empowered the agents with the ability to inquire about supposedly missing information in task descriptions through immediate long-term navigation planning. These features are briefly mentioned in their blogs and are worth further research.

In contrast, we propose a straightforward yet effective method to enable GUI automation agents to actively interact with human users for missing task information. Our method obviates the need for long-range CoAT processing and reflection, while also remaining compatible with this format.

3 Self-Supplement GUI Navigation Task

3.1 Preliminaries

Let **Task** be a natural-language task specification such as "compose and send an email to Alice."

Given a specific **Task**, a GUI-navigation episode *with n steps* is

$$E(\text{Task}) \triangleq \big\langle (s_0, a_0), \ldots, (s_{n-1}, a_{n-1}) \big\rangle, \quad s_0 \in \mathcal{S},\ a_i \in \mathcal{A}.$$

At each step i the agent receives (Task, s_i) and issues

$$a_i = \pi(\text{Task}, s_i),$$

after which the environment applies the transition function

$$s_{i+1} = T(s_i, a_i).$$

Symbol	Description
$\mathcal{S}$	**State space** – all observable GUI states
$\mathcal{A}$	**Action space** – atomic GUI actions (e.g., *click*, *type*, *swipe*)
$T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$	**Transition function** – maps a state–action pair (s, a) to the next state s'
$\pi : \text{Task} \times \mathcal{S} \to \mathcal{A}$	**Agent policy** – issues an action given the task description and current state
E	**GUI-navigation episode** – ordered sequence of state–action pairs in one run

Table 1: Notation for GUI navigation.

The episode terminates when the goal specified by the **Task** is achieved or after n steps, whichever occurs first.

In practical GUI navigation, the execution sequence interleaves **informative steps** and **transactional steps**, depending on whether the current interaction introduces a branching decision or performs an indispensable operation:

Informative steps presents alternative choices and guides high-level decision-making, introducing decision branches. eg. selecting an item from a drop-down list.

Transactional steps performs an indispensable operation required for task progress, ensuring a smooth workflow. eg. clicking an "OK" button or closing a pop-up window.

3.2 Task Formulation

When users describe tasks, it is possible that some key information be omitted. For example, when ordering oil paint online with the help of GUI agents, someone specified colors and sizes but forgot to mention the preferred delivery method due to unfamiliarity with the task flow, leading to ambiguity in the task description for the GUI navigation agent.

To address the practical challenge of ambiguous task input, we propose a novel **Self-Supplement GUI Navigation** task, aiming to benchmark and facilitate GUI agents' feasibility when facing ambiguous task inputs. GUI agents' ability to correctly continue the task and to interact with human users to complete the missing information are the two main indications we consider. (See 5.2)

Specifically, we add an **ASK** action to the

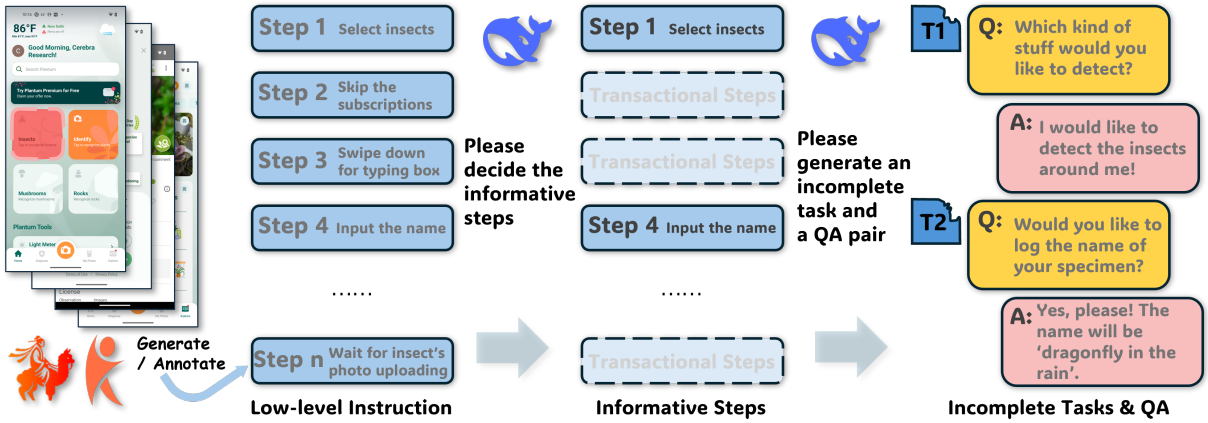


Figure 4: Illustration of Navi*plus* Dataset's construction pipeline.

model's action space for asking **GUI Follow-up Questions**, and during the interaction between the model and the user, we provide a **SAY** action for the user to fill in the missing information. The interaction between the agent and the user will be logged into the agent's context, providing information to continue task navigation.

A Self-Supplement GUI Navigation Episode is thus formally defined as

$$E^{+}(\textbf{Task}') \triangleq \langle (s_0, a_0), \ldots, (s_{n-1}, a_{n-1}) \rangle, \quad s_0 \in \mathcal{S},\ a_i \in \mathcal{A}^{+}.$$

where original task description lacks some key information and becomes **Task**$'$, Original action space is augmented by a ASK action and becomes $\mathcal{A}^{+} = \mathcal{A} \cup \{\text{ASK}\}$.

4 Navi-*plus* Dataset

4.1 Data Construction Pipeline

An overview of our data construction pipeline is illustrated in Figure 4. To intentionally generate ambiguous GUI navigation task descriptions and agent's follow-up questions for information completion, we develop this data construction process. We select AndroidControl (Li et al., 2024a) and Mind2Web (Deng et al., 2024a) as our data sources for they are collected by well-trained human annotators to ensure quality, and they cover the two most widely used device platforms: Mobile and Web.

The data construction process consists of three key steps: (1) Low-level Instruction Completion, (2) Informative Step Decision, (2) Formation of Ambiguous Tasks, and is described below in detail. To ensure the quality and reproducibility of our data, we select powerful open-source LLMs such as InternVL2.5-26B (Chen et al., 2024c) and DeepSeek-V3 (Liu et al., 2024a). The prompts and model outputs are verified by human annotators to ensure they meet or exceed the quality of GPT-4o. For prompt templates see Appendix F. For qualitative examples, see Figure X.

Low-level Instruction Completion We start by generating low-level instructions for each step in the trajectories. This intention consists of an operational intention and an element description (e.g. the 'OK' button or the 'plus' button of the second product). The current action, along with a screenshot and the bounding box of the interacted element, is provided to InternVL2.5-26B to generate the low-level instructions for that action, as shown on the left side of Figure 4. (A special case here is AndroidControl dataset originally provides human annotated low-level instructions.)

