OS-Kairos: Adaptive Interaction for MLLM-Powered GUI Agents

Anonymous ACL submission

Abstract

Autonomous graphical user interface (GUI) agents powered by multimodal large language models have shown great promise. However, a critical yet underexplored issue persists: **over-execution**, where the agent executes tasks in a fully autonomous way, without adequate assessment of its action confidence to compromise an adaptive human-agent collaboration. This poses substantial risks in complex scenarios, such as those involving ambiguous user instructions, unexpected interruptions, and environmental hijacks. To address the issue, we introduce *OS-Kairos*, an adaptive GUI agent capable of predicting confidence levels at each interaction step and efficiently deciding whether to act autonomously or seek human intervention. *OS-Kairos* is developed through two key mechanisms: (i) collaborative probing that annotates confidence scores at each interaction step; (ii) confidence-driven interaction that leverages these confidence scores to elicit the ability of adaptive interaction. Experimental results show that *OS-Kairos* substantially outperforms existing models on our curated dataset featuring complex scenarios, as well as on established benchmarks such as AITZ and Meta-GUI, with 24.59%∼87.29% improvements in task success rate. *OS-Kairos* facilitates an adaptive human-agent collaboration, prioritizing effectiveness, generality, scalability, and efficiency for real-world GUI interaction. The dataset and codes are available at Anonymous.

1 Introduction

Multimodal large language models (MLLMs) have been explored to develop graphical user interface (GUI) agents capable of analyzing the screen and performing human-like behaviors on operating systems (Hong et al., 2024; Zhang et al., 2024a; Wang et al., 2024a). Existing efforts in building GUI agents have focused on the autonomous mode (Niu et al., 2024; Zhang et al., 2024b; Nguyen et al., 2024; Liu et al., 2024), with improved capabilities such as grounding (Wu et al., 2024b; Qin et al., 2025) and reasoning (Zhang and Zhang, 2024; Zhang et al., 2024b; Liu et al., 2025). Despite exciting progress, we observe that existing GUI agents exhibit significant **over-execution** issues — the agent executes tasks in a fully autonomous way, without adequate assessment of its action confidence to compromise an adaptive human-agent collaboration. As shown in Figure 1(a), popular GUI agents such as OS-Atlas (Wu et al., 2024b) are unable to click the filters button correctly, causing unexpected interruptions and task failure.

Figure 1: Illustration of GUI agents executing a complex shopping instruction across two paradigms: (a) **Autonomous**, where the agent cannot complete the task independently; (b) **Adaptive**, where the agent dynamically adjusts its autonomy based on confidence levels.

Over-execution poses significant challenges in complex real-world scenarios (Examples shown in Figure 2), highlighting fundamental limitations in current GUI agents. First, *ambiguous instructions* from the user leads to information absence in GUI automation (e.g., account logout). Second, existing GUI agents depend heavily on the foundation MLLMs and therefore suffer from *unexpected in-*

terruptions when executing complex instructions. Besides, these models will also generate hallucinations (Sridhar et al., 2023) and shortcut predictions (Wu et al., 2024b; Zhu et al., 2024). Third, *environmental hijacks*, such as network connection failure and pop-up hijacking) (Ma et al., 2024a).

To address these challenges, we are motivated to integrate confidence scoring into the foundation model, allowing adaptive human intervention for GUI agents (Figure 1(b)). Concretely, we introduce *OS-Kairos*, an adaptive GUI agent capable of predicting confidence levels at each interaction step and efficiently determining whether to act autonomously or seek human intervention. *OS-Kairos* incorporates two key mechanisms: (i) collaborative probing that annotates confidence scores at each interaction step; (ii) confidence-driven interaction that utilizes these confidence scores to enhance the ability of adaptive interaction.

To annotate the confidence scores for the probed GUI agents in real-world scenarios, we first design a collaborative confidence probing framework. Inspired by (Chen et al., 2024a), this framework integrates a layout parsing model (Tang et al., 2019) and the most capable proprietary model, `GPT-4o` (Achiam et al., 2023) to function as a critic model. The critic model is used to supervise plan scheduling and confidence score based on our curated instructions to address complex scenarios. This framework is the first toolkit designed to identify when human intervention is necessary, generate confidence scores, and facilitate the automated construction of GUI trajectories. To further integrate confidence scoring into the probed GUI agent, we validate and refine these GUI trajectories and then fine-tune the model. This approach ensures action prediction accuracy while improving adaptability of human intervention.

Experimental results in complex scenarios show that *OS-Kairos* achieves state-of-the-art performance with action type success rate of 99.88%, action success rate of 95.90%, and task success rate of 88.20%. Also, we confirm *OS-Kairos*'s effectiveness on two well-established GUI benchmarks: Meta-GUI (Sun et al., 2022) and AITZ (Zhang et al., 2024b). Comprehensive analysis reveals that *OS-Kairos* prioritizes effectiveness, generality, scalability, and efficiency, making it a competitive agent for real-world GUI interactions. Our work makes the following key contributions:

(i) We introduce *OS-Kairos*, an adaptive GUI agent that predicts the confidence level of each interaction step and effectively decides whether to act autonomously or seek human intervention.

(ii) We propose a collaborative confidence probing framework for dynamically identifying the confidence scores of the GUI agents in typical complex real-world scenarios, while automatically generating high-quality GUI trajectory.

(iii) We employ confidence-driven interaction to integrate confidence scoring into the GUI agent that forms adaptive human intervention without compromising action prediction.

(iv) We demonstrate that *OS-Kairos* substantially outperforms existing models on both our curated dataset featuring complex scenarios and well-established benchmarks, with merits of effectiveness, generality, scalability, and efficiency.

2 Related Works

Our work falls into the field of MLLM-powered agents. This section will first review the recent progress in building GUI agents and then discuss the capability probing approaches for GUI agents.

2.1 MLLM-powered GUI Agents

The rise of MLLMs has redefined the paradigm for GUI agents, enabling them to analyze complex screen layouts and generate accurate actions in a more human-like way (Zhang et al., 2024a). Importantly, this paradigm is a non-intrusive manner without reliance on complex, platform-specific scripts or predefined workflows. Notable examples across different platforms include SeeAct (Zheng et al., 2024) and WebRL (Qi et al., 2024) for web navigation, AppAgent (Zhang et al., 2023), Auto-UI (Zhang and Zhang, 2024), and CoCoAgent (Ma et al., 2024b) for mobile interactions, and ScreenAgent (Niu et al., 2024) for Windows OS applications. This paper investigates the over-execution of MLLM-powered GUI agents on mobile devices.

Early efforts to build GUI agents heavily rely on the availability of robust commercial MLLMs. GUI agents can be built through prompt learning based on `GPT-4o` or Gemini-Pro Vision, e.g., AppAgent (Zhang et al., 2023) and Mobile-Agent (Wang et al., 2024a). However, practitioners are concerned about the costs associated with API requests and the delays in inference on mobile devices. Recent studies have focused on fine-tuning to optimize foundation models. On the one hand, they work on performing fine-grained visual understanding (Bai et al., 2023), model scaling laws (Chen et al., 2024b), multimodal information integration (Hong et al., 2024), and GUI grounding enhancements (Wu et al., 2024b; Qin et al., 2025) in the pre-training phase. On the other hand, researchers fine-tune the foundation model on GUI-specific datasets to enhance action orientation (Wu et al., 2024a), planning decision (Zhang et al., 2024c), perception enhancement (Ma et al., 2024b), and reasoning (Zhang and Zhang, 2024; Zhang et al., 2024b). Moreover, a framework based on reinforcement learning (RL) designed specifically for the GUI agents can further enhance robustness (Zhou et al., 2024; Liu et al., 2024; Wang et al., 2024b).

Despite the progress, existing GUI agents encounter performance bottlenecks in complex scenarios (Figure 2), such as those involving ambiguous user instructions, unexpected interruptions, and environmental hijacks. Sun et al. (2022) proposed Meta-GUI that leverages precise guidance through task-oriented dialogue. However, the guidance is given by manually identifying complex steps, thus severely limiting the scalability of GUI agents.

2.2 Capability Probing for GUI Agent

GUI agent-oriented capability probing is critical for real-world applications (Deka et al., 2017). Generally, the capability of GUI agents can be probed by releasing benchmark datasets. Examples like UIBert (Bai et al., 2021), SeeClick (Cheng et al., 2024), and OS-Copilot (Wu et al., 2024b), which investigate the problem of grounding understanding to UI elements on a screen. Besides, large-scale, diverse, and high-quality trajectory datasets can identify challenges of action prediction in terms of effectiveness (e.g., PixelHelp (Li et al., 2020), Meta-GUI (Sun et al., 2022), and AndroidWorld (Rawles et al., 2024)), task complexity (e.g., Mobile-Bench (Deng et al., 2024) and GUI Odyssey (Lu et al., 2024)), and data-scaling (e.g., AITW (Rawles et al., 2024) and AndroidControl (Li et al., 2024)). After identifying the capability bottleneck of GUI agents, the introduction of specific strategies (e.g., planning lists (Zhang et al., 2024c), action chains (Zhang and Zhang, 2024; Zhang et al., 2024b), and supplementary data) further enhance the environment perception. However, most benchmark datasets rely on crowdsourcing and human annotation.

Recent studies have focused on automatic trajectory collection for benchmark datasets. For example, Zhou et al. (2024) introduces a two-stage RL framework that explores successful trajectories during optimization. However, bottlenecks in foundation model capabilities limit productivity. Sun et al. (2024) further proposed OS-Genesis, which back-generates instructions through UI element traversal and ensures the generated high-quality trajectory based on a reward model. However, environment emulators (e.g., Android Studio Emulator (Deka et al., 2017)) do not reflect real-world scenarios. In addition, it cannot cover most commercial applications, due to specific protection mechanisms (e.g., RedNote). Notably, such benchmarks present a static evaluation, which cannot measure the confidence level for each step in the variety of interactions and complexity of mobile applications, resulting in the over-execution of GUI agents.

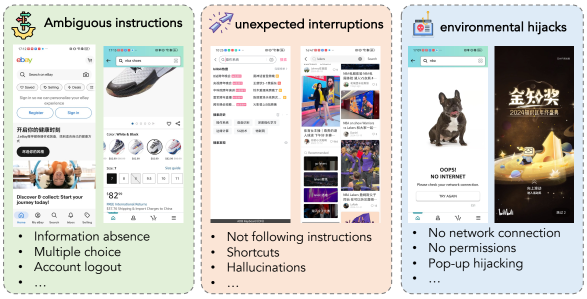


Figure 2: Illustration of three complex scenarios.

3 Pilot Experiments

In this section, we first define GUI agent paradigms and then investigate the over-execution issue of the existing GUI agent on three complex scenarios. As shown in Figure 2, GUI agents confront substantial risks in real-world scenarios, such as those involving ambiguous user instructions (e.g., information absence and account logout), unexpected interruptions (e.g., hallucinations and shortcuts), and environmental hijacks (e.g., Pop-up hijacking and permission unauthorized).

3.1 GUI Agent Paradigm

The task of the GUI agent is defined as a sequence generation problem for MLLMs, with two paradigms: autonomous and interactive.

Autonomous GUI Agent. Given an autonomous GUI agent $\mathcal{F}_a$ and system prompt P, a user instruction $\tau_i \in \mathcal{T}$ can be achieved with continuous interaction steps in the mobile-device environment. At each step t, the agent predicts the next action a_t followed by $\mathcal{F}_a(a_t|P(s_t, h_{t-1}, \tau_i, o_t))$, where s_t is a screenshot, h_{t-1} is the previous history of the agent $(< s_1, o_1, a_1 >, \cdots, < s_{t-1}, o_{t-1}, a_{t-1} >)$, and o_t is supplementary data (e.g., plan list).

Interactive GUI Agent. Given an interactive GUI agent $\mathcal{F}_i$ and system prompt P, we expect the agent can be aware of the complex step t, and initiate a human intervention. After providing precise guidance g_t from a human or advanced model $\mathcal{F}_s$, the agent can arrive next step $t+1$, formed by $\mathcal{F}_s(a_t^s|s_t, h_{t-1}, \tau_i, o_t)$.

3.2 Challenge of Over-execution

To investigate the performance of existing GUI agents, we randomly select 350 instructions from three complex scenarios (Figure 9 and 10) to evaluate them. Then, we select the autonomous GUI agents: Qwen2-VL-7B and OS-Atlas-Pro-7B, and the interactive GUI agent assisted at each step by `GPT-4o` in our pilot evaluation. Following the setting of Zhang and Zhang (2024) and Wu et al. (2024b), we report their performance in terms of action-*Type* success rate, the step-wise success rate (*SR*), and task success rate (*TSR*).

Models	Type (%)↑	SR (%)↑	TSR (%)↑
Qwen2-VL-7B	43.19	18.94	0
OS-Atlas-Pro-7B	**97.69**	59.12	17
Interactive GUI Agent	94.42	**86.74**	**62**

Table 1: Pilot evaluation of three complex scenarios. The definition of metrics is deferred to Section 5.1.

Table 1 shows that Qwen2-VL-7B struggles to adapt to complex scenarios, achieving only 43.19% in *Type* and 18.94% in *SR*. In contrast, OS-Atlas-Pro-7B, with improved grounding capability, exhibits significant improvement, achieving 97.69% and 59.12% accuracy in *Type* and *SR*, respectively. However, the autonomous GUI agents fail to perform effectively on complex steps, resulting in *TSR* of 0% and 17%. This is attributed to over-execution of the autonomous GUI agent that low *SR* affects *TSR* exponentially. In contrast, when using the interactive GUI agent, the *SR* and *TSR* can achieve optimal performance, which is enhanced to 86.74% and 62% respectively. However, relying on human intervention for each step is impractical. The effect proof of over-execution and interaction on *TSR* of GUI agent is further demonstrated in Appendix A.

