ScaleTrack: Scaling and Backtracking Automated GUI Agents

Anonymous ACL submission

Abstract

Automated GUI agents aim to streamline user interaction by automatically executing complex tasks across digital environments, such as web, mobile, and desktop devices. Given a textual task instruction and a GUI description, the agent generates a sequence of executable actions (*e.g.*, clicks) and operation steps. Training a GUI agent typically involves grounding and planning stages, where the GUI grounding stage focuses on locating the executable interface coordinates based on the given task, while the planning stage aims to predict the next action using the history of previous actions. However, previous work suffers from insufficient training data for GUI grounding and overlooks the importance of backtracking historical behaviors in GUI planning. To handle the above challenges, we propose ScaleTrack, a training framework that integrates scalable grounding and backtracking planning for automated GUI agents. Specifically, we systematically collected GUI samples from a wide range of sources, with each source employing distinct synthesis criteria, and unified them into a standardized template for training GUI grounding models. Moreover, ScaleTrack introduces a novel training strategy that predicts the next action based on the current GUI image while simultaneously backtracking the historical actions that led to it. This approach enables ScaleTrack to effectively capture the correspondence between GUI states and actions, modeling the dynamic evolution of the GUI environment. Extensive experimental results on grounding tasks, as well as both offline and online agent evaluations, demonstrate the effectiveness of ScaleTrack.

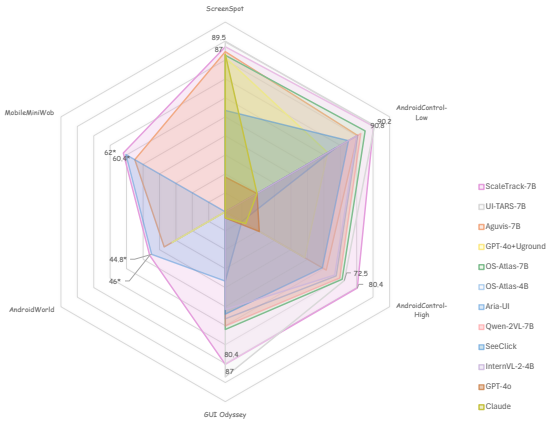

Figure 1: Performance of ScaleTrack compared with previous state-of-the-art methods. The online benchmarks (AndroidWorld and MobileMiniWob) use GPT-4 as the planner.

1 Introduction

The development of native automated agents for Graphical User Interfaces (GUI), known as GUI agents, is driven by the increasing demand to automate complex user tasks in digital environments, especially on mobile and desktop platforms. Their ability to perform tasks accurately and enhance user interaction has garnered widespread interest. A key driver for this progress is the advancement of large language models (LLMs) (Team et al., 2024; Achiam et al., 2023), whose emerging reasoning capabilities allow GUI agents to effectively plan and execute multi-step actions from task instructions.

Early GUI agents (Kim et al., 2023; Zheng et al., 2023) extract structured representations of the GUI environment (*e.g.*, website HTML), and then leverage LLMs to perform reasoning and planning for generating executable actions. In fact, such structured textual representations are often lengthy, not readily accessible, and lack critical attribute information such as spatial layout and element locations. To address these limitations, recent research (Cheng et al., 2024; Wu et al., 2024) explores visual GUI agents based on multimodal large language models (MLLMs), which interact with users using only GUI screenshots.

To train GUI agents using visual screenshots, existing methods (Xu et al., 2024; Qin et al., 2025) typically transfer general multimodal knowledge from MLLMs (*e.g.*, QWen2-VL) to GUI environments. Previous work often fine-tunes MLLMs on two types of GUI data: GUI grounding, which involves predicting coordinate positions based on task instructions, and GUI planning, which involves predicting multi-step executable actions to complete tasks. The former focuses on spatial localization, while the latter aims at sequential decision-making for automated task execution.