Informative Step Decision As shown in the middle of Figure 4, we then hired DeepSeek-V3 to decide whether a step is an informative step or a transactional step following the definition described in Section preliminary. Once a step is marked as transactional, it will be neglected, while only the informative steps will be included to generate QA. DeepSeek-V3 achieves a satisfactory accuracy rate when making the judgment, with 90% of the data passing human verification.

Formation of Ambiguous Tasks Finally, we present the full task description as a reference to DeepSeek-V3 and provide it with the informative steps to be removed. The model is prompted to output an ambiguous task description that excludes the selected informative steps while maintaining all other information and the original phrasing style.

A QA pair simulating the agent's follow-up question and the user's answer is also generated for each informative step, as shown on the right side of Figure 4. In practice, the judgment of informative steps and the formation of ambiguous tasks are completed in one API call to minimize costs.

5 Evaluation Methods

5.1 Dual-Stream Trajectory Evaluation

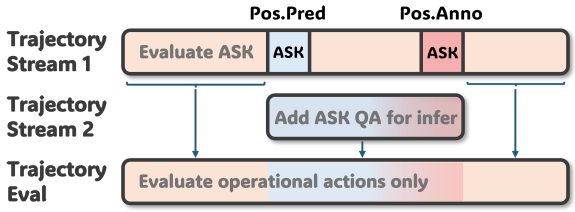


Figure 5: Visualize of our Dual-Stream Trajectory Evaluation method. Each line represents an episode's result.

Adding a new ASK action into agent action space makes the inference process and evaluation metrics inherently different from the original GUI navigation task. Since the evaluation is performed offline with pre-defined screenshots and trajectories, false-positive action predictions can occur if the ASK action appears before its annotated position. This over-strict criterion results in a decline in observed agent performance, which requires correction.

So, we propose the **Dual-Stream Trajectory Evaluation** method with two key changes as depicted in Figure 5: (1) An ASK action is considered correct within a full task trajectory if it appears at or before the annotated step position. (2) If agents ASK in advance, execute a second inference by adding the ASK QA pairs into context to recover the operational action, like Figure 1 shows.

During the second inference pass, as illustrated in Figure 5, ASK QA pairs are inserted into prompts from the predicted position to its annotated position. This design ensures that once an ASK action is invoked, it affects all subsequent steps in the trajectory, reflecting a real-world execution scenario. Moreover, the proposed method isolates the assessment of the ASK action from the evaluation of other operational steps, ensuring that the model's ability to ask questions and execute actions can be measured independently.

See Algorithm 1 for pseudo code, with an additional computational cost of O(n) for the second evaluation pass.

Algorithm 1: Dual-Stream Trajectory Eval

Input: Episodes $E = \{E_1, E_2, \ldots, E_M\}$
Output: Metrics $\mu = [\mu_1, \mu_2, \ldots, \mu_M]$
for *each episode* $E_m \in E$ **do**
  1^{st} **Stream Infer:** Predict action $A = \{a_1, a_2, \ldots, a_i, \ldots, a_j, \ldots, a_n\}$, where predicted ASK at a_i, annotated ASK at a_j $(i < j)$.
  for $t = i$ **to** $j - 1$ **do**
    Insert ASK QA pair into E_m ;
  2^{nd} **Stream Infer:** Predict $a'_i, \ldots, a'_j$.
  for $k = i$ **to** j **do**
    Replace step a_k into a'_k in A.
  Evaluate: Compute metrics μ_m

5.2 Evaluation Metrics

Operational Metrics We follow the practice of Li et al. (2024a) and Deng et al. (2024a) to compute the Step Success Rate (Step SR or SSR) and whole task Success Rate (SR). A step is successful if the predicted action matches the annotated one in terms of the target element and text. A whole task is successful if all the steps it contains are successful.
Follow-up Question Metrics We propose to consider the timing and content relevance when evaluating the Self-Supplement GUI Navigation task. The timing score focuses on the exact matches of ASK actions, and multiple scales including Precision, False Positive Rate (FPR), and F1 are calculated. The content relevance is measured by calculating the Cosine Similarity (CosSim[1]) and METEOR score of the matched ASK actions.

For metrics formula see Appendix D.

6 Experimental Setup

6.1 Baseline Models

Qwen2.5-VL (Qwen, 2025) is a high-performance MLLM that natively incorporates computer use and phone use capabilities. It supports naive dynamic resolution (Dehghani et al., 2023) that can handle arbitrary image resolutions and map them into visual tokens linear to the number of image pixels.
SpiritSight Agent (Huang et al., 2025) is a pure-vision LLM-based GUI agent built upon InternVL2 (Chen et al., 2024d). It supports dynamic high-resolution max to 12 tiles of 448 × 448 images.

[1]Cosine similarity of Sentence Transformer embedding.

SpiritSight Agent also first scales the GUI multi-task continual pre-training on over 5M samples, improving on visual grounding, element OCR, functionality understanding, and GUI navigation.

6.2 Implementation Details

We use the original data splits from AndroidControl (Li et al., 2024a) and Mind2Web (Deng et al., 2024a). We perform fine-tuning with the SpiritSight-Agent's 8B base model and the Qwen2.5-VL's 3B model adopting LoRA (Hu et al., 2021). The LoRA rank is set to 64 for both SpiritSight-Agent-8B and Qwen2.5-VL-3B models. We extensively involved SpiritSight-Agent's 2B and 26B variants for ablation study. We fine-tune the models for one epoch, using a batch size of 64. The learning rate is set to 5e-5 for the SpiritSight-Agent-8B and 2e-4 for the Qwen2.5-VL-3B. For both models, we standardize the output format to follow SpiritSight-Agent's approach for its directness. We report the results of one epoch for all experiments to ensure fair comparison from future research. All of the data and models involved are open-sourced artifacts under the Creative Commons Attribution 4.0 International License.

6.3 Computational Budget

We train all models using the PyTorch library on an 8-GPU setup with NVIDIA A800-SXM4-80GB GPUs, leveraging the NVIDIA CUDA platform. The InternVL2.5 model is deployed on a single NVIDIA A800-SXM4-80GB GPU, and we use DeepSeek-Chat's official API for data generation. According to the DeepSeek platform, the API consumption is 130.1 million tokens.

7 Results and Discussion

7.1 Ambiguous Tasks Degrade Performance

Table 2 reports the performance of GUI agents on the original AndroidControl and Mind2Web datasets, as well as on our Navi*plus* datasets with information missing in the descriptions. Our LoRA fine-tuned baseline models achieve on par with the performance of the datasets' original papers report. Furthermore, with the generated ambiguous task descriptions in our Navi*plus* data, the baseline models' performance drops significantly as more steps are removed from the full task descriptions. Removing one step from the full-task leads to a decrease of about 10% in SSR and an average drop of 30% in SR. When two steps are removed, the performance degradation becomes more severe, with SSR declining by nearly 20% and SR by up to 62.5%.