These observations motivate our exploration of adaptive interaction, where the system can dynamically decide whether to operate autonomously or request human intervention.

4 Methodology

This section presents *OS-Kairos*. We first introduce a collaborative probing framework that dynamically annotates the confidence scores at each interaction step. Then, we will describe confidence-driven interaction that integrates confidence scoring into GUI agents, resulting in adaptive interaction. Figure 3 shows an overall illustration.

4.1 Collaborative Probing Framework

This framework integrates instruction collection, confidence annotation, and data refinement, enabling the generation of a high-quality trajectory dataset with a confidence score for each step.

Instruction Collection. We first collect complex instructions $\mathcal{T} = \{\tau_1, \tau_2, \cdots, \tau_N\}$ from publicly available datasets and human designers, and then expanded by LLMs (e.g., GPT-4) to increase diversity. To comprehensively probe the model's confidence at each step, these instructions incorporate factors such as language type (both English and Chinese), 12 APPs, and 12 topics. The distribution of APP and topic is shown in Appendix B.

Confidence Annotation. Our confidence probing framework employs an agent-critic collaborative paradigm. To address the challenge of dynamic evaluation and expand coverage to commercial applications, the framework first utilizes Android Studio to connect real mobile devices and establish bidirectional communication with the probed GUI agent $\mathcal{F}_p$ deployed at the service station. Second, inspired by (Ma et al., 2024b; Wang et al., 2024a), we assume that the layout-parse model enhanced `GPT-4o` is the state-of-the-art critic model $\mathcal{F}_c$, capable of effectively supervising and guiding $\mathcal{F}_p$, thereby ensuring the dynamic probing of the entire trajectory. Additionally, $\mathcal{F}_c$ can monitor the entire probing process, including the planned schedule, current step, and instruction completion. The details of the prompt are provided in Appendix C.1.

Specifically, given a user instruction τ_i, $\mathcal{F}_p$ predicts the next action a_t^p using the action prompt P_p at step t, followed by $\mathcal{F}_p(a_t^p|P_p(s_t, h_{t-1}, \tau_i))$. For example, $\mathcal{F}_p$ responds to the instruction τ_i with the first step "Open Amazon APP" as "CLICK <616, 371>". Meanwhile, $\mathcal{F}_c$ first generates a plan schedule L using the prompt P_l. Based on the current step L_t at step t, it will also respond with a supervisory action a_t^c using the action prompt P_c. Subsequently, $\mathcal{F}_c$ evaluates the effectiveness of the current step execution with the scoring prompt P_h:

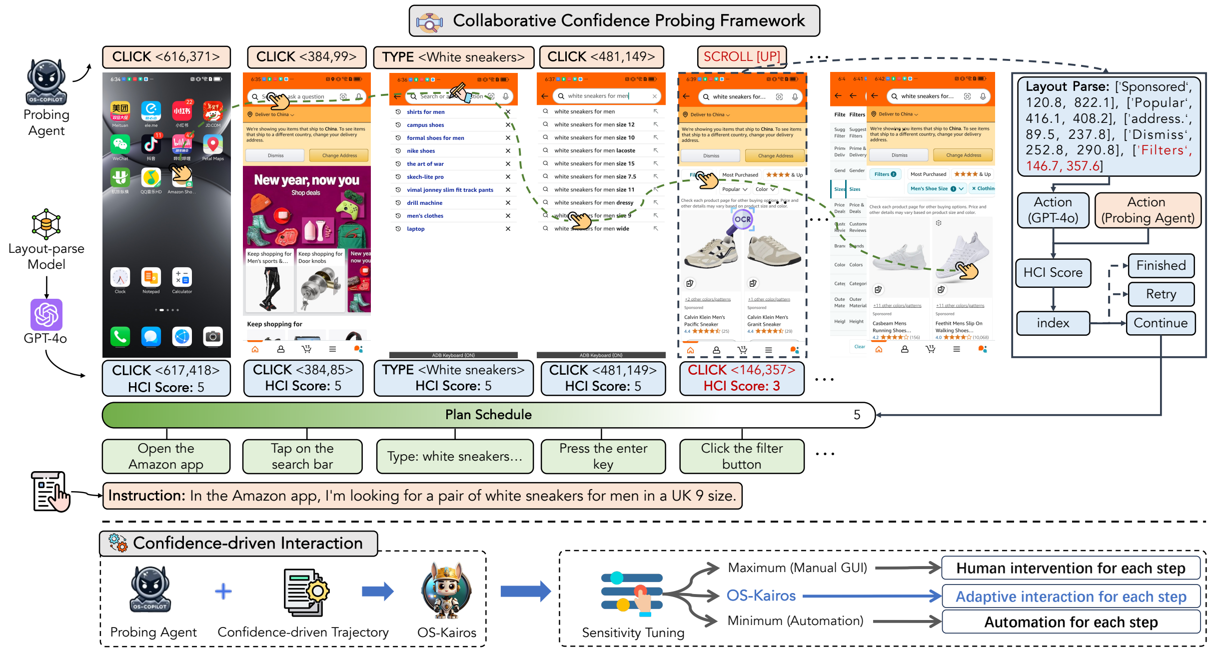


Figure 3: Overall pipeline of *OS-Kairos*: After collecting instructions for each complex scenario, we annotate confidence scores at each interaction step of the probing agent through a collaborative probing framework. Finally, confidence-driven interaction integrates the adaptive human intervention into the GUI agent, resulting in *OS-Kairos*.

$$f_{\text{score}} :< \tau_i, L_t, s_t, a_t^p, a_t^c, h_{t-1} > \xrightarrow{\mathcal{F}_c} \text{score}_t, \quad (1)$$

where f_{score} ranges from 1 to 5. When score_t is 5, we consider that $\mathcal{F}_p$ is correct to execute the current step, otherwise, the framework will execute action a_t^c to continue probing the next step until the instruction is judged finished by $\mathcal{F}_c$. For instance, $\mathcal{F}_p$ provides the action "SCROLL[UP]" at step 5, while the corrective action is "CLICK $< 146, 357 >$" on the filter button. The framework also incorporates reflective mechanisms to monitor the plan schedule. At each step, $\mathcal{F}_c$ determines the completion of instruction τ_i and the current step L_t:

$$f_t : \langle L, s_t \rangle \xrightarrow{\mathcal{F}_c} \text{index}, \quad (2)$$

where the framework will retry the current step if $t = \text{index}$, otherwise to continue execution.

Data Refinement. In this phase, we validate and refine these GUI trajectories, ensuring alignment between action and confidence score. The distribution of each step in the complex scenarios is based on its score, as shown in Appendix B. Notably, the distribution of actions scored 5 is concentrated in normal steps, such as "Open APP" or "Click Search Bar". However, once the instructions contain complex steps, the confidence scores of the probing agent decrease significantly. Hence, we can identify the over-execution steps of the probed GUI agent and treat these steps as requiring advanced GUI agent guidance or human intervention.

4.2 Confidence-driven Interaction

This phase integrates confidence scoring, resulting in a GUI agent with adaptive interaction.

Confidence Scoring Integration. Employing the trajectory from the collaborative probing framework, we introduce *OS-Kairos*, which integrates confidence scoring with the probed GUI agent $\mathcal{F}_p$. Specifically, we employ supervised training to fine-tune $\mathcal{F}_p$. Formally, the training objective $\mathcal{L}_{\textit{OS-Kairos}}$ of the next-word prediction can be expressed as:

$$\mathcal{L}_{\textit{OS-Kairos}} = \sum_{i=1}^{N} \mathcal{P}_\theta(a_t^i || \text{score}_t \mid P_p(s_t, \tau_i, h_{t-1}, a_t^{<i})), \quad (3)$$

where N is the token number of a_t and score_t, $||$ is the concatenated operator of the prediction of action and score, and θ is the trainable parameters in *OS-Kairos*. This optimization is more stable compared to multi-task learning, as it not only preserves *OS-Kairos*'s action prediction ability but also generates confidence in the predicted actions.

Adaptive Interaction GUI Agent. To ensure interactive adaptivity, we introduce a threshold to control *OS-Kairos*'s sensitivity. Formally, for a given threshold γ, *OS-Kairos* satisfies:

$$f_{\text{confidence}} : \langle a_t, \text{score}_t \rangle \xrightarrow{<\gamma} \text{Interactive}, \quad (4)$$

where human intervention is triggered if the current

Models	API	SCROLL	PRESS	STOP	CLICK		TYPE		Total		TSR
					Type (%) ↑	SR (%) ↑	Type (%) ↑	SR (%) ↑	Type (%) ↑	SR (%) ↑	
GPT-4o	✓	22.22	**100.00**	46.67	86.95	74.63	93.62	90.07	87.59	76.35	39.13
GLM-4V-Plus	✓	0.00	0.00	20.00	95.88	37.65	21.99	20.57	81.57	33.80	4.35
Qwen-VL-MAX	✓	0.00	**100.00**	92.21	51.25	38.33	96.45	92.21	58.73	46.89	29.81
Auto-UI	✗	44.44	0.00	0.00	2.93	0.15	0.00	0.00	2.81	0.59	0.00
Qwen2-VL-7B	✗	22.22	85.71	0.00	37.98	15.69	55.32	42.55	40.75	20.49	0.00
OS-Atlas-Pro-7B	✗	66.67	0.00	20.00	97.80	62.46	99.29	63.12	95.90	61.36	14.29
OS-Kairos	✗	**100.00**$_{33.33\uparrow}$	**100.00**$_{100.00\uparrow}$	**100.00**$_{80.00\uparrow}$	**99.85**$_{2.05\uparrow}$	**96.33**$_{33.87\uparrow}$	**100.00**$_{0.71\uparrow}$	**92.86**$_{29.74\uparrow}$	**99.88**$_{3.98\uparrow}$	**95.90**$_{34.54\uparrow}$	**88.20**$_{73.91\uparrow}$

Table 2: Comparison of OS-Kairos with baselines in complex scenarios (zero-shot setting). We report the overall accuracy for *Type*, *SR*, and *TSR*, along with fine-grained accuracy for each action. Subscripts indicate relative improvement over the OS-Atlas-Pro-7B, with the best result highlighted in **bold**.

action's confidence is below γ, otherwise, it is automatic. Notably, *OS-Kairos* switches to autonomous mode if the γ is set to minimum value, or to fully interactive GUI if it is set to maximum value.

5 Experiments

This section will introduce the experimental setup, followed by our empirical results and analysis.

5.1 Experiment Setup

Datasets. Thanks to the confidence probing framework, we can evaluate the *OS-Kairos* in complex scenarios by splitting the generated trajectories. Moreover, we evaluate it on established benchmarks such as AITZ and Meta-GUI. Details are provided in Appendix C.2.

Models. In the confidence probing framework, we use the open-source GUI agent OS-Atlas-Pro-7B (Wu et al., 2024b) as our probing model. Our objective is to probe the confidence score of the GUI model at each step, thereby introducing *OS-Kairos* to enhance its effectiveness. Additionally, we use `GPT-4o` (Achiam et al., 2023) as our critic model. The layout-parse model is resnet18 and convnextTiny for OCR-detection and recognition models, respectively (Tang et al., 2019).

Baselines. We compare the proposed *OS-Kairos* with the following types:

- **Multimodal API-based models.** We consider MLLM-powered GUI agents, including `GPT-4o` (Achiam et al., 2023), `GLM-4V-Plus` (GLM et al., 2024) and `Qwen-VL-MAX` (Bai et al., 2023), which are strong baselines in zero-shot settings.
- **Multimodal Open-source models.** In the zero-shot setting, we also consider GUI-adapted MLLMs, including CogAgent (Hong et al., 2024), Auto-UI (Zhang and Zhang, 2024), Qwen2-VL-7B (Bai et al., 2023), OS-Atlas-Pro-7B (Wu et al., 2024b). In the fine-tuning setting, we compare *OS-Kairos* with fine-tuned models on datasets.

Metrics. Following Wu et al. (2024b), we report the action type accuracy (*Type*), step-wise success rate (*SR*), and task success rate (*TSR*). Besides, we evaluate the human intervention success rate (*HSR*), intervention precision (IP), autonomous precision (AP), and relative efficiency (*RE*). More details of metrics and implementation can be found in Appendix C.3 and Appendix C.4.