Regarding the synthesis of GUI grounding data, existing methods primarily synthesize training data using isolated criteria, leveraging extensive metadata, such as texts, icons, and widgets, across mobile, web, and desktop platforms, typically with the aid of a powerful large language model (*e.g.*, GPT-4o). For instance, Uground (Gou et al., 2024) synthesizes grounding data with multiple reference modes; Aria-UI (Yang et al., 2024) generates context-aware grounding data; and Augvis (Xu et al., 2024) leverages template-enhanced synthesis. However, these approaches often overlook the complementary strengths of different synthesis methods, thereby limiting the scalability and effectiveness of GUI grounding training.

As for GUI planning, existing research only focuses on forward-planning, predicting the next action based on task instructions and the current GUI state. This is typically achieved either through human-annotated action trajectories for various tasks (Deng et al., 2023), or by prompting MLLMs to produce detailed reasoning traces (Xu et al., 2024). However, GUI environments exhibit inherent task-driven patterns that support not only forward planning but also back-tracing, the reconstruction of historical actions leading to the current state. Despite its potential, this backward dynamic remains largely underexplored.

To handle the above challenges, we introduce ScaleTrack, a novel training framework that scales GUI grounding and incorporates backtracking into GUI planning. We enhance GUI grounding through several data-driven approaches, including element referring, context awareness, and functional descriptions. By unifying diverse grounding samples synthesized under distinct criteria, our method enables scalable training within a consistent training template. Moreover, we introduce a data template that annotates actions across consecutive GUI images to facilitate planning. Specifically, our method captures both next-actions for forward planning and historical actions for backtracking (see Figure 2). Finally, we devise a hybrid training strategy to jointly learn forward planning and backtracking, allowing GUI agents to predict both upcoming and previous actions based on the current state. Empirical results demonstrate that incorporating backtracking significantly enhances task execution performance.

The main contributions of this work are as follows:

- We propose the first GUI agent with backtracking capability, and devise an effective data construction as well as training strategy.
- We integrate several data synthesis criteria to enrich GUI element descriptions, significantly scaling the training process that leads to consistent performance improvements.
- We conduct extensive experiments on several benchmark datasets, including grounding evaluation, offline and online evaluation, and the experimental results verify the effectiveness of our proposed ScaleTrack.

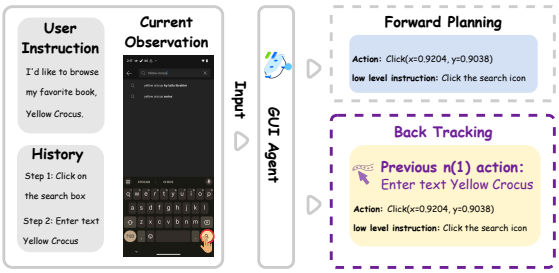

Figure 2: Comparison between forward-planning and backtracking in a GUI agent task. The top section illustrates forward planning, where the agent predicts the next action based on the current state (e.g., viewing search results). The bottom section demonstrates backtracking, where the agent infers a sequence of previous actions (e.g., search box interaction and query entry) that led to the current state.

2 Related Works

2.1 Multimodal LLMs

Recent advances in closed-source and open-source multimodal large-language models (MLLMs) have significantly enhanced non-natural image understanding and GUI task planning. Closed-source models like GPT-4V (OpenAI, 2023) and

GPT-4o (Hurst et al., 2024), with strong visual and task-planning abilities, often serve as the "brain" in planning and grounding decoupled GUI agent framework. Open-source models have enabled domain-specific capability refinement for GUI agents. The Qwen-VL (Bai et al., 2023; Wang et al., 2024a; Bai et al., 2025) series distinguishes itself through fine-grained visual comprehension and multimodal capabilities, with Qwen2-VL (Wang et al., 2024a) notably serving as the foundational MLLM backbone for the majority of previous GUI agent implementations. Other open-source models have distinct technical advantages in the GUI agent field. InternVL-2 (Chen et al., 2024b) improves multimodal task performance through progressive alignment. CogVLM (Wang et al., 2024b) uses a visual expert module to fuse vision and language features. Ferret (You et al., 2023) boosts human-computer interaction precision with enhanced spatial comprehension. The LLAVA (Liu et al., 2023, 2024; Li et al., 2024a) series is used for its lightweight projection layer, enabling fast training and multimodal understanding.