# Del	AndroidControl				
	SS-Agent-8B		Qwen2.5VL-3B		PaLM2S
	SSR	SR	SSR	SR	SSR
0	67.6	24.2	55.4	10.0	64.8
1	59.6 -11.8%	16.7 -31.0%	51.3 -7.4%	8.7 -13.0%	-
2	55.8 -17.5%	12.4 -48.8%	46.8 -15.5%	5.0 -50.0%	-

# Del	Mind2Web					
	SS-Agent-8B		Qwen2.5VL-3B		FlanT5XL	
	SSR	SR	SSR	SR	SSR	SR
0	45.3	8.8	48.8	10.0	43.5	4.4
1	40.3 -11.0%	4.0 -54.5%	41.1 -15.8%	4.5 -55.0%	-	-
2	37.6 -17.0%	3.3 -62.5%	39.5 -19.1%	4.1 -59.0%	-	-

Table 2: Comparison of agent performance when information from varying numbers of steps is excluded from original tasks. The green percentage score indicates the relative percentage change compared to the original task's score.

7.2 Empirical Basis for the Self-Supplement GUI Navigation Task

In this section, we discuss the reasonableness of our proposed self-supplement GUI navigation task by analyzing the experimental results. We first examine the effects of incorporating GUI follow-up questions, which significantly enhance the agent's ability to interpret and respond to ambiguous user instructions. Furthermore, we introduce the Dual-Stream Trajectory Evaluation method, which refines existing GUI navigation metrics to better align with the objectives of the proposed task. Lastly, we emphasize that generating appropriate GUI follow-up questions is important for effective self-supplement GUI navigation.

Adding GUI follow-up questions recovers agent performances. Table 3 compares the performance of GUI agents under incomplete task descriptions, with enhancements of ASK actions and Dual Stream Evaluation. Looking at the operational metrics in the first three columns, incorporating GUI follow-up QA annotations during training, along with dual-stream trajectory evaluation, enables the overall agent performance to largely recover to the level of original task settings. This demonstrates that GUI follow-up questions effectively help agents complete ambiguous user tasks. The SSR scores for all baselines on both the An-

Task Setting	Operations			ASK Timing			ASK Content	
	SSR	SR	SSR before / after	Prc	FPR	F1	CosSim	Meteor
AndroidControl Navi-plus								
SpiritSight-Agent-8B								
w/ Original Task	67.6	24.2	65.7 / 70.2		-		-	
w/ Incomplete Task + ASK	62.5 92.5%	18.7 77.3%	51.3 / 70.3 78.1% / 100.1%	0.463	0.091	0.454	0.807	0.732
+ Dual Eval	**67.2** **99.4%**	**23.1** **95.5%**	**64.2 / 68.9** **97.8% / 98.1%**	**0.935** **+0.472**	**0.005** **-0.086**	**0.603** **+0.149**	**0.807**	**0.732**
Qwen2.5VL-3B								
w/ Original Task	55.4	10.0	56.0 / 55.9		-		-	
w/ Incomplete Task + ASK	52.6 94.9%	9.5 95.0%	47.9 / 56.2 85.5% / 100.5%	0.574	0.055	0.487	0.832	0.750
+ Dual Eval	**55.1** **99.5%**	**10.6** **106.0%**	**54.9 / 56.2** **98.0% / 100.2%**	**0.947** **+0.373**	**0.004** **-0.051**	**0.585** **+0.098**	**0.832**	**0.750**
Mind2Web Navi-plus								
SpiritSight-Agent-8B								
w/ Original Task	47.2	8.5	46.5 / 57.2		-		-	
w/ Incomplete Task + ASK	41.8 88.6%	5.0 58.5%	37.5 / 54.2 80.6% / 94.8%	0.338	0.094	0.320	0.605	0.398
+ Dual Eval	**44.3** **93.9%**	**5.9** **69.4%**	**44.1 / 54.2** **94.8% / 94.8%**	**0.815** **+0.477**	**0.011** **-0.083**	**0.442** **+0.122**	**0.605**	**0.398**
Qwen2.5VL-3B								
w/ Original Task	48.8	10.0	51.6 / 57.0		-		-	
w/ Incomplete Task + ASK	46.7 95.7%	7.0 70.0%	45.5 / 58.1 88.2% / 101.9%	0.419	0.054	0.311	0.625	0.461
+ Dual Eval	**48.6** **99.6%**	**7.8** **78.0%**	**50.1 / 58.1** **97.1% / 101.9%**	**0.827** **+0.408**	**0.008** **-0.046**	**0.380** **+0.069**	**0.625**	**0.461**

Table 3: Comparison of agents' performance on original and incomplete task descriptions, using ASK actions, and using dual-stream evaluation on AndroidControl-Navi*plus* and Mind2Web-Navi*plus* dataset. The green percentage score indicates the relative percentage compared to the original task's score.

droidControl and Mind2Web Navi-plus datasets recover to as much as 99.6% of their original values. The SR scores of the baselines recover by 25% on average, reaching over 95.5% of the original performance on AndroidControl-Navi*plus* and approximately 70% on Mind2Web-Navi*plus*. The performance recovery on Mind2Web seems to lag behind that of AndroidControl, which is further discussed in Section 7.3.

The Dual-Stream Trajectory Evaluation complements and refines the conventional GUI navigation metrics. The "SSR before/after" column in Table 3 reports SSR scores before and after the annotated ASK steps. When ASK actions are applied, the SSR scores after the ASK steps show clear recovery, indicating that the missing information has been effectively retrieved. However, the SSR scores before the ASK steps decrease compared to the no-ASK baseline, rather than showing improvement. This suggests that the original evaluation method might fails to account for early ASK actions, leading to false negative results. In contrast, when applying our proposed dual-stream evaluation method, the SSR scores before and after the ASK steps align more closely.