5.2 Main Results

We present comparison results for complex scenarios and two benchmarks with zero-shot settings in Table 3, Appendix C.6.1, and Appendix C.6.2. Table2 provides a comprehensive comparison of fine-tuning settings. Our key findings are as follows:
In zero-shot setting: Superior Performance and Better Effectiveness. Without changing the model capabilities, *OS-Kairos* significantly outperforms the zero-shot baseline across three datasets, highlighting its effectiveness. Specifically, the adaptive interaction of *OS-Kairos* effectively identifies complex steps that trigger human intervention. This not only improves the prediction accuracy for each action but also enhances overall performance. For example, it achieves 95.90% in *SR* and 88.20% in *TSR* for complex scenarios. Although API-based and proprietary models realizes domain enhancement for GUI tasks, they cannot identify complex steps, resulting in over-execution and task failure. Moreover, *OS-Kairos* yields promising results on the other two datasets when applying confidence scoring integration to the original dataset, highlighting its generality (see Appendix C.6.3).
In fine-tuning setting: Competitive Performance and Precise Improvement. Although fine-tuning can alleviate the over-execution of GUI agents, *OS-Kairos* still outperforms them, achieving high *SR* and notable improvements in *TSR*. For example, it shows relative improvements ranging from 26.09% to 85.72% in complex scenarios. Furthermore, *OS-*

Models	Mode	SCROLL	PRESS	STOP	CLICK		TYPE		Total		TSR
					Type (%) ↑	SR (%) ↑	Type (%) ↑	SR (%) ↑	Type (%) ↑	SR (%) ↑	
OS-Kairos Dataset											
Auto-UI	FT	0.00$_{44.44\downarrow}$	71.43$_{71.43\uparrow}$	80.00$_{80.00\uparrow}$	98.83$_{95.90\uparrow}$	67.16$_{67.01\uparrow}$	97.16$_{97.16\uparrow}$	0.71$_{0.71\uparrow}$	96.96$_{94.15\uparrow}$	55.74$_{55.15\uparrow}$	2.48$_{2.48\uparrow}$
Qwen2-VL-7B	FT	55.56$_{33.34\uparrow}$	42.86$_{43.85\downarrow}$	**100.00**$_{100.00\uparrow}$	98.83$_{60.85\uparrow}$	85.34$_{69.65\uparrow}$	99.29$_{43.97\uparrow}$	90.78$_{48.23\uparrow}$	98.48$_{57.73\uparrow}$	85.83$_{65.34\uparrow}$	62.11$_{62.11\uparrow}$
OS-Atlas-Pro-7B	FT	22.22$_{44.45\downarrow}$	14.29$_{14.29\uparrow}$	93.33$_{73.33\uparrow}$	99.56$_{1.76\uparrow}$	84.75$_{22.29\uparrow}$	99.29$_{0.00\uparrow}$	91.49$_{28.37\uparrow}$	98.71$_{2.81\uparrow}$	84.78$_{23.42\uparrow}$	55.90$_{41.61\uparrow}$
OS-Kairos	ZS	**100.00**$_{33.33\uparrow}$	**100.00**$_{100.00\uparrow}$	**100.00**$_{80.00\uparrow}$	**99.85**$_{2.05\uparrow}$	**96.33**$_{33.87\uparrow}$	**100.00**$_{0.71\uparrow}$	**92.86**$_{29.74\uparrow}$	**99.88**$_{3.98\uparrow}$	**95.90**$_{34.54\uparrow}$	**88.20**$_{73.91\uparrow}$
AITZ Benchmark											
CogAgent	FT	70.22$_{13.81\uparrow}$	45.95$_{2.35\downarrow}$	24.60$_{19.84\uparrow}$	88.23$_{8.33\uparrow}$	66.15$_{14.64\uparrow}$	45.80$_{21.60\downarrow}$	21.80$_{10.20\downarrow}$	72.59$_{6.73\uparrow}$	53.28$_{8.76\downarrow}$	/
Auto-UI	FT	61.40$_{13.48\downarrow}$	57.70$_{8.61\uparrow}$	74.40$_{14.28\uparrow}$	74.56$_{30.19\uparrow}$	32.20$_{19.48\uparrow}$	87.80$_{14.80\uparrow}$	81.40$_{13.60\uparrow}$	82.98$_{9.19\uparrow}$	47.69$_{13.23\uparrow}$	/
Qwen2-VL-7B	FT	71.38$_{52.74\uparrow}$	21.85$_{0.66\uparrow}$	78.57$_{78.57\uparrow}$	88.30$_{17.25\uparrow}$	51.10$_{18.21\uparrow}$	87.80$_{5.00\uparrow}$	45.00$_{0.00\uparrow}$	85.14$_{18.86\uparrow}$	55.23$_{26.98\uparrow}$	1.78$_{1.78\uparrow}$
OS-Atlas-Pro-7B	FT	62.23$_{34.83\uparrow}$	28.48$_{27.82\uparrow}$	73.61$_{68.45\uparrow}$	90.75$_{2.56\downarrow}$	58.74$_{23.87\uparrow}$	89.00$_{3.80\uparrow}$	44.00$_{16.60\uparrow}$	86.69$_{1.49\uparrow}$	58.32$_{24.66\uparrow}$	11.15$_{11.15\uparrow}$
OS-Kairos	ZS	**91.17**$_{63.77\uparrow}$	**73.51**$_{72.85\uparrow}$	**91.65**$_{86.49\uparrow}$	**98.43**$_{5.12\uparrow}$	**89.46**$_{54.59\uparrow}$	**99.20**$_{14.00\uparrow}$	**72.80**$_{45.40\uparrow}$	**96.81**$_{11.61\uparrow}$	**87.54**$_{53.88\uparrow}$	**24.51**$_{24.51\uparrow}$
Meta-GUI Benchmark											
Auto-UI	FT	42.95$_{17.95\downarrow}$	65.91$_{65.91\uparrow}$	53.08$_{53.08\uparrow}$	84.23$_{57.33\uparrow}$	53.99$_{51.30\uparrow}$	86.55$_{86.55\uparrow}$	1.75$_{1.75\uparrow}$	73.02$_{53.00\uparrow}$	48.49$_{42.04\uparrow}$	20.42$_{20.42\uparrow}$
Qwen2-VL-7B	FT	89.10$_{89.10\uparrow}$	72.73$_{72.73\uparrow}$	90.02$_{89.59\uparrow}$	94.61$_{43.07\uparrow}$	83.19$_{80.86\uparrow}$	97.08$_{59.07\uparrow}$	64.33$_{46.20\uparrow}$	93.17$_{57.27\uparrow}$	83.43$_{80.39\uparrow}$	57.29$_{57.08\uparrow}$
OS-Atlas-Pro-7B	FT	84.62$_{68.59\uparrow}$	70.45$_{70.45\uparrow}$	89.38$_{89.38\uparrow}$	96.01$_{1.48\uparrow}$	85.53$_{48.24\uparrow}$	95.91$_{35.68\uparrow}$	65.50$_{50.30\uparrow}$	93.49$_{27.40\uparrow}$	84.27$_{60.68\uparrow}$	57.29$_{56.78\uparrow}$
OS-Kairos	ZS	**99.36**$_{83.33\uparrow}$	**100.00**$_{100.00\uparrow}$	**94.73**$_{94.73\uparrow}$	**99.81**$_{5.28\uparrow}$	**96.66**$_{59.37\uparrow}$	**98.83**$_{38.60\uparrow}$	**95.32**$_{80.12\uparrow}$	**98.49**$_{32.40\uparrow}$	**96.36**$_{72.77\uparrow}$	**87.71**$_{87.29\uparrow}$

Table 3: Comparison of *OS-Kairos* in the fine-tuning setting. ZS and FT denote zero-shot and fine-tuning evaluations, respectively. We report overall accuracy for *Type*, *SR*, and *TSR*, as well as fine-grained accuracy for each action. Subscripts indicate relative improvement over the ZS baseline, with the best result highlighted in **bold**.

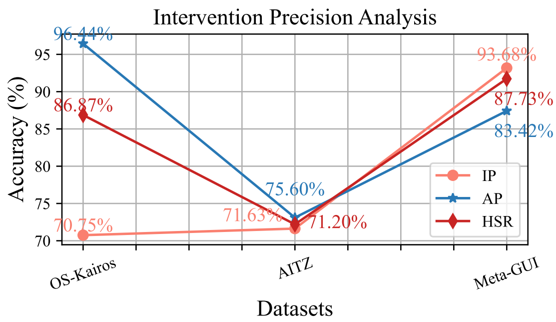


Figure 4: Analysis of intervention precision.

Kairos achieves precise improvements by identifying complex steps (e.g., `SCROLL`), while fine-tuning may introduce side effects in specific actions and encounter optimization bottlenecks.

Confidence Scoring: Effective interaction. In Figure 4, *OS-Kairos* shows accurate confidence evaluation (*HSR*). Thus, It does not interfere with the autonomous steps (*AP*), as seen in complex scenarios (96.44%) and MetaGUI (93.18%). Notably, *OS-Kairos* achieves over 70% precision in all human intervention steps, highlighting its effective interaction. With high-quality sampling, we consider *OS-Kairos*'s precision can be improved further (e.g., AITZ). Additionally, the ablation study of the critic model shows that `GPT-4o` is the optimal choice (see Appendix C.7).

5.3 Analysis

5.3.1 Dynamic Evaluation of TSR

Previous benchmark evaluations have been based on static analysis, which limits the autonomous planning and generality of the GUI agent. Thus, we also report the real-world *TSR* on mobile devices. As shown in Table 4, the baselines only achieve *TSR* of 4% and 26%. Given that the *TSR* of `GPT-4o` is 36%, we see that *OS-Kairos* is approaching this upper limit. When *OS-Kairos*$_{\text{human}}$ is assisted by human intervention, the *TSR* increases from 32% to 70%, indicating adaptive interaction is an effective paradigm for real-world GUI agents.

Models	Human Steps	Actual Steps	RE (%)↑	TSR (%)↑
GPT-4o	229	302	75.83	36.00
Qwen2-VL-7B	229	397	57.68	4.00
OS-Atlas-Pro-7B	229	359	63.79	26.00
OS-Kairos$_{\text{GPT-4o}}$	229	**245**	**93.47**	32.00
OS-Kairos$_{\text{human}}$	229	265	86.42	**70.00**

Table 4: Analysis of efficiency and dynamic *TSR*.

5.3.2 Efficiency Evaluation

Table 4 reports the efficiency in a real-world environment. First, we count the optimal number of human steps on 50 instructions, about 429 steps. Next, we evaluate the actual step counts for baseline and *OS-Kairos*, respectively. Notably, the model max steps are set to 10. We observe that the baseline models tend to over-execute when faced with a complex step. In contrast, *OS-Kairos* more closely resembles human manipulation of a GUI, achieving 86.42% and 93.47% in *RE*.

5.3.3 Comparing Prompt-based Interaction

Table 5 presents a comparison of *OS-Kairos* with prompt-based interactive models. We see that the interactive mechanism of *OS-Kairos* outperforms

Models	Interactive	Type (%)↑	SR (%)↑	TSR (%)↑	HSR (%)↑
GPT-4o	Prompt	88.80	79.25	46.58	/
GLM-4V-Plus	Prompt	88.34	79.03	47.83	/
Qwen2-VL-7B	Prompt	76.42	38.44	25.47	/
OS-Atlas-Pro-7B	Prompt	59.02	95.67	9.94	0.00
OS-Kairos	FT	**99.88**	**95.90**	**88.20**	**86.87**

Table 5: Analysis of interactive paradigms vs. prompt-based baseline in complex scenarios.

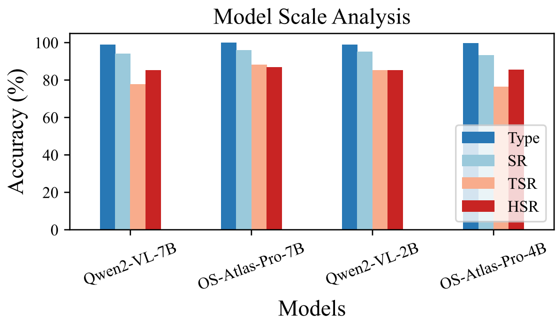


Figure 5: Generality of *OS-Kairos* across model scale.

the prompt-based paradigm, particularly surpassing the prompt-based OS-Atlas-Pro-7B in terms of *HSR*. Despite the strong grounding capabilities of `GPT-4o` and `GLM-4V-Plus`, API-based agents present instability, resulting in over-execution and sub-optimal performance. Among open-source GUI agents, Qwen2-VL-7B performs more consistently than OS-Atlas-Pro-7B, because prompt-based interactive severely disrupts the latter's instruction-following ability.

5.4 General Effectiveness across Scales

Model Scale. Although the dynamic detection framework is built on the OS-Atlas-Pro-7B backbone, confidence scores and actions generated are supposed to be downwardly compatible. In other words, weaker models can be enhanced through data distillation and confidence scoring integration. Figure 5 shows that *OS-Kairos* can be successfully generalized to the 2B∼7B model. First, *Type* and *SR* are effective, guaranteeing a TSR of 76.40% on the Qwen2-VL-7B model, 77.64% on OS-Atlas-Pro-4B, and 85.09% on Qwen2-VL-2B. Thus, the combination of confidence scoring and data distillation will enhance weak models, thus satisfying the deployment in resource-constrained environments.

Data Scale. To evaluate the effect of data scaling on confidence scoring integration, we divide the trajectories from the probing framework into different scales for training and test data. As shown in Table 6, *OS-Kairos* remain stable in *Type* and *SR* scores across scales. Benefiting from its high *HSR*, *OS-Kairos*'s *TSR* accuracy reaches 76.19%∼88.20%, proving that the integration of confidence scoring into *OS-Kairos* requires only a small number of probing data at a significantly lower cost than fine-tuning the GUI agent.

Data Scaling	Type (%)↑	SR (%)↑	TSR (%)↑	HSR (%)↑
9:1	99.25	92.21	76.19	84.67
8:2	**99.88**	**95.90**	**88.20**	**86.87**
7:3	99.46	94.16	83.94	84.79
6:4	99.41	94.05	78.30	84.47

Table 6: Varying data scale in confidence scoring.