2.2 GUI Agents

Recent GUI agents, predominantly built on MLLMs, can execute autonomous operations on phones or computers as per human instructions, applicable in web, desktop, and mobile scenarios. Their operation involves planning and grounding phases. Based on whether these two phases are handled by separate models, existing GUI agent works can be categorized into pure GUI grounding models like Aria-UI (Yang et al., 2024) and Uground (Gou et al., 2024), and unified GUI agent frameworks such as Aguvis (Xu et al., 2024), OS-ATLAS (Wu et al., 2024) and UI-TARS (Qin et al., 2025).

In terms of perceptual content, early works such as WebPilot (Zhang et al., 2025), Hybrid Agent (Song et al., 2024), WebDreamer (Gu et al., 2024), Agent-Q (Putta et al., 2024), LASER (Ma et al., 2023), and SeeAct (Zheng et al., 2024) concentrated on web platforms to automate web interaction and navigation. They incorporated structured text (*e.g.*, accessibility trees, HTML-DOM) with environmental visual structures (screenshots), establishing a foundation in the GUI agent field. However, the difficulty of accessing structured text in real-world settings like desktop and iOS applications, along with its token inefficiency impacting MLLM performance, has driven a shift toward vision-only approaches. Claude 3.5 Sonnet (Computer Use) (Hu et al., 2024)pioneered this paradigm in desktop task automation, integrating task planning and environmental interaction action prediction into a single system.

The rise of vision-only methods has also sparked cross-platform unified modeling. Recent methods like Aguvis (Xu et al., 2024), UI-TARS (Qin et al., 2025), and OS-ATLAS (Wu et al., 2024) use a uniform action space and are trained on multi-platform datasets. Aguvis integrates explicit planning and reasoning into its framework, enhancing interaction with complex digital environments. OS-ATLAS proposes a unified action space for standardized cross-platform datasets, covering basic and custom actions. UI-TARS apply diverse reasoning patterns in the model's planning phase, including task decomposition, reflection, and milestone recognition.

3 Method

ScaleTrack follows the paradigm of a two-stage training framework, including GUI grounding and GUI planning. It learns GUI grounding capability based on a basic vision-language model (*i.e.* Qwen25-VL), and then continues to learn GUI planning capability through trajectory data.

3.1 Task Formulation

Given a task description T and the initial environment observation o_1, the autonomous GUI carries out planning and predicts an action that belongs to the action space, namely $a_1 \in \mathbb{A}$. Subsequently, the client-side environment is updated upon receiving this action and provides a new observation o_2. The above process is repeated continuously until the predicted action is *terminate*, which signifies the completion of the task. The entire process can be formatted as:

$$a_n \leftarrow (T, (o_1, a_1), ..., (o_{n-1}, a_{n-1}), o_n), \quad (1)$$

here the task description T is textual input, and the observation o is visual input.

3.2 GUI Grounding

GUI Grounding aims to maximize the training scale through format unification, cross-platform generalization, and multi-round operation, providing a sufficient training foundation for the next stage of GUI planning.

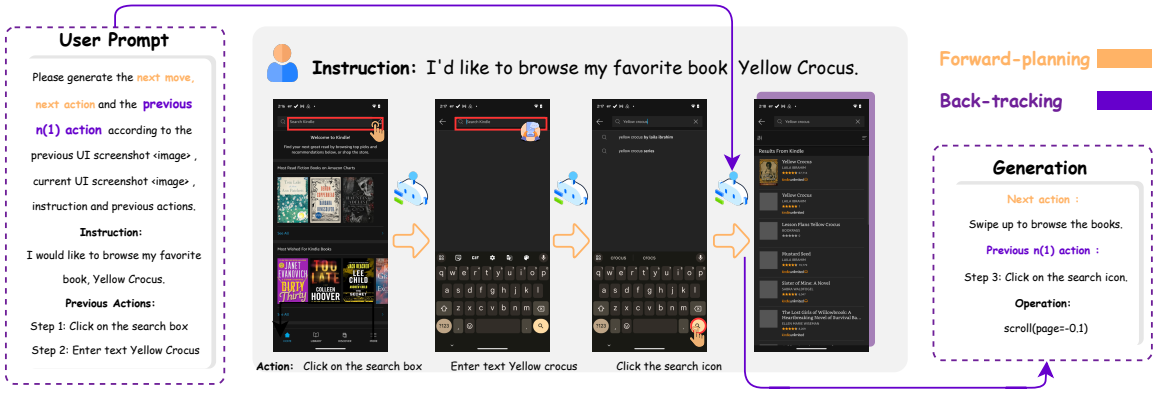

Figure 3: Overall description of our proposed ScaleTrack in processing task instruction and generating actions via forward-planning and back-tracking, as well as the format of training data.