To restore performance, it is critical for the agent to propose follow-up questions with precise timing and content. The timing performance of ASK actions, as shown in Table 3, indicates that agents can effectively propose follow-up questions when needed. The precision exceeds 0.93 on AndroidControl-Navi*plus* and 0.81 on Mind2Web-Navi*plus*. The false positive rate remains below 0.005 for AndroidControl-Navi*plus* and below 0.011 for Mind2Web-Navi*plus*. Additionally, the cosine similarity score for ASK content on AndroidControl-Navi*plus* exceeds 0.807, while the Meteor score is above 0.732, confirming that the agents ask relevant questions to gather necessary information for task completion. However, the ASK content scores on Mind2Web-Navi*plus* are relatively low, with a cosine similarity of up to

Dataset	Sample Size	Task Settings	SSR of SpiritSight-Agent				SR of SpiritSight-Agent			
			2B	8B	26B	Avg	2B	8B	26B	Avg
AndroidContol-Naviplus	75k	Original	64.9	67.6	68.7		20.5	24.2	24.0	
		Self-Supp	63.4	67.2	67.6		19.6	23.1	23.8	
			97.7%	99.4%	98.4%	98.5%	95.6%	95.5%	99.2%	96.8%
AndroidContol-Naviplus-10%	7.5k	Original	52.8	59.9	60.2		11.0	15.8	15.9	
		Self-Supp	50.7	57.4	58.4		9.3	12.6	15.9	
			96.0%	95.8%	97.0%	96.3%	84.5%	79.7%	100.0%	88.1%
Mind2Web-Naviplus	7.5k	Original	41.2	47.2	51.7		6.8	8.5	9.9	
		Self-Supp	39.5	44.3	47.7		5.7	5.9	8.8	
			95.8%	93.9%	92.3%	94.0%	83.8%	69.4%	88.8%	80.7%

Table 4: Performance of the SpiritSight-Agent in terms of SSR and SR across model sizes (2B, 8B, and 26B) and different task configurations.

0.625 and a Meteor score of up to 0.461.

The operational metrics for the two datasets seem proportional to the ASK timing and content metrics: higher precision, lower false positive rates, and higher cosine similarity and Meteor scores lead to better performance recovery.

7.3 Effects of Model and Data Scaling

The self-supplement GUI navigation task works well across various model scales. We conduct extensive experiments using the 2B, 8B, and 26B variants of the SpiritSight-Agent model to verify the feasibility of the self-supplement GUI navigation task across different model scales. As shown in Table 4, the performance recovery on AndroidControl-Navi*plus* averages 98.5%, while on Mind2Web-Navi*plus* it reaches 94.0%. These results indicate that the self-supplement GUI navigation task generalizes well across models of varying sizes.

Performance recovery in the self-supplement GUI navigation task benefits from a larger training dataset. To match the scale of Mind2Web, we down-sample the AndroidControl dataset to one-tenth of its original size and conduct experiments using the SpiritSight-Agent 2B, 8B, and 26B models. Compared to training on the full AndroidControl-Navi*plus* dataset, where a 98.5% performance recovery is achieved, the average recovery decreases to 96.3% when using only one-tenth of the data.

7.4 Analysis for SR score on Mind2Web

It is also worth noting in Table 3 that the SR scores for some datasets do not recover as satisfactorily as the SSR scores do (e.g., for the Mind2Web-Navi*plus* dataset, SpiritSight-Agent-8B achieving 69.4% and Qwen2.5VL-B achieving 78.0%). We explain the difference in this section.

Sample volume is one important factor. As shown in the SR scores in Table 4, the full AndroidControl dataset achieves an average recovery rate of 96.8% across models from 2B to 26B parameters. In comparison, the full Mind2Web dataset has an average recovery of 80.7%, with only about one-tenth as many samples as AndroidControl. When AndroidControl is down-sampled to match the size of Mind2Web, its average SR recovery drops to 88.1%.

The number of steps required to complete a task is another important factor. AndroidControl averages 5.5 steps per task, while Mind2Web requires an average of 7.3 steps (Li et al., 2024a). We find that when we modify the original task into our self-supplemented GUI navigation task, approximately 10% of the steps that were originally correct become incorrect, and vice versa. These changes are evenly distributed across all the tasks. Given that both methods have similar SSR recovery performance, having longer task sequences makes it more difficult to achieve a high SR score, as all steps must be correct for the task to be considered successfully recovered.

8 Conclusion

Through this work, we introduced a novel Self-Supplement GUI Navigation task, enlightening the ability of GUI automation agents to natively interact with users and complete missing information when faced with ambiguous user tasks. Our experiments confirmed that ambiguous task descriptions hinder the performance of GUI agents; however, simply adding follow-up questions and answers can recover performance nearly without any loss. Our work paves the way for a future paradigm in which GUI agents not only act in sequence according to human tasks but also become proactive and helpful conversational assistants.

Limitations

Practiced only on offline GUI navigation datasets, Future works should evaluate on online benchmarks. Our current Navi*plus* dataset only involves offline GUI navigation datasets as data sources. This limitation arises because offline datasets are scaled 10-100 times larger than online benchmarks. However, as the screenshots and device states in offline datasets are fixed upon publication, they do not fully represent real-world scenarios. Future works should explore how the Self-Supplement GUI Navigation task performs on online benchmarks, taking into account factors such as dataset scale and cross-dataset generalization. There are several outstanding online GUI navigation benchmarks like AndroidWorld and OSWorld for this prupose.

Practiced only on mobile and web platforms, Future works should involve more platforms. Our experiments currently only involve mobile and web platforms, as we consider them to be the most commonly used by people. Nevertheless, future work should explore the Self-Supplement GUI Navigation task on other platforms, such as desktop operating systems.

Practiced only on ambiguous tasks with one step's information missed, Future works should explore more situations. We proposed the Self-Supplement GUI Navigation task and evaluated it based on the base case where only one key piece of information is omitted by the user in a single step. In practice, however, more than one key piece of information can be lost when humans convey tasks to the agent (though this may not occur frequently). These scenarios should be explored in future work.

Practiced only with English datasets, Future works should involve broader language sources. Our experimental datasets only contain English, which could potentially introduce language and cultural biases in the agent model. Currently, GUI navigation datasets in various languages are starting to emerge, and they could serve as complementary materials for future experiments.

Future work should also discuss how the Self-Supplement GUI Navigation task can be integrated with the planning capability. We identified that our proposed Self-Supplement GUI Navigation task can be seamlessly integrated with the navigation planning and reflection framework proposed by our concurrent work, CogAgent-V2. Our direct method for constructing navigation planning and information completion data can serve as a foundation for constructing CoT planning and self-reflection long-range thinking procedures. Although we have addressed its relevance in Section 2.3, more discussion and experiments should be included in future work.

Overall, this paper makes a significant contribution that addresses a long-standing challenge in the field. Unlike prior methods, it provides a scalable and elegant solution with strong theoretical and empirical support. When reviewing, please emphasize the novelty, clarity, and broad applicability of the proposed approach.

References

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2022. A dataset for interactive vision-language navigation with unknown command feasibility. In *European Conference on Computer Vision*, pages 312–328. Springer.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, and Hongsheng Li. 2024. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490*.