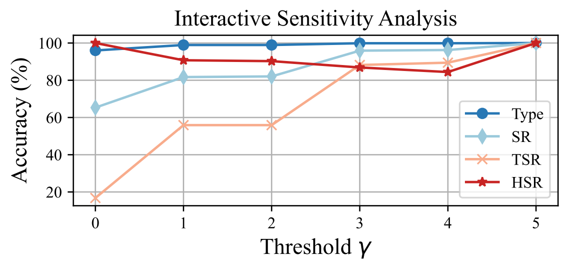


Figure 6: Threshold impact on interactive sensitivity.

5.4.1 Interactive Sensitivity

OS-Kairos use a threshold to achieve adaptive interaction. To analyze the adaptive interaction sensitivity of *OS-Kairos*, we ablate threshold γ from 0 to 5. In Figure 6, *TSR* and *SR* increase with the rise in interactive sensitivity, indicating that human intervention enhances the effectiveness of GUI agents in complex scenarios. The *HSR* and *Type* accuracy remain stable across different thresholds, indicating that *OS-Kairos* can effectively identify complex steps, especially in coordinates and input scenarios, alleviating over-execution of the GUI agent. The ablation study of adaptive interaction shows that *OS-Kairos* is more flexible (See Appendix C.8).

6 Conclusion

This study identifies a key challenge of over-execution in GUI agents, which poses substantial risks in complex scenarios, such as those involving ambiguous user instructions, unexpected interruptions, and environmental hijacks. To address the challenge, we introduce *OS-Kairos*, an adaptive GUI agent capable of predicting confidence levels at each step and efficiently deciding whether to act autonomously or seek human intervention. Concretely, we propose a collaborative probing framework for annotating confidence scores at each interaction step. By integrating confidence scoring, *OS-Kairos* outperforms previous GUI agents and API-based models, with improved effectiveness, scalability, generality, and efficiency.

Limitation

We acknowledge two primary limitations in our study. First, we only sampled instructions from three typical complex scenarios, as our focus was to investigate why existing GUI agents struggle with *TSR* and generate action confidence scores without loss of generality. Notably, we demonstrate the effectiveness of *OS-Kairos* on the AITZ and Meta-GUI benchmarks, which provide additional diverse instructions for complex scenarios. Besides, the generalization capabilities of *OS-Kairos* can mitigate these limitations. Second, experiments were focused on our probing dataset and two benchmark datasets, highlighting the need for complex scene probing and confidence scoring integration. Given that confidence scoring relies on proprietary models and high-quality human sampling, we anticipate that future research will explore the optimization of our approach to confidence scoring and evaluate new benchmark datasets.

Ethics Statement

This section presents the ethics statements in the following aspects: (i) Privacy. The probing instructions are sourced from publicly available datasets, human designers, and `GPT4o`, covering 12 apps and 12 topics. Temporary accounts were used to register these apps, and the trajectories generated by our collaborative probing framework are available, ensuring that no personal data or personally identifiable information was collected. The two benchmarks employed also implemented safeguards to protect privacy (Zhang et al., 2024b; Sun et al., 2022). Moreover, *OS-Kairos*, as an open-source GUI agent that does not rely on any external information and supports local deployment. (ii) System security. *OS-Kairos* follows the first principles thinking (Zhang and Zhang, 2024), manipulates the GUI like a human being, and can initiate human intervention in scenarios involving system security to ensure safety. (iii) Potential social impacts. *OS-Kairos* can improve the effectiveness of GUI execution instructions. Unlike fully autonomous GUI agents, *OS-Kairos* will proactively request authorization and acquire personal information, thus reducing malicious abuse.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024a. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. In *Forty-first International Conference on Machine Learning*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2024b. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Paulo Roberto Guimarães Couto, Jailton Carreteiro Damasceno, SP de Oliveira, and WK Chan. 2013. Monte carlo simulations applied to uncertainty in measurement. *Theory and applications of Monte Carlo simulations*, 2:27–51.

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.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Liujianfeng Liujianfeng, Ang Li, Jian Luan, Bin Wang, Rui Yan, et al. 2024. Mobile-bench: An evaluation benchmark for llm-based mobile agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8813–8831.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

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.

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

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020. Mapping natural language instructions to mobile UI action sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8198–8210, Online. Association for Computational Linguistics.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. 2024. 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*.

Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. 2024a. Caution for the environment: Multimodal agents are susceptible to environmental distractions. *arXiv preprint arXiv:2408.02544*.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024b. CoCo-agent: A comprehensive cognitive MLLM agent for smartphone GUI automation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9097–9110, Bangkok, Thailand. Association for Computational Linguistics.

James B McDonald and Yexiao J Xu. 1995. A generalization of the beta distribution with applications. *Journal of Econometrics*, 66(1-2):133–152.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, et al. 2024. Gui agents: A survey. *arXiv preprint arXiv:2412.13501*.

Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. 2024. Screenagent: a vision language model-driven computer control agent. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Xinyue Yang, Jiadai Sun, Yu Yang, Shuntian Yao, Tianjie Zhang, et al. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*.

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

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.

Abishek Sridhar, Robert Lo, Frank F Xu, Hao Zhu, and Shuyan Zhou. 2023. Hierarchical prompting assists large language model on web navigation. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022. Meta-gui: Towards multi-modal conversational agents on mobile gui. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6699–6712.

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*.

Jun Tang, Zhibo Yang, Yongpan Wang, Qi Zheng, Yongchao Xu, and Xiang Bai. 2019. Seglink++: Detecting dense and arbitrary-shaped scene text by instance-aware component grouping. *Pattern recognition*, 96:106954.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Taiyi Wang, Zhihao Wu, Jianheng Liu, Derek Yuen, HAO Jianye, Jun Wang, and Kun Shao. 2024b. Distrl: An asynchronous distributed reinforcement learning framework for on-device control agent. In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*.

Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Liujian Liujianfeng, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024a. MobileVLM: A vision-language model for better intra- and inter-UI understanding. In *Findings of the Association for Computational*

Linguistics: EMNLP 2024, pages 10231–10251, Miami, Florida, USA. Association for Computational Linguistics.

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*.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Qingwei Lin, Saravan Rajmohan, et al. 2024a. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024b. Android in the zoo: Chain-of-action-thought for GUI agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031, Miami, Florida, USA. Association for Computational Linguistics.

Shaoqing Zhang, Zhuosheng Zhang, Kehai Chen, Xinbei Ma, Muyun Yang, Tiejun Zhao, and Min Zhang. 2024c. Dynamic planning for LLM-based graphical user interface automation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1304–1320, Miami, Florida, USA. Association for Computational Linguistics.

Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3132–3149, Bangkok, Thailand. Association for Computational Linguistics.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yifei Zhou, Hao Bai, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. In *Automated Reinforcement Learning: Exploring Meta-Learning, AutoML, and LLMs*.

Zichen Zhu, Hao Tang, Yansi Li, Kunyao Lan, Yixuan Jiang, Hao Zhou, Yixiao Wang, Situo Zhang, Liangtai Sun, Lu Chen, et al. 2024. Moba: A two-level agent system for efficient mobile task automation. *arXiv preprint arXiv:2410.13757*.

A Why GUI agents have poor *TSR*?

In our pilot experiment, the *TSR* of autonomous GUI agents is significantly lower than interactive GUI agents. This difference is attributed to the poor exact match for *SR*, particularly for the `CLICK` and `TYPE` actions. In contrast, interactive GUI agents can mitigate this limitation through human intervention. Intuitively, we consider the impact of exact matching on trajectory steps to be exponential. Formally, for a trajectory with k steps, the probability that instruction τ_i can be completed is:

$$\mathit{TSR}_{\tau_i} = \prod_{j=1}^{k} \mathit{SR}_j, s.t., \mathit{SR}_j \sim \beta(u, l). \quad (5)$$

Herein, we assume that SR_j follows β distribution (McDonald and Xu, 1995). u and l are hyperparameters that control the distribution of *SR*. The expectation $\mathbb{E}[\mathit{SR}_{\tau_i,j}] = \frac{u}{u+l}$ and variance $\mathrm{Var}(\mathit{SR}_{\tau_i,j}) = \frac{ul}{(u+l)^2(u+l+1)}$. Additionally, the expectation $\mu = \mathbb{E}[\ln(\mathit{SR}_{\tau_i,j})] = \psi(u) - \psi(u+l)$, and the variance $\sigma^2 = \mathrm{Var}[\ln(\mathit{SR}_{\tau_i,j})] = \psi'(u) - \psi'(u+l)$, where $\psi(\cdot)$ and $\psi'(\cdot)$ denote the digamma and trigamma functions, respectively. The TSR_{τ_i} follows a normal distribution:

$$\ln(\mathit{TSR}_{\tau_i}) \sim \mathcal{N}\left(k \cdot \mu\ ,\ k \cdot \sigma^2\right), \quad (6)$$

then,

$$\begin{aligned} \mathit{TSR}_{\tau_i} \sim \mathrm{LogNormal}(&\exp^{k\mu + \frac{k\sigma^2}{2}}, \\ &\exp^{2k\mu + k\sigma^2}(\exp^{k\sigma^2} - 1)). \end{aligned} \quad (7)$$

Considering the boundedness of k, we utilize Monte Carlo simulations (Couto et al., 2013) to estimate the TSR_{τ_i} probability distribution. As shown in Figure 7(a), we assume that $\mathit{SR}_{\text{auto}}$ exhibits high variance in the beta distribution, due to the effect of step complexity. The $\mathit{SR}_{\text{manual}}$, determined by the human intervention executed at each step, lies within the right-interval and represents the upper limit of the GUI agent's capability. In this study, we aim to adaptively interact to bring *OS-Kairos* closer to this upper bound. As shown in Figure 7(b), we observe that $\mathit{TSR}_{\tau_i,\text{auto}}$ is impacted by the complexity of the step, with the probability of TSR_{τ_i} falling below 20%. In contrast, *OS-Kairos* can align with expectations and remains consistently close to the upper bound of performance.

When generalized to N independent and identically distributed instructions, the average *TSR*

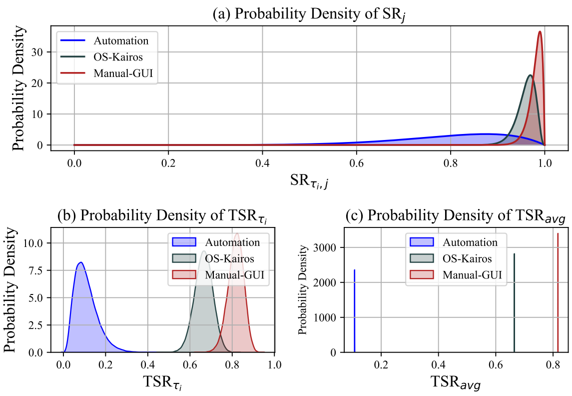


Figure 7: Illustration of the probability density of $SR_{\tau_i,j}$, $TSR\tau_i$ for a single trajectory, and TSR_{avg} for N trajectories.

satisfies:

$$TSR_{avg} = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{k} SR_j. \quad (8)$$

According to center limit theory, TSR_{avg} also satisfies normal distribution:

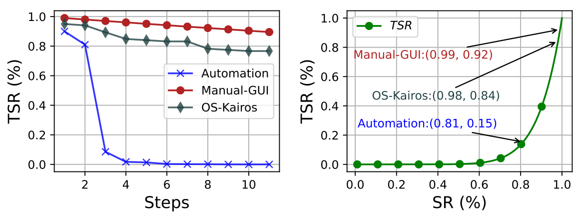
$$\begin{aligned} TSR_{avg} \sim \mathcal{N}(&\exp^{k\mu+\frac{k\sigma^2}{2}}, \\ &\exp^{2k\mu+k\sigma^2}(\exp^{k\sigma^2}-1)/N). \end{aligned} \quad (9)$$

As shown in Figure 7(c), we observe the exponential effect of *SR* on *TSR*. In the autonomous mode, $TSR_{\text{avg,auto}}$ is nearly 0%. In contrast, *OS-Kairos* and the fully interactive GUI agent both achieve success rates exceeding 60%.

Subsequently, we further assume that the *SR* of single, complex, and interactive steps are m, q, p respectively. When δ complex steps are available, TSR_{τ_i} satisfies:

$$0 \approx m^{\delta}\cdot q^{k-\delta} < TSR_{\tau_i} < m^{\delta}\cdot p^{k-\delta} \approx p^k. \quad (10)$$

As shown in Figure 8(a), the *SR* effect on the *TSR*



(a) Impact of TSR on Trajectory Length (b) Impact of SR on TSR

Figure 8: Illustration of the effect of *SR* on TSR_{avg} for N trajectories.

of a single trajectory is consistent with Figure 7(a). In other words, once there are complex steps in the trajectory, the TSR_{τ_i} will decrease significantly, while human intervention can jump such steps, thus remaining effective. Therefore, *OS-Kairos* aims to recognize such steps, seek human intervention, and thus exponentially enhance *TSR*, as shown in Figure 8(b).

B Instruction Distribution

In our dynamic capability probing, we collect 1,000 instructions for three complex scenarios, covering 12 topics and 13 apps. The distributions of topics and apps are shown in Figure 9 and Figure 10. The distribution of scenarios is shown in Figure 11.