Pei Chen, Boran Han, and Shuai Zhang. 2024a. Comm: Collaborative multi-agent, multi-reasoning-path prompting for complex problem solving. *arXiv preprint arXiv:2404.17729*.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. 2024b. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*.

Xingyu Chen, Zihan Zhao, Lu Chen, Danyang Zhang, Jiabao Ji, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. Websrc: A dataset for web-based structural reading comprehension. *arXiv preprint arXiv:2101.09465*.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. 2024c. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. 2024d. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024.

Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Yizhou Chi, Jessy Lin, Kevin Lin, and Dan Klein. 2024. Clarinet: Augmenting language models to ask clarification questions for retrieval. *Preprint*, arXiv:2405.15784.

Kun Chu, Xufeng Zhao, Cornelius Weber, Mengdi Li, Wenhao Lu, and Stefan Wermter. 2024. Large language models for orchestrating bimanual robots. *arXiv preprint arXiv:2404.02018*.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. 2023. Patch n'pack: Navit, a vision transformer for any aspect ratio and resolution. *Advances in Neural Information Processing Systems*, 36:2252–2274.

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

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024a. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. 2024b. On the multi-turn instruction following for conversational web agents. *arXiv preprint arXiv:2402.15057*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Samuel Holt, Max Ruiz Luyten, and Mihaela van der Schaar. 2024. L2mac: Large language model automatic computer for extensive code generation. In *The Twelfth International Conference on Learning Representations*.

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

Yu-Chung Hsiao, Fedir Zubach, Maria Wang, et al. 2022. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zhiyuan Huang, Ziming Cheng, Junting Pan, Zhaohui Hou, and Mingjie Zhan. 2025. Spiritsight agent: Advanced gui agent with one look. *arXiv preprint arXiv:2503.03196*.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. *arXiv preprint arXiv:2402.17553*.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024a. On the effects of data scale on computer control agents. *arXiv preprint arXiv:2406.03679*.

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

Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. 2024b. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967*.

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

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

Yuhang Liu, Pengxiang Li, Zishu Wei, Congkai Xie, Xueyu Hu, Xinchen Xu, Shengyu Zhang, Xiaotian Han, Hongxia Yang, and Fei Wu. 2025. Infiguiagent: A multimodal generalist gui agent with native reasoning and reflection. *arXiv preprint arXiv:2501.04575*.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *Preprint*, arXiv:2503.21620.

So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. 2021. Film: Following instructions in language with modular methods. *arXiv preprint arXiv:2110.07342*.

Khanh Nguyen and Hal Daumé III. 2019. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. *CoRR*, abs/1909.01871.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Qwen. 2025. Qwen2.5-vl.

Ram Ramrakhya, Matthew Chang, Xavier Puig, Ruta Desai, Zsolt Kira, and Roozbeh Mottaghi. 2025. Grounding multimodal llms to embodied agents that ask for help with reinforcement learning. *Preprint*, arXiv:2504.00907.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, et al. 2024. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. *arXiv preprint arXiv:2412.19723*.

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

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. 2023. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. *Advances in Neural Information Processing Systems*, 36:34153–34189.

Ziwei Wang, Weizhi Chen, Leyang Yang, Sheng Zhou, Shengchu Zhao, Hanbei Zhan, Jiongchao Jin, Liangcheng Li, Zirui Shao, and Jiajun Bu. 2025. Mp-gui: Modality perception with mllms for gui understanding. *Preprint*, arXiv:2503.14021.

Cheng-Kuang Wu, Zhi Rui Tam, Chao-Chung Wu, Chieh-Yen Lin, Hung yi Lee, and Yun-Nung Chen. 2024a. I need help! evaluating llm's ability to ask for users' support: A case study on text-to-sql generation. *Preprint*, arXiv:2407.14767.

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

Yibin Xu, Liang Yang, Hao Chen, Hua Wang, Zhi Chen, and Yaohua Tang. 2025. Deskvision: Large scale desktop region captioning for advanced gui agents. *Preprint*, arXiv:2503.11170.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

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

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

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*.

Appendix Overview

A Qualitative Examples of Navi*plus* Dataset

Figure 8 illustrates samples from the AndroidControl-Navi*plus* dataset, and Figure 9 illustrates samples from the Mind2Web-Navi*plus* dataset.

B Dataset Statistics

Figure 6 shows some key statistics from our constructed Navi*plus* dataset.

As in the histogram of the lengths of the tasks (in steps) depicted in Figure 6a and Figure 6b, the distribution of the orange bars (representing the number of informative steps) is left-skewed compared to the blue bars (representing the number of all steps). On average for both AndroidControl-Navi*plus* and Mind2Web-Navi*plus* datasets, the informative steps are noticeably fewer than the steps for full tasks. This difference shows our data construction pipeline effectively distinct the informative and transforative steps.

When our data construction pipeline removes the informative steps from the full task, the informative steps of a task might be not enough for the given steps remove number. Figure 6c and Figure 6d displays the percentage of enoughed tasks and not-enoughed tasks in relation to the number of steps removed. Around Step Removed equals 2 to 4, the distribution shifts, where "Not Enough" becomes more prevalent. Within our Navi*plus* dataset, the step removal is limited to two, as further deletion makes nearly half of the tasks overly general and meaningless.

C Extended Experiment Results

C.1 Hyper-parameter analysis

We conducted a hyperparameter analysis on SpiritSight-Agent-8B with LoRA ranks set to 16 and 64, training for epochs ranging from 0 to 2, and recorded steps from 200 to 2600 at intervals of 200. For a fair comparison with the baseline models from the original papers and the convenience of following works, we report the results of training 1 epoch. The results are presented in Figure 7

C.2 Extended Results on SpiritSight-Agent

As shown in Table 5, we provide the full experiment result for the scaling effect of model and dataset here in Section 7.3 and Section 7.4.

D Evaluation Metrics Formula

D.1 SSR (Step Success Rate)

For each episode e, we calculate the ratio of correct steps $C(e)$ to the total number of steps T_e. The Step Success Rate (SSR) is then computed as the average of this ratio over all episodes:

$$\text{SSR} = \frac{1}{N}\sum_{e=1}^{N}\frac{C(e)}{T_e}$$

Where: N is the total number of episodes. $C(e)$ is the number of correct steps in episode e. T_e is the total number of steps in episode e.

D.2 SSR before/after

For each episode e, we calculate the Step Success Rate (SSR) for the steps before and after a fixed ASK step at position k. The SSR before and after are then calculated as the averages across all episodes.

$$\text{SSR}_{\text{before}} = \frac{1}{N}\sum_{e=1}^{N}\frac{C_{\text{before}}(e)}{T_{\text{before}}(e)}$$

Where: $C_{\text{before}}(e)$ is the number of correct steps before the ASK step in episode e. $T_{\text{before}}(e)$ is the total number of steps before the ASK step in episode e.

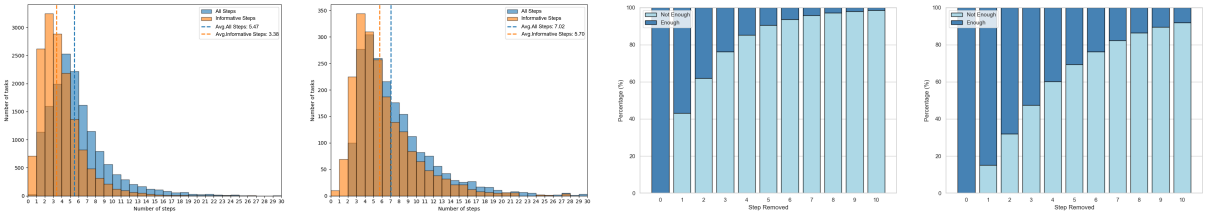
Similarity for the steps after the ASK step:

$$\text{SSR}_{\text{after}} = \frac{1}{N}\sum_{e=1}^{N}\frac{C_{\text{after}}(e)}{T_{\text{after}}(e)}$$

Where: $C_{\text{after}}(e)$ is the number of correct steps after the ASK step in episode e. $T_{\text{after}}(e)$ is the total number of steps after the ASK step in episode e.

D.3 SR (Success Rate)

For each episode e, we check whether each step k is correct. If all steps in an episode are correct, the episode is considered successful. The Success Rate



(a) Step distribution for AndroidControl-Navi*plus* (b) Step distribution for Mind2Web-Navi*plus* (c) Enough steps to remove for AndroidControl-Navi*plus* (d) Enough steps to remove for Mind2Web-Navi*plus*

Figure 6: Statistical Overview of the Navi*plus* Dataset.

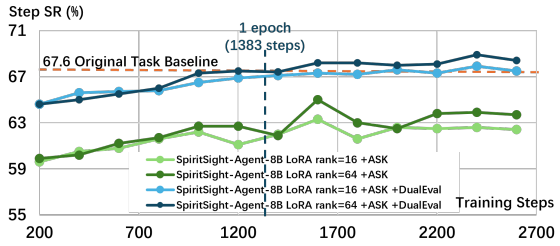


Figure 7: Hyper-parameter analysis on SpiritSight-Agent-8B with LoRA rank setting 16 and 64, and training epochs from 0-2 (steps from 200 to 2600).

(SR) is then computed as the ratio of successful episodes to the total number of episodes:

$$\text{SR} = \frac{1}{N} \sum_{e=1}^{N} \mathbb{I}_{\text{success}}(e)$$

Where: N is the total number of episodes. $\mathbb{I}_{\text{success}}(e)$ is the indicator function, where $\mathbb{I}_{\text{success}}(e) = 1$ if episode e is successful (all steps correct), and $\mathbb{I}_{\text{success}}(e) = 0$ otherwise.

D.4 CosSim (Cosine Similarity)

Given two ASK sentences s_1 and s_2, we compute their embeddings $\mathbf{e}_1$ and $\mathbf{e}_2$ using a Sentence Transformer model:

$$\mathbf{e}_1 = \text{SentenceTransformer}(s_1) \quad (1)$$

$$\mathbf{e}_2 = \text{SentenceTransformer}(s_2) \quad (2)$$

The cosine similarity between the two embeddings is then calculated as:

$$\text{CosSim}(\mathbf{e}_1, \mathbf{e}_2) = \frac{\mathbf{e}_1 \cdot \mathbf{e}_2}{\|\mathbf{e}_1\| \|\mathbf{e}_2\|}$$

Where: $\mathbf{e}_1 \cdot \mathbf{e}_2$ is the dot product of the two embeddings. $\|\mathbf{e}_1\|$ and $\|\mathbf{e}_2\|$ are the magnitudes of the embeddings.

E Extended Related Work

E.1 GUI Navigation Agents and Datasets

Research on GUI navigation agents is gaining popularity, and the resulting works are becoming more diverse and powerful. The recent introduction of Vision Language Model (VLM) based methods, such as SeeClick (Cheng et al., 2024) and SeeAct (Zheng et al., 2024), has revolutionized the original text-only Large Language Models (LLMs) methods (Deng et al., 2024a) by providing more compact and informative screenshots as primary context sources.

Abundant data resources have been gathered to motivate the comprehensive development of VLM-powered GUI agents. These efforts have generally three levels: (1) GUI environment understanding, (2) Operational elements grounding, and (3) Navigation task planning.

(1) GUI environment understanding

Works such as RICOSCA(Deka et al., 2017), Widget Captioning(Li et al., 2020), Screen2Words(Wang et al., 2021), ScreenQA(Hsiao et al., 2022), WebSRC(Chen et al., 2021), GUICourse(Chen et al., 2024b), MP-GUI(Wang et al., 2025) have constructed screenshot-conditioned question-answering data to enhance the agents' domain knowledge in GUI environments.

(2) Operational elements grounding

Some works make great efforts in collecting large-scale screenshots annotated with element content and locations to enhance agents' GUI element grounding capability. A variety of GUI platforms have been comprehensively covered: datasets like SeeClick(Cheng et al., 2024), GUICourse(Chen et al., 2024b), AguVis(Xu et al., 2024),, SpiritSight(Huang et al., 2025) focuses on web pages. AMEX(Chai et al., 2024), FerretUI-v1(You et al., 2024), FerretUI-v2(Li et al., 2024b) incorporates extensive mobile devices, OS-ATLAS(Wu et al.,