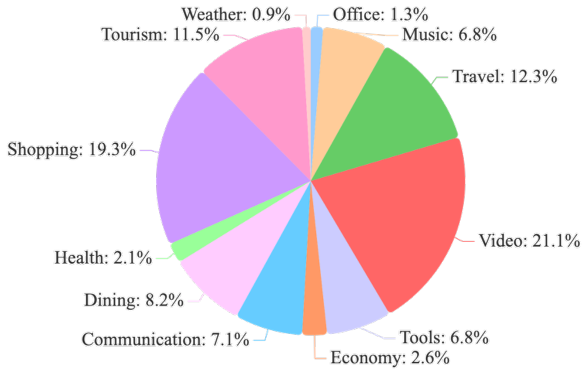


Figure 9: Subject distribution of instructions.

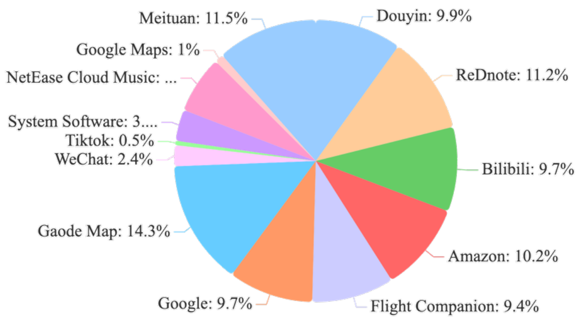


Figure 10: APP distribution of instructions.

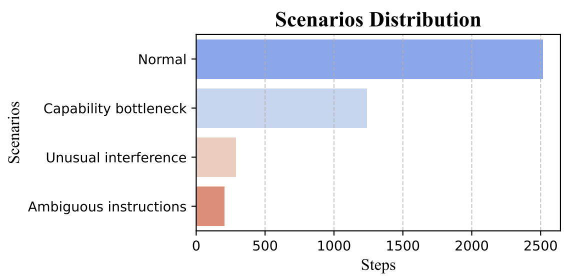


Figure 11: Distribution of steps in different scenarios.

C Implementation Details

C.1 Prompt Templates

Below are the prompt templates for designing *OS-Kairos*. In the collaborative probing framework, we design a planning prompt, action phase prompt, action prompt for probed GUI agent $\mathcal{F}_p$ and critic model $\mathcal{F}_c$, scoring prompt, and finishing judgment prompt. In the confidence-driven interaction phase, we only use the $\mathcal{F}_p$ action prompt to optimize and evaluate *OS-Kairos*. The pipeline controller fills these variables in them according to the context.

C.1.1 Planning Prompt Template

In the GUI capability probing framework, the critic model $\mathcal{F}_c$ generates the planning schedule of the user instruction based on the `GPT-4o` and the instruction planning prompt, as shown in Figure 12.

Planning Prompt Template

```
"You are now an expert in using mobile software. I need you to help me break down an instruction for using mobile software into multiple step-by-step instructions. Please follow my example format strictly.\n"
"For example:\n"
"Original instruction: Search the distance from Earth to Mars on Google.\n"
"Broken down instructions:\n"
"Step list":[
        "Open Google",
        "Click on the search box on the screen",
        "Type the distance from Earth to the Moon",
        "Select the correct search result or press Enter",
        "Instructions complete"
]
f"Original instruction: {goal}\n"
"Broken down instructions:\n"
"Please output the broken-down instructions directly in List format.\n"
```

Figure 12: Prompts of the critic model for generating the planning schedule of the user instruction.

C.1.2 Action Phase Prompt Template

In the collaborative probing framework, the critic model $\mathcal{F}_c$ determines the current step based on the `GPT-4o` and action phase prompts, as shown in Figure 13.

C.1.3 Action Prompt Template

In the collaborative probing framework, we obtain the prediction of GUI agents using action prompts. The action prompts for the probed GUI agent $\mathcal{F}_p$ and critic model $\mathcal{F}_c$ are shown in Figure 14 and Figure 15, respectively. Following (Wu et al., 2024b; Zhang et al., 2024b), we define the actions set, which comprises 7 kinds of actions: `CLICK`, `SCROLL`, `TYPE`, `PRESS_BACK`, `PRESS_HOME`, `COMPLETE`, and `IMPOSSIBLE`.

Action Phase Prompt Template

```
"### Background ###\n"
"You are an expert in completing tasks based on screenshots and instructions. I will provide you with a mobile screenshot and a step list. You should be able to tell from the screenshot and the list of steps what step you are currently at.\n"

f"The step list is: {step_list}\n"
"### Response requirements ###\n"
"You can only output the index of a list of steps."
"For example:\n"
"step_list: [
        "Open WeChat",
        "Click the Contacts or Search button (depending on your version of WeChat and settings)",
        "If you click on Contacts, find and click on your wife's avatar; if you click on the Search button, enter your wife's name or note in the search box",
        "Go to your and your wife's WeChat",
        "Go to the chat screen between you and your wife",
        "Click on the input box",
        "Enter: I'm HIMA the Intelligence, I'm going home for the weekend tonight, no more studying, thanks!",
        "Click the send button"
]\n"
If you think you're still in the main screen, output {step index: 0}; if you've completed the task 'Open WeChat', you output {step index: 1}; if you think you've completed clicking on the input box, you output {step index: 5}.. Your output should just be a number."

"That means that the output is produced after a certain number of steps have been completed."

"### Output format ###\n"
"Your output must strictly follow the format below:\n"
"{step index: }"\n"
```

Figure 13: Prompts of the critic model for generating the action phase.

C.1.4 Scoring Prompt Template

In the collaborative probing framework, the critic model $\mathcal{F}_c$ generates the score for the action of $\mathcal{F}_p$ based on `GPT-4o` and scoring prompt, as shown in Figure 16.

C.1.5 Completion Judgment Prompt Template

In the collaborative probing framework, the critic model $\mathcal{F}_c$ exploits `GPT-4o` to judge whether the instruction is completed. The prompt is shown in Figure 17.

C.2 Details of Datasets

We consider evaluating *OS-Kairos* on three customized complex scenarios and two benchmarks: AITZ (Zhang et al., 2024b) and Meta-GUI (Sun et al., 2022). The statistics of the dataset are shown in the Table 7.

- **AITZ** (Zhang et al., 2024b): The first dataset to employ chain-of-action thought (CoAT) connects perception (of screen layouts and UI elements) with cognition (of action decision-making) to enhance the AITW benchmark. This dataset comprises 2,504 operation trajectories across 18.6K real-world intentions. Based on the application domain, AITZ is also divided into five subsets: General, Install, GoogleApps, Single, and WebShopping.

- **Meta-GUI** (Sun et al., 2022): task-oriented dialogue dataset is released for interactive GUI agent. These utterances cut a trajectory into several dialogue turns. Meta-GUI consists of 1,009 trajectories with 16.4K steps. The data diversity lies in 11 applications of 6 topics.

OS-Kairos	**Trajectory**	**Screen**	**Goal**
Train	800	4078	759
Test	200	1054	198
AITZ	**Trajectory**	**Screen**	**Goal**
General	479	3607	479
Install	420	3627	420
Google Apps	242	1889	242
Single	844	2594	844
Web Shopping	519	6926	519
Meta-GUI	**Trajectory**	**Screen**	**Goal**
Train	897	14539	2286
Test	116	1923	336

Table 7: Dataset statistics.

C.3 Details of Evaluation Metrics

To ensure fair comparison across all baseline methods, we standardize the evaluation metrics for each action. We define the *SR* metrics for the three complex actions as follows:

- `CLICK`: GUI agent predictions are considered correct if and only if both action types and position coordinates <x, y>. Following (Zhang and Zhang, 2024), we measure performance by calculating the distance between the predicted and ground truth coordinates. We consider the coordinates to be correct if the distance between the coordinates and the ground truth is within 14% of the screen width.
- `TYPE`: GUI agent predictions are considered correct if and only if both action type and action content are correct.
- `SCROLL`: GUI agent predictions are considered correct if and only if both action type and direction argument (i.e., UP, DOWN, LEFT, and RIGHT) are correct.

Furthermore, *Type* measures the exact match score between the predicted action types (e.g., CLICK, SCROLL) and the ground truth. *TSR* requires that all steps in a trajectory be correctly executed. For *HSR*, we define four statistical metrics with the threshold γ:

- **True positive (TP)**: Neither the prediction confidence nor the ground truth exceeds the γ, i.e., the agent does not require and perform interactions.
- **False positive (FP)**: The prediction confidence is greater than the γ, but the ground truth does not, meaning the agent is not required, but interaction is performed.
- **True negative (TN)**: Both the prediction confidence and the ground truth exceed the γ, meaning the agent must also perform interaction.
- **False negative (FN)**: The prediction confidence is less than γ, but the ground truth is greater than γ, which means that the agent needs but does not perform interactions.

Hence, HSR can be calculated:

$$HSR = \frac{TP + TN}{TP + TN + FP + FN}. \tag{11}$$

In addition, *IP* calculates the accuracy of the intervention step where intervention is actually needed, while *AP* measures the accuracy of the autonomous step where autonomy is truly required. Hence, *IP* and *AP* can be calculated:

$$IP = \frac{TN}{TN + FN}, AP = \frac{TP}{TP + FP} \tag{12}$$

Following (Wang et al., 2024a), *RE* measures the relative efficiency of the GUI agent compared to the steps taken by humans. It demonstrates whether *OS-Kairos* can use the mobile device more efficiently.

C.4 Implement Details

For each dataset, we randomly split 80% trajectories as training data, and 20% trajectories as testing data. Dataset statistics are presented in Table 7 of Appendix C.2. To ensure a fair comparison with the baseline, we use `GPT-4o` to score between the probing model and ground truth actions in the two benchmarks, without relying on high-quality sampling. In the zero-shot scenario, we evaluate the GUI agent directly using prompt learning. In the fine-tuning scenario, we fine-tune the model for 8 epochs on the corresponding dataset with a learning rate of 1e-5. In the interactive mode, if not specifically mentioned, the threshold γ is set to 4. When human intervention is required at the current step, *OS-Kairos* uses ground truth for the evaluation of the data set or human guidance for the dynamic evaluation. Our experiments are conducted on 8×NVIDIA A100 80 GB GPUs.

C.5 Usage of Existing Artifacts

For API-based MLLMs, we access them directly via the official interface. For open-source MLLMs, we either download the model weights from Hugging Face[1] or reproduce the model using the same training strategy. In our proposed OS-Kairos, the layout-parse pipeline of the collaborative probing framework is built upon Modelscope [2]. Furthermore, we utilize `LLaMA-Factory`[3] to fine-tune the probed model on three datasets for confidence integration. Notably, the InternVL-based models are fine-tuned using `Xtuner`[4]. All licenses for these packages permit their use for standard academic research purposes.

C.6 Further Analysis

C.6.1 AITZ Benchmark

Table 8 presents a comparison of *OS-Kairos* with the baselines in the AITZ benchmark. In API-based MLLMs, although `GPT-4o` performs the best, it is nearly impossible to finish user instructions. Among the open-source GUI agents, OS-Atlas-Pro-7B outperforms the other baselines due to the adaptation of AITZ, but still exhibits low *SR* and cannot fully complete user instructions. In contrast, *OS-Kairos* achieves precise intervention in complex steps on top of OS-Atlas-Pro-7B, with significant improvements in actions and overall performance. As a result, *OS-Kairos*'s *TSR* increased from 0% to 24.51%.

C.6.2 Meta-GUI Benchmark

Meta-GUI benchmark dataset is an out-of-domain (OOD) task against probing models, which allows for probing more complex steps and generating the confidence level for each step. Table 9 presents the performance of *OS-Kairos* on the Meta-GUI benchmark compared to the baseline. First, API-based MLLMs exhibit lower *SR* (17.19% to 32.72%) and *Type* (54.74% to 69.85%), which can be attributed to over-execution on complex steps such as `SCROLL`. Hence, Qwen-VL-MAX only achieves a *TSR* of 15.42%, while `GLM-4v-Plus` performs weakly, with only 1.67% *TSR*. In addition, three open-source GUI agents such as OS-Atlas-Pro-7B are even less effective, as they cannot adapt to OOD instructions. In contrast, *OS-Kairos* achieves the accuracies of 98.49% in *Type*, 96.36% in *SR* and 87.71% in *TSR*, respectively. Similarly, the fine-grained *Type* and *SR* outperform the baseline.

C.6.3 Generality Evaluation of *OS-Kairos*

OS-Kairos outperforms the baseline model across three datasets due to the integration of the confidence scoring. To verify the generality of adaptive interaction, we train *OS-Kairos* on each of the three datasets and then test it on the other two. The evaluation results are presented in Figure 18. We see that *OS-Kairos* is able to achieve a decent performance, though the domains vary. Compared to the main results, it significantly outperforms the baseline in the zero-shot setting across three datasets, particularly in *SR* and *TSR* metrics. Also, its generalization performance is comparable to that of models fine-tuned directly on the target dataset. We also note that the more complex the dataset on which confidence scoring is integrated, the better the generalization of *OS-Kairos*. For example, *OS-Kairos* exhibits the best generalization with confidence scoring integration on the Meta-GUI dataset (29.05% vs. 21.74% in the AITZ benchmark, and 83.85% vs. 88.20% in the OS-Kairos dataset).

C.7 Ablation of Critic Models

As the advanced judgment capabilities of `GPT-4o` (Chen et al., 2024a), we utilize it as the critic model in the collaborative probing framework. To analyze the impact of the critic model on *OS-Kairos* confidence integration and GUI adaptive interaction, we select Qwen-VL-MAX as an alternative. Table 10 presents the adaptive interaction performance of *OS-Kairos* across different critic models. The results show that the scoring quality of `GPT-4o` significantly outperforms Qwen-VL-Max, with an HSR of 86.87% compared to 57.63%. In addition, the precision of the intervention decreases by 4.59% in the autonomous steps and 9.25% in the complex steps. Although GUI performance is similar, Qwen-VL-Max leads to more frequent interventions with *OS-Kairos*.

[1] https://huggingface.co/models
[2] https://modelscope.cn/home
[3] https://github.com/hiyouga/LLaMA-Factory
[4] https://github.com/InternLM/xtuner

Models	API	SCROLL	PRESS	STOP	CLICK		TYPE		Total		TSR
					Type (%) ↑	SR (%) ↑	Type (%) ↑	SR (%) ↑	Type (%) ↑	SR (%) ↑	
GPT-4o	✓	24.17	23.84	0.00	63.80	27.71	35.20	16.00	58.32	22.69	0.00
GLM-4V-Plus	✓	11.65	7.28	0.00	79.15	27.65	43.80	20.40	68.95	20.92	0.00
Qwen-VL-MAX	✓	7.89	13.04	10.2	/	72.3	/	34.04	/	52.41	/
CogAgent	✗	56.41	48.30	4.76	79.90	51.50	67.40	34.00	65.86	44.52	/
Auto-UI	✗	74.88	49.09	60.12	44.37	12.72	73.00	67.80	73.79	34.46	/
Qwen2-VL-7B	✗	18.64	21.19	0.00	71.05	32.89	82.80	45.00	66.28	28.25	0.00
OS-Atlas-Pro-7B	✗	27.40	0.66	5.16	93.31	34.87	85.20	27.40	85.20	33.66	0.00
OS-Kairos	✗	**91.17**$_{63.77\uparrow}$	**73.51**$_{72.85\uparrow}$	**91.65**$_{86.49\uparrow}$	**98.43**$_{5.12\uparrow}$	**89.46**$_{54.59\uparrow}$	**99.20**$_{14.00\uparrow}$	**72.80**$_{45.40\uparrow}$	**96.81**$_{11.61\uparrow}$	**87.54**$_{53.88\uparrow}$	**24.51**$_{24.51\uparrow}$

Table 8: Comparison of OS-Kairos with baselines in the AITZ benchmark (zero-shot setting). We report the overall accuracy for *Type*, *SR*, and *TSR*, along with fine-grained accuracy for each action. Subscripts indicate relative improvement over the OS-Atlas-Pro-7B, with the best result highlighted in **bold**.

Models	API	SCROLL	PRESS	STOP	CLICK		TYPE		Total		TSR
					Type (%) ↑	SR (%) ↑	Type (%) ↑	SR (%) ↑	Type (%) ↑	SR (%) ↑	
GPT-4o	✓	33.97	25.00	12.79	94.12	42.30	66.47	28.14	69.85	32.72	6.67
GLM-4V-Plus	✓	0.00	0.00	1.06	95.45	26.53	38.01	22.81	65.05	17.19	1.67
Qwen-VL-MAX	✓	14.74	40.91	1.91	70.87	37.85	74.85	45.03	54.74	27.86	15.42
Auto-UI	✗	60.90	0.00	0.00	26.90	2.69	0.00	0.00	20.02	6.45	0.00
Qwen2-VL-7B	✗	0.00	0.00	0.43	51.54	2.33	38.01	18.13	35.90	3.04	0.21
OS-Atlas-Pro-7B	✗	16.03	0.00	0.00	94.53	37.29	60.23	15.20	66.09	23.59	0.42
OS-Kairos	✗	**99.36**$_{83.33\uparrow}$	**100.00**$_{100.00\uparrow}$	**94.73**$_{94.73\uparrow}$	**99.81**$_{5.28\uparrow}$	**96.66**$_{59.37\uparrow}$	**98.83**$_{38.60\uparrow}$	**95.32**$_{80.12\uparrow}$	**98.49**$_{32.40\uparrow}$	**96.36**$_{72.77\uparrow}$	**87.71**$_{87.29\uparrow}$

Table 9: Comparison of OS-Kairos with baselines in the Meta-GUI benchmark (zero-shot setting). We report the overall accuracy for *Type*, *SR*, and *TSR*, along with fine-grained accuracy for each action. Subscripts indicate relative improvement over the OS-Atlas-Pro-7B, with the best result highlighted in **bold**.

C.8 Ablation of Adaptive Interaction

To understand the advantages of adaptive integration in *OS-Kairos*, we compare its performance with and without adaptive integration: when treated as regression optimization or classification optimization in complex scenarios datasets. As shown in Table 11, we see that *OS-Kairos* without adaptive interaction is quite accurate at *HSR* of 86.99%, but its overall performance and intervention precision are suboptimal. For example, the *TSR* is 82.61%, *IP* is 70.66%, and *AP* is 95.84%. The results show that the adaptive interaction does not significantly affect the performance of *OS-Kairos*. In contrast, *OS-Kairos* has the advantage of adaptive interaction by tuning the threshold, which balances the sensitivity between autonomous and human intervention.

D Case study

To further illustrate the execution process of *OS-Kairos*, we present four examples from three complex scenarios, along with two examples from the benchmark datasets. First, for simple instructions, *OS-Kairos* can be fully autonomous, as shown in Figure 19. Second, for complex instructions across the three scenarios, *OS-Kairos* adaptively identifies the complex steps requiring human intervention, while automating other steps, as shown in Figure 20, Figure 21 and Figure 22. Similarly, *OS-Kairos* performs effectively on the AITZ benchmark (Figure 23). In an extreme case, *OS-Kairos* requests human intervention at nearly every step to complete the task, as the Meta-GUI benchmark represents an OOD scenario for OS-Atlas-Pro-7B, as shown in Figure 24.

Models	**Type (%)↑**	**SR (%)↑**	**TSR (%)↑**	**HSR (%)↑**	**IP(%)↑**	**AP(%)↑**
GPT-4o	98.49	**96.36**	**87.71**	**86.87**	**70.75**	**96.44**
Qwen-VL-MAX	**99.65**	96.01	85.71	57.63	61.50	91.55

Table 10: Ablation of critic models.

Models	**Type (%)↑**	**SR (%)↑**	**TSR (%)↑**	**HSR (%)↑**	**IP(%)↑**	**AP(%)↑**
OS-Kairos	**99.88**	**95.90**	**88.20**	86.87	**70.75**	**96.44**
OS-Kairos$_{\text{w/o adaptive interaction}}$	99.53	95.31	82.61	**86.99**	70.66	95.84

Table 11: Ablation of adaptive interaction.

Action Prompt Template for Probing GUI Agent

```
"You are now operating in Executable Language Grounding mode. Your goal is to help users accomplish tasks by suggesting executable actions that best fit their needs. Your skill set includes both basic and custom actions:\n"
"1. Basic Actions\n"
"Basic actions are standardized and available across all platforms. They provide essential functionality and are defined with a specific format, ensuring consistency and reliability.\n"

"Basic Action 1: CLICK\n"
    "- purpose: Click at the specified position.\n"
    "- format: CLICK <point>[[x-axis, y-axis]]</point>\n"
    "- example usage: CLICK <point>[[101, 872]]</point>\n"

"Basic Action 2: TYPE\n"
    "- purpose: Enter specified text at the designated location.\n"
    "- format: TYPE [input text] \n"
    "- example usage: TYPE [Shanghai shopping mall] \n"

"Basic Action 3: SCROLL\n"
    "- Purpose: SCROLL in the specified direction.\n"
    "- Format: SCROLL [direction (UP/DOWN/LEFT/RIGHT)] \n"
    "- Example Usage: SCROLL [UP]\n"

"2. Custom Actions\n"
"Custom actions are unique to each user's platform and environment. They allow for flexibility and adaptability, enabling the model to support new and unseen actions defined by users. These actions extend the functionality of the basic set, making the model more versatile and capable of handling specific tasks. \n"

"Custom Action 1: PRESS_BACK\n"
    "- purpose: Press a back button to navigate to the previous screen.\n"
    "- format: PRESS_BACK\n"
    "- example usage: PRESS_BACK\n"

"Custom Action 2: PRESS_HOME\n"
    "- purpose: Press a home button to navigate to the home page.\n"
    "- format: PRESS_HOME\n"
    "- example usage: PRESS_HOME\n"

"Custom Action 3: COMPLETE\n"
    "- purpose: Indicate the task is finished.\n"
    "- format: COMPLETE\n"
    "- example usage: COMPLETE\n"

Custom Action 4: IMPOSSIBLE
    "- purpose: Indicate the task is impossible.\n"
    "- format: IMPOSSIBLE\n"
    "- example usage: IMPOSSIBLE\n"

"In most cases, task instructions are high-level and abstract. Carefully read the instruction and action history, then perform reasoning to determine the most appropriate next action.\n"

"And your previous actions, current task instruction, step list and associated screenshot are as follows:\n"

f"Final goal: {obs['task']}\n"
f"current goal: {obs['list'][obs['now_step']]}\n"
f"step list: {obs['list']}\n"
f"previous actions: {obs['previous_actions']}\n"
f"Screenshot: \n"
```

Figure 14: Prompt of the probed GUI agent for generating action.

Action Prompt Template for Critic Model

```
"### Background ###\n"
"You are an expert in completing tasks based on screenshots and instructions. Based on the mobile screenshot, the final goal, the current goal and the step list. I need you to determine the action to take. The Current Goal may not be accurate, but the correct Current Goal must be one of the steps in the step list. If you feel that the Current Goal is not accurate, please use the step list to determine the appropriate Current Goal to execute.\n"

f"Final Goal: {final_goal}\n"
f"Current Goal: {current_goal}\n"
f"previous actions : {previous_actions}"
f"step list: {step_list}\n"

"### Screenshot information ###\n"
"To help you understand the information in the screenshot, I first performed OCR. Here are the names and coordinates of the icons obtained through OCR:\n"
f"Coordinates of the icons: {ocr}\n"

"### Response requirements ###\n"
"Your skill set includes both basic and custom actions:\n"

"Basic Action 1: CLICK\n"
    "- purpose: Click at the specified position.\n"
    "- format: CLICK <point>[[x-axis, y-axis]]</point>\n"
    "- example usage: CLICK <point>[[101, 872]]</point>\n"

"Basic Action 2: TYPE\n"
    "- purpose: Enter specified text at the designated location.\n"
    "- format: TYPE [input text] \n"
    "- example usage: TYPE [Shanghai shopping mall] \n"

"Basic Action 3: SCROLL\n"
    "- Purpose: SCROLL in the specified direction.\n"
    "- Format: SCROLL [direction (UP/DOWN/LEFT/RIGHT)] \n"
    "- Example Usage: SCROLL [UP]\n"

"2. Custom Actions\n"
"Custom actions are unique to each user's platform and environment. They allow for flexibility and adaptability, enabling the model to support new and unseen actions defined by users. These actions extend the functionality of the basic set, making the model more versatile and capable of handling specific tasks. \n"

"Custom Action 1: PRESS_BACK\n"
    "- purpose: Press a back button to navigate to the previous screen.\n"
    "- format: PRESS_BACK\n"
    "- example usage: PRESS_BACK\n"

"Custom Action 2: PRESS_HOME\n"
    "- purpose: Press a home button to navigate to the home page.\n"
    "- format: PRESS_HOME\n"
    "- example usage: PRESS_HOME\n"

"Custom Action 3: COMPLETE\n"
    "- purpose: Indicate the task is finished.\n"
    "- format: COMPLETE\n"
    "- example usage: COMPLETE\n"

Custom Action 4: IMPOSSIBLE
    "- purpose: Indicate the task is impossible.\n"
    "- format: IMPOSSIBLE\n"
    "- example usage: IMPOSSIBLE\n"

"### Output format ###\n"
"Your response must exactly follow the template:\n"
"{action: ACTION_NAME}\n"
"Replace `ACTION_NAME` with one of:\n"
"- CLICK <point>[[x,y]]</point>\n"
"- TYPE [input text]\n"
"- SCROLL [UP/DOWN/LEFT/RIGHT]\n"
"- PRESS_BACK\n"
"- PRESS_HOME\n"
"- ENTER\n"
"- IMPOSSIBLE\n"
```

Figure 15: Prompt of the critic model for generating action.

Scoring Prompt Template

"### Background ###\n"
"You are an expert in completing tasks based on screenshots and instructions. You will grade the student action based on the goal, screenshot, and teacher action. I hope you can be a bit stricter in your scoring. I will provide you with a mobile screenshot, a final goal, the current goal, the previous actions,a student action and a teacher action. I hope you evaluate this student action based on the screenshot , the teacher action and the goal, giving it a score from 1 to 5. \n"

"The teacher action is an example you consider worthy of a full score (5 points). If you believe the student action does not achieve the same level of performance, points should be deducted accordingly. Pay special attention to cases involving coordinates; significant discrepancies in coordinates must result in point deductions.\n"

f"Final goal：{final_goal}\n"
f"current goal：{current_goal}\n"
f"student action:{osatlas_action}\n"
f"previous actions : {previous_actions}"
f"teacher action:{teacher_action}\n"

"### Screenshot information ###\n"
"To help you understand the information in the screenshot, I first performed OCR. Here are the names and coordinates of the icons obtained through OCR:"
f"Coordinates of the icons: {ocr}"

"### Response requirements ###\n"
"I hope you evaluate this action based on the screenshot and the goal, giving it a score from 1 to 5.\n"

"A higher score indicates that you believe this action is more likely to accomplish the current goal for the given screenshot.\n"

"1 means you believe this action definitely cannot achieve the goal.\n"
"2 means you believe this action is very unlikely to achieve the goal.\n"
"3 means you believe this action has a certain chance of achieving the goal.\n"
"4 means you believe this action is very likely to achieve the goal.\n"
"5 means you believe this action will definitely achieve the goal.\n"

"If the teacher action and student action are of different types, the score should only be between 1 and 3 points.\n"

"If both the teacher action and student action are CLICK, a full score of 5 points can be given if the coordinate difference is minimal. However, if the coordinate difference is significant, points must be deducted.\n"

"### Output format ###\n"
"Your output must strictly follow the format below:\n"
"{score: }"

Figure 16: Prompt of the critic model for generating action score.