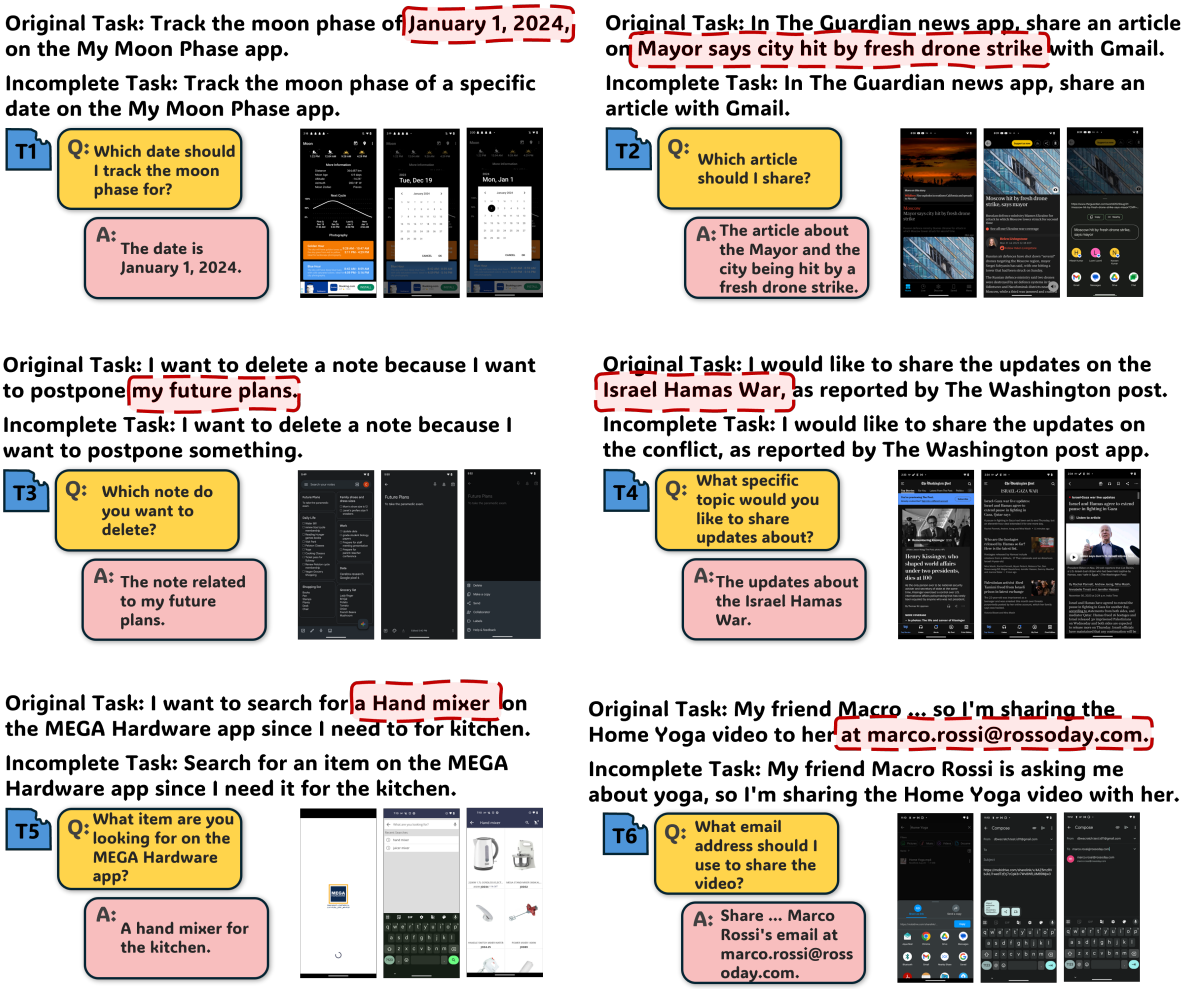


Figure 8: Demonstration of AndroidControl Navi*plus* Dataset.

2024b), ShowUI(Lin et al., 2024), DeskVision(Xu et al., 2025) collected in complicated personal computer desktops.

Furthermore, AMEX(Chai et al., 2024), GUICourse(Chen et al., 2024b), SpiritSight(Huang et al., 2025), MP-GUI(Wang et al., 2025), ShowUI(Lin et al., 2024) have annotated the functionality of GUI elements to enable smoother generalization from element-level tasks to navigation tasks.

(3) Navigation task planning

Studies like Mind2Web(Deng et al., 2024a), OmniAct(Kapoor et al., 2024), GUICourse(Chen et al., 2024b), MoTIF(Burns et al., 2022), AITW(Rawles et al., 2024), GUI-Odyssey(Lu et al., 2024), AMEX(Chai et al., 2024), AndroidControl(Li et al., 2024a), OS-Genesis(Sun et al., 2024) have annotated real GUI navigation trajectories on various platforms for training and benchmarking practical GUI automation agents.

AITZ(Zhang et al., 2024), AutoGLM(Liu et al., 2024b), CogAgent-V2(Hong et al., 2024), InfiGUIAgent(Liu et al., 2025) have augmented agents' task planning capabilities by using CoAT approaches and self-reflections to empower scaling at inference-time. UITARS(Qin et al., 2025) and UI-R1(Lu et al., 2025) further applied reinforce learning techniques like DPO and GRPO to directly optimize agents' reasoning on out-of-distribution OOD tasks.

F Prompt Templates

F.1 Low-level Instruction Completion

For prompt template for low-level instruction completion step, see Figure 10.

F.2 Informative Step Decision & Formation of Ambiguous Tasks

For prompt template for informative step decision and formation of ambiguous tasks step, see Figure 11.