Completion Judgment Prompt Template

"You are an expert in completing tasks based on screenshots and instructions.\n"

"I am now providing you with a screenshot of the previous state, a screenshot of the current state, and the overall goal.\n"

"You should be able to tell from the screenshot and the list of steps what step you are currently at.\n"

f"The overall goal is: {current_task}\n"
"Please determine whether the overall goal has been achieved based on the overall goal and the screenshots. If you believe it has been achieved, output 1. If you believe it has not been achieved, output 0.\n"

"Your output must strictly follow the format below:\n"
"{Is_final_finished: 0} or {Is_final_finished: 1}"

Figure 17: Prompt for the critic model to judge instruction completion.

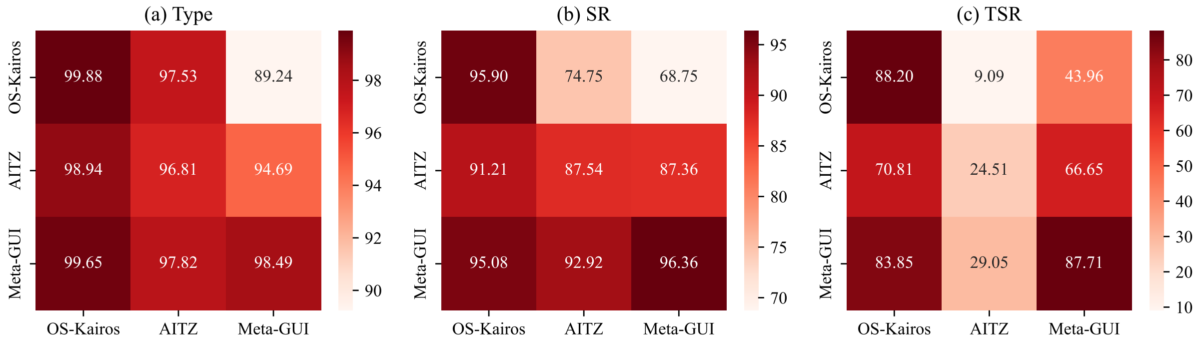


Figure 18: Generality analysis of *OS-Kairos* for adaptive interaction from original dataset to target datasets.

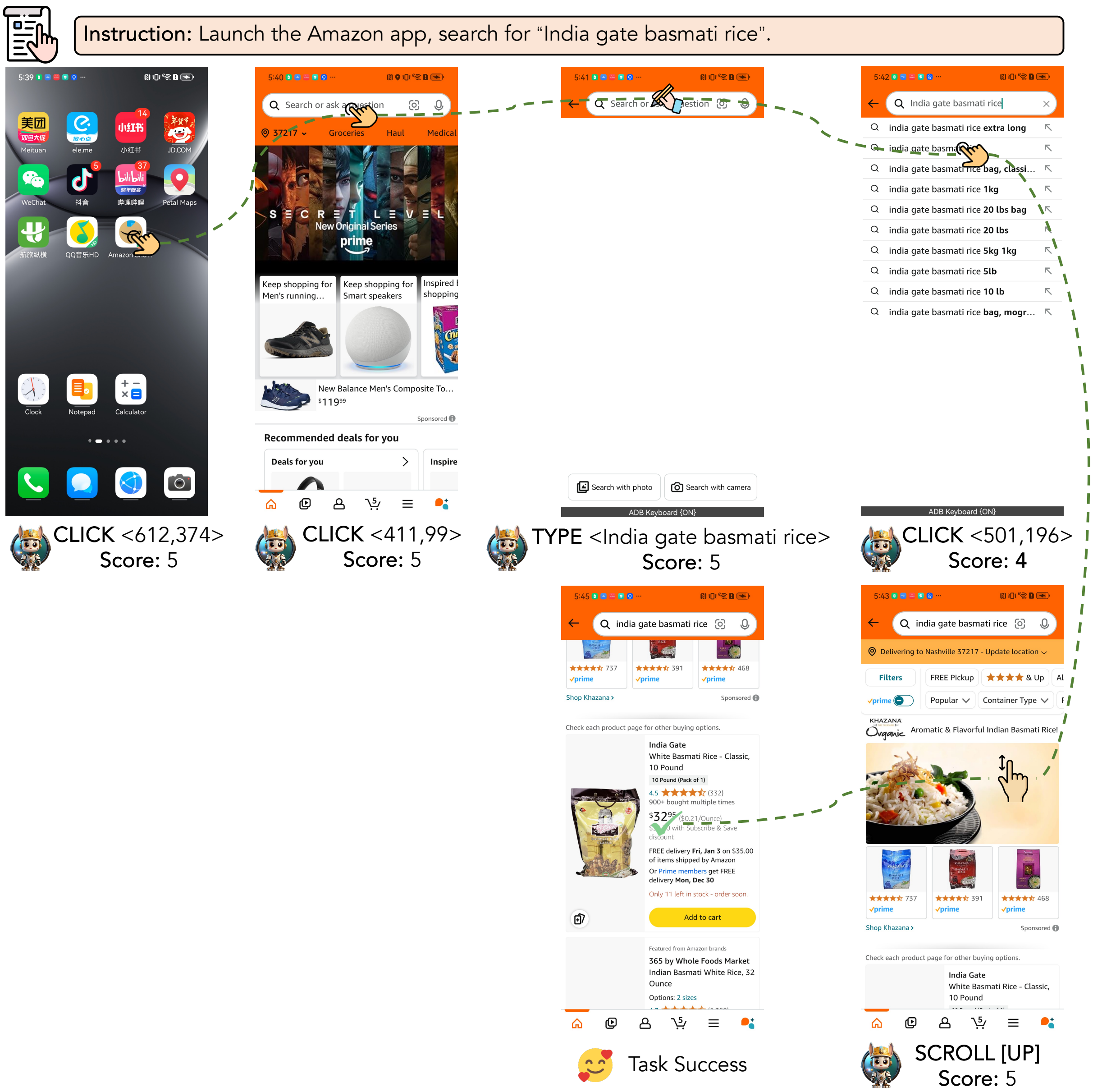


Figure 19: Case study of *OS-Kairos* in the normal scenario. At each step, *OS-Kairos* outputs both the action and the confidence score. If the score falls below a specified threshold, human intervention is initiated to ensure task success.

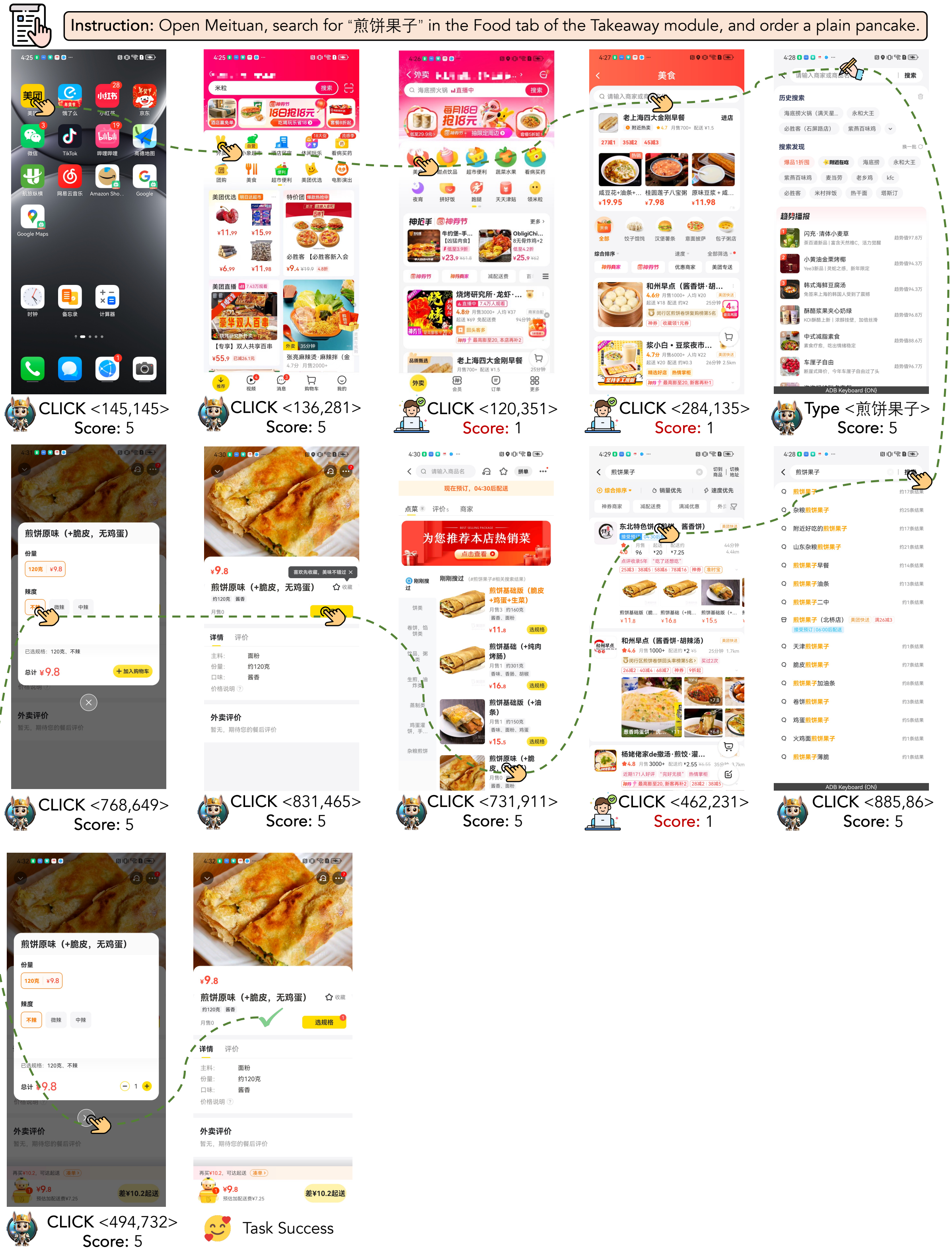


Figure 20: Case study of *OS-Kairos* in Scenario 1 (capability bottleneck). At each step, *OS-Kairos* outputs both the action and the confidence score. If the score falls below a specified threshold, human intervention is initiated to ensure task success.

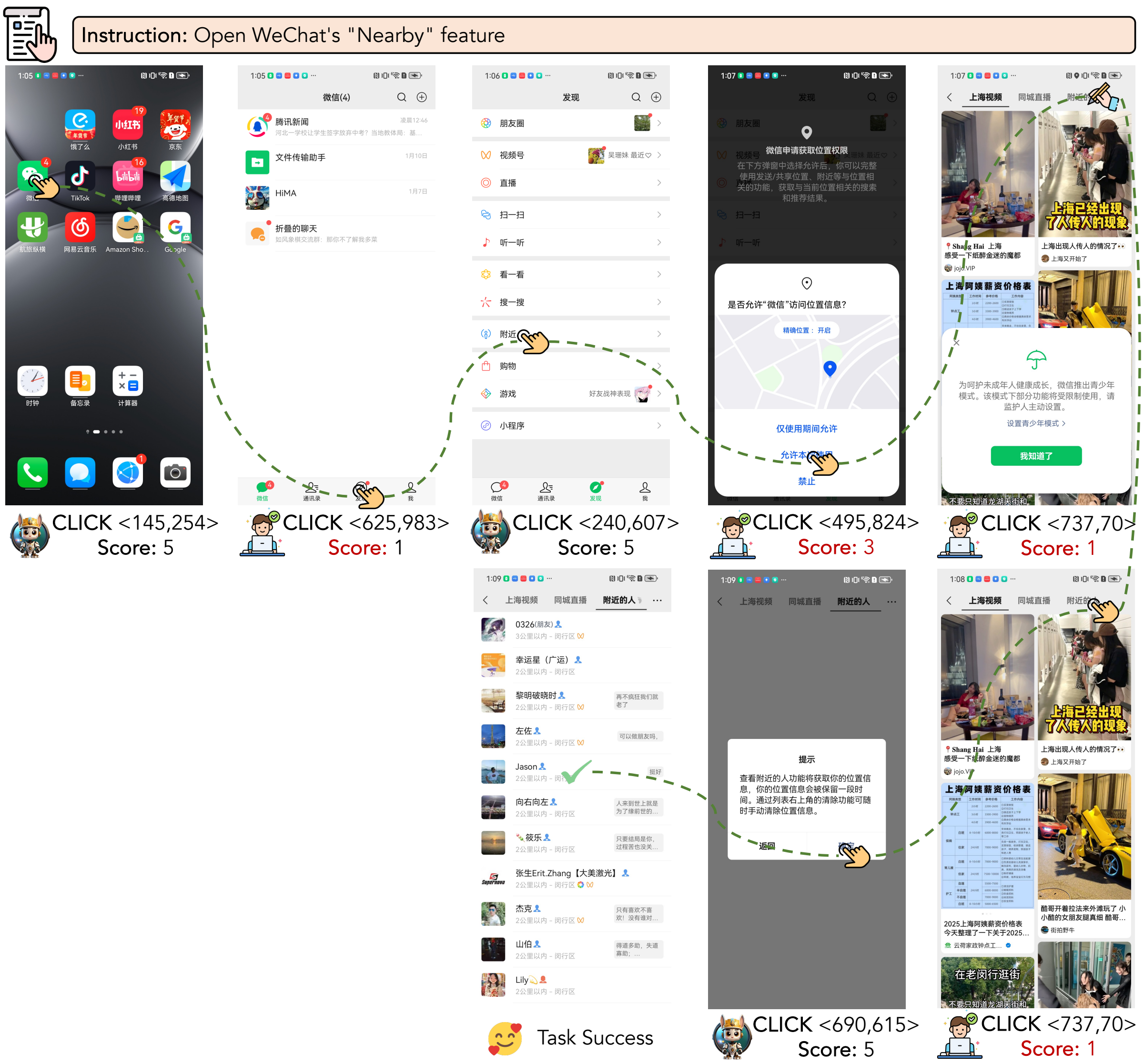


Figure 21: Case study of *OS-Kairos* in Scenario 2 (no location permission). At each step, *OS-Kairos* outputs both the action and the confidence score. If the score falls below a specified threshold, human intervention is initiated to ensure task success.

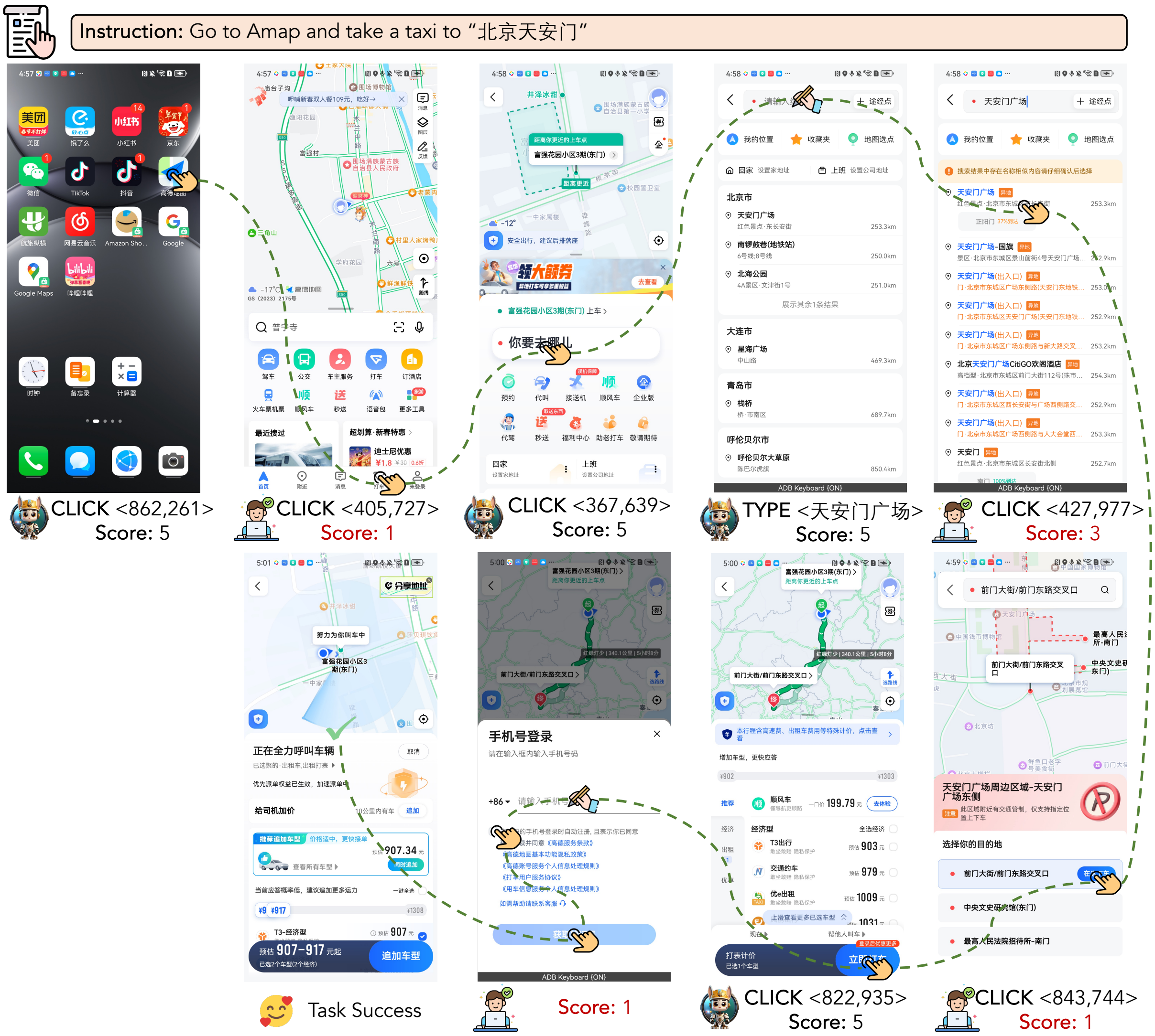


Figure 22: Case study of *OS-Kairos* in Scenario 3 (information absence). At each step, *OS-Kairos* outputs both the action and the confidence score. If the score falls below a specified threshold, human intervention is initiated to ensure task success.

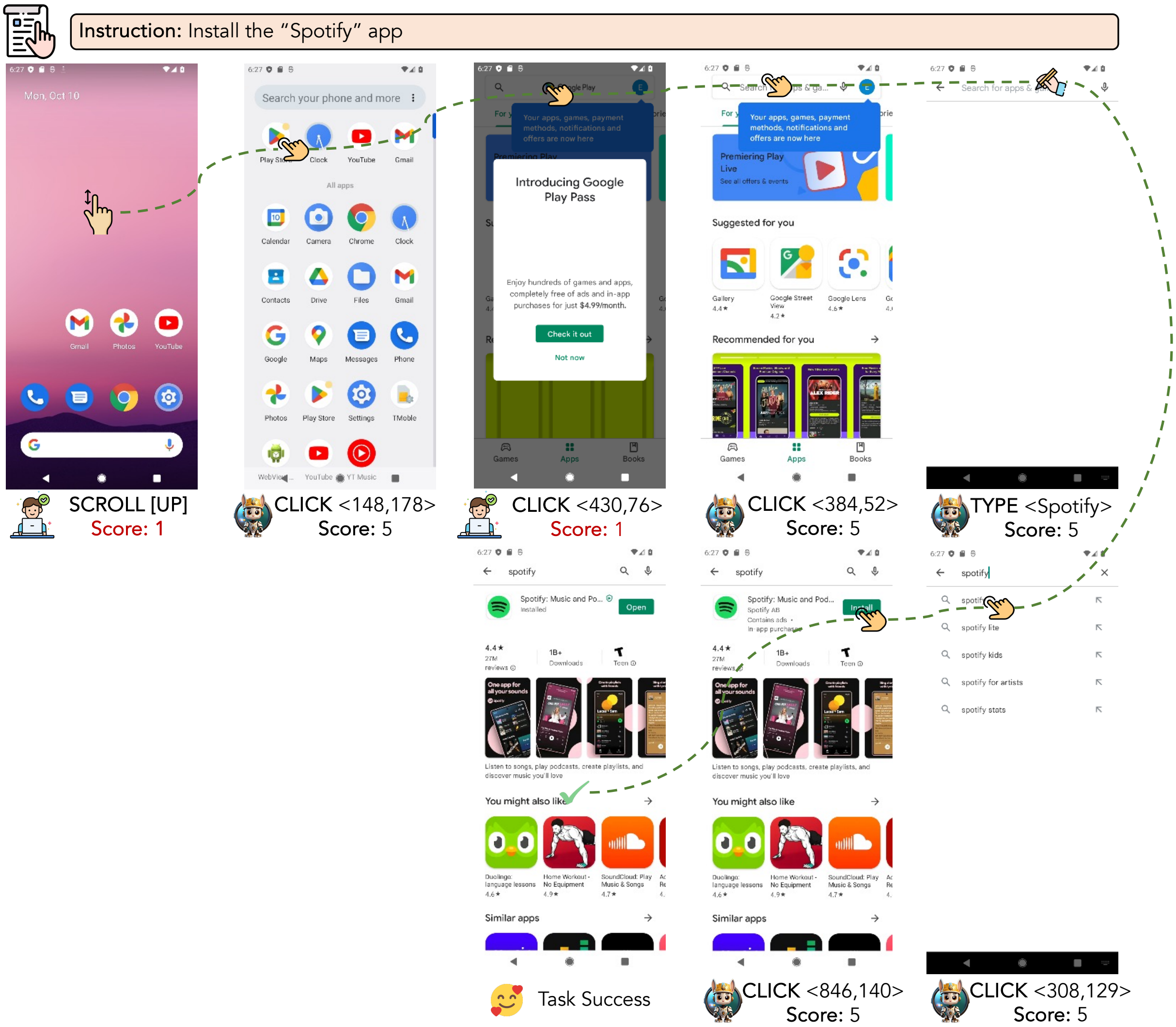


Figure 23: Case study of *OS-Kairos* in the AITZ benchmark. At each step, *OS-Kairos* outputs both the action and the confidence score. If the score falls below a specified threshold, human intervention is triggered to ensure task success.

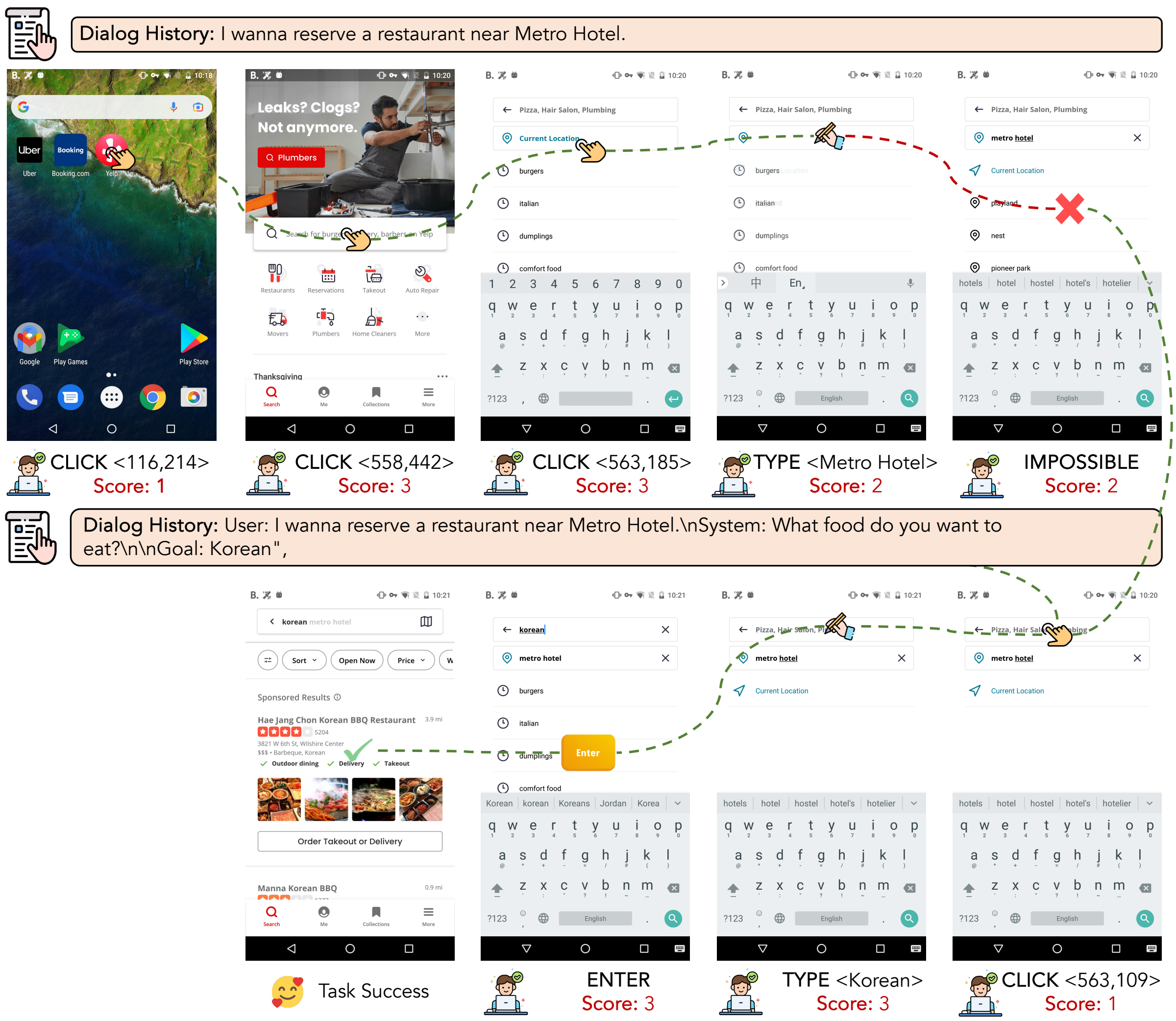


Figure 24: Case study of *OS-Kairos* in the Meta-GUI benchmark. At each step, *OS-Kairos* outputs both the action and the confidence score. If the score falls below a specified threshold, human intervention is triggered to ensure task success.