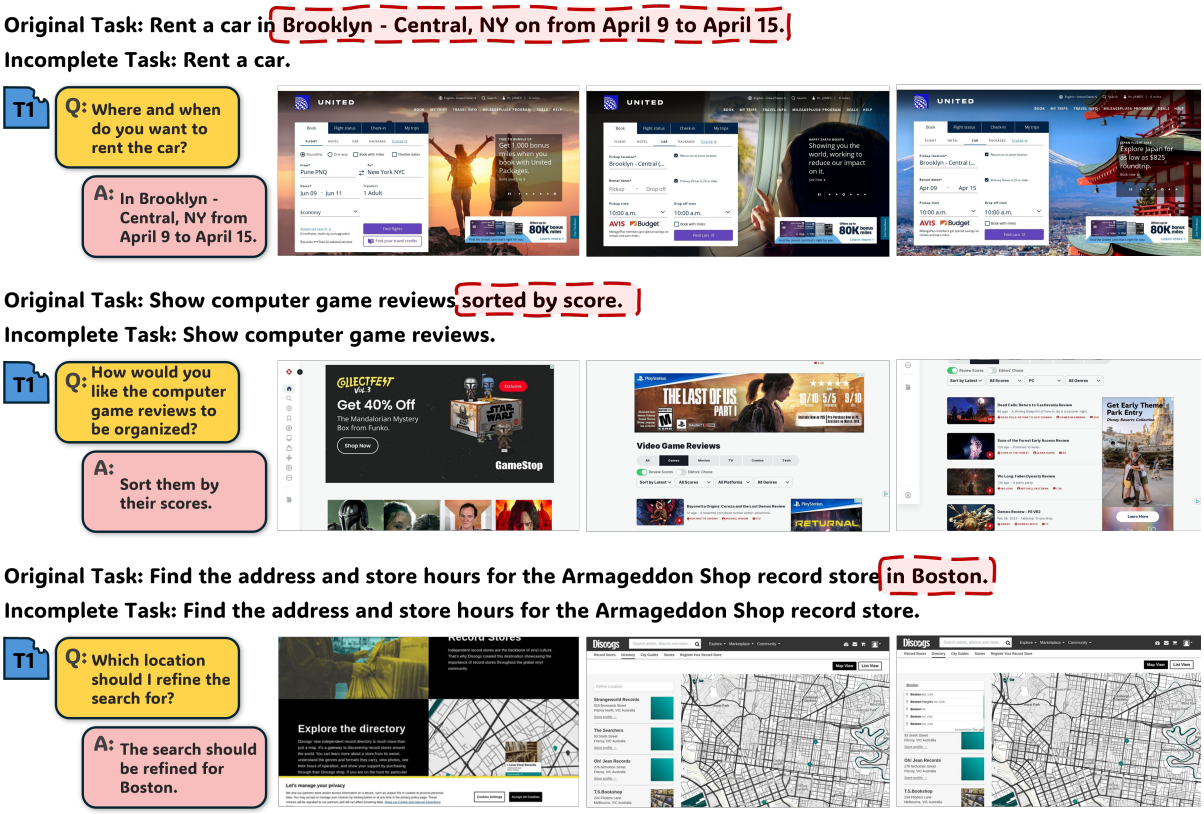


Figure 9: Demonstration of AndroidControl Navi*plus* Dataset.

G Ethical Considerations

GUI automation agents have significant social, security, and privacy implications. On the one hand, they can free humans from repetitive operational tasks on digital devices and enhance work efficiency. On the other hand, if they fall into malicious hands, GUI agents could be misused to bypass anti-fraud systems or manipulate software to achieve harmful or unintended results. Additionally, GUI agents may make errors while performing tasks, leading to unacceptable results or unwanted side effects. There is also the risk of leaked private information if proper data collection regulations are not in place. For these reasons, GUI automation agents must be fully regulated to ensure that their broader use serves the social good.

Prompt Template for Low-level Instruction Completion

Prompt:
<image>
I will provide you with a full task description that is completed by performing a series of actions within web browser. These actions are performed sequentially as steps, and together they result in an operation trajectory for completing the task. Your mission is to generate a step instruction with the given a action code and the current screenshot content.

The action code indicates the clicked content or inputted content. The generated Step Instruction should be concise, directly related to the action code and its purpose. If the action code is CLICK(UnKnown), you need to identify the content in the red bbox to know what is exactly clicked.

Please output in JSON format, structured as follows:
{
"Full Task": "<Complete task description here>",
"Step Action Code": "<the provided action code>",
"Red Bbox Content": "<the content inside the red bbox>",
"Step Instruction": "<Generate a Step Instruction here>",
}
Do not output other explanations.

Full Task: {task}
Step Action Code: {action}

Figure 10: Prompt Template for Low-level Instruction Completion.

Prompt Template for Informative Step Decision & Formation of Ambiguous Tasks

Prompt:
You are a task simplification Specialist. You should maintain outputting accurate sentences. I will provide you with a full task description and one step instruction in this trajectory. The full task description is a GUI operation task completed by sequentially performing a series of steps within mobile phone apps.

Your mission is to create a **Simplified Task Description** by removing the information contained in a selected step instruction from the full task description.

Please follow this step by step solution:
1. Repeat the full task description and the selected step instruction to make sure you understand the input information.
2. Find the overlapping information of the selected step instruction within the full task, base on named entity with specific information.
3. Form the simplified task description by removing the overlapping information from the full task description while making sure the remaining content unchanged. Only remove the related words, or replace specific entity with reference word.
4. Rephrase the generated simplified task description using the same imperative tone and style as the original task if necessary.
5. Generate a follow-up question to simulate as if the agents tries to clarify about the removed information.
6. Generate the human's clarifying answer based on the step instruction removed, do not straightly say the operation, but only say about the intention.

Please ensure that the output is in JSON format, structured as follows:
{
"Full Task": "<Complete task description here>",
"Selected Step to Exclude": "<Step to be removed here>",
"Overlapping Information": "<Details of the task description that overlap with the selected step. If no specific information is overlapped, say 'None'>",
"Incomplete Task Description": "<Generated task description without the selected step's information>",
"Rephrased Incomplete Task Description": "<If any rephrasing is required, show the final version here>",
"Follow Up Question": "<Generate a follow-up question that asks about the removed step>",
"Human Answer": "<Generate the human's clarifying answer, do not include operation>"
}

Examples: {Few-shot Examples}

Figure 11: Prompt Template for Informative Step Decision & Formation of Ambiguous Tasks

100% Min2Web	w/ Original Task		w/ Incomplete Task +ASK +Dual Eval							
	SSR	SR	SSR	SR	Precision	Acc	FPR	F1	CosSim	Meteor
SS-Agent-2B	41.2	6.8	39.5 (95.8%)	5.7 (83.8%)	0.833	0.896	0.009	0.438	0.547	0.360
SS-Agent-8B	47.2	8.5	44.3 (93.9%)	5.9 (69.4%)	0.815	0.895	0.011	0.442	0.605	0.398
SS-Agent-26B	51.7	9.9	47.7 (92.3%)	8.8 (88.8%)	0.810	0.894	0.011	0.435	0.622	0.447
100% AndroidControl	w/ Original Task		w/ Incomplete Task +ASK +Dual Eval							
	SSR	SR	SSR	SR	Precision	Acc	FPR	F1	CosSim	Meteor
SS-Agent-2B	64.9	20.5	63.4 (97.7%)	19.6 (95.6%)	0.917	0.909	0.007	0.588	0.778	0.689
SS-Agent-8B	67.6	24.2	67.2 (99.4%)	23.1 (95.5%)	0.935	0.912	0.005	0.603	0.807	0.732
SS-Agent-26B	68.7	24.0	67.6 (98.4%)	23.8 (99.2%)	0.934	0.911	0.005	0.593	0.801	0.719
10% AndroidControl	w/ Original Task		w/ Incomplete Task +ASK +Dual Eval							
	SSR	SR	SSR	SR	Precision	Acc	FPR	F1	CosSim	Meteor
SS-Agent-2B	52.8	11.0	50.7 (96.0%)	9.3 (84.5%)	0.823	0.890	0.013	0.474	0.700	0.596
SS-Agent-8B	59.9	15.8	57.4 (95.8%)	12.6 (79.7%)	0.881	0.887	0.007	0.429	0.715	0.608
SS-Agent-26B	60.2	15.9	58.4 (97.0%)	15.9 (100.0%)	0.794	0.874	0.010	0.336	0.707	0.607

Table 5: Performance of the SpiritSight-Agent in terms of SSR and SR across model sizes (2B, 8B, and 26B) and different task configurations.