3.2.1 Format Unification

How to represent the spatial information of texts, icons and controls in GUI screenshots is the primary issue that GUI agents need to address. We use absolute coordinates to train ScaleTrack, which helps the model build a sense of the real world.

In the previous GUI grounding dataset annotation, the coordinates of the target element are typically in the form of a box: $(x1, y1, x2, y2)$, where $(x1, y1)$ and $(x2, y2)$ indicate the coordinates of the top-left and bottom-right corners of the smallest bounding box enclosing the element. However, in many GUI agent operations, click-based interaction is more prevalent. For instance, when users click buttons or links on the GUI, it is essentially a point operation. Data in the form of boxes can help the agent have a better understanding of the boundaries and shapes of UI elements, and data in the form of points is helpful for more sophisticated operations such as clicking, so we combine data in these two formats for mixed training.

3.2.2 Cross-platform Generalization

However, compared with the large amount of general image data accumulated in the field of computer vision, the grounding data involved in the GUI agents field still has great limitations: data from different platforms and devices is difficult to generalize, and different tasks use isolated data synthesis criteria, which makes it difficult to complement each other.

In order to solve the generalization problem in GUI grounding, we fused data from different sources and with different synthesis criteria in the field of GUI agents. Specifically, following Uground (Gou et al., 2024), we introduced a large amount of website grounding data constructed by integrating multiple reference modes. Following Aria-UI (Yang et al., 2024), we introduced context-aware grounding data constructed by integrating context perception. Inspired by Aguvis (Xu et al., 2024), we introduced template-enhanced grounding data constructed by a unified action space. As shown in Table 1, our work unifies these data and studies whether different data synthesis criteria can complement each other, thereby improving the generalization ability of agent positioning from all aspects.

Data Source	Data Type	Grounding		MultiStep	
		Elements	Screenshots	Trace	avg steps
Uground	Web	9M	773K	/	/
OS-Atlas	Web	7.8M	1.6M	/	/
	Mobile	1.1M	107K	/	/
	Desktop	11.3M	54K	/	/
Aria-UI	Web	-	173K	/	/
	Mobile	-	104K	/	/
	Desktop	-	7.8K	/	/
Aguvis	Web	723K	-	6.3k	6.7
	Mobile	232K	-	28.7K	8.5
	Desktop	7K	-	/	/
Ours	Ours	*	7.5M	*	8.2

Table 1: Statistics of the open-source grounding and planning data.

3.2.3 Multi-round Operation

In real-world scenarios, a complex screenshot may contain hundreds of UI elements, and existing open-source grounding datasets naturally include multiple objects within a single image. To avoid redundant operations of repeatedly loading images and to reduce training overhead, we merged different instruction/description-answer pairs from the

same screenshot into a single conversation, thus creating multi-turn training data. Training with multi-round data not only reduces training computational overhead but also enhances the model's understanding of different elements in the UI scene.

3.2.4 Training Paradigm

We train from a vision-language model VLM to predict the grounding coordinates related to the task. It receives the user task T, the observation o to predict a grounding coordinate $\mathcal{T} =< x1, y1, x2, y2 >$, which can be formulated as:

$$\mathcal{T} = VLM_\theta(T, o), \tag{2}$$

The outputs are optimized using a generative loss in an auto-regressive manner:

$$\mathcal{L}_{\text{grounding}}(\theta) = -\log p\left(\mathcal{T} \mid \mathcal{Y}_g; \theta\right), \tag{3}$$

where $\mathcal{Y}_g$ indicates the ground-truth coordinate tokens, and θ indicates the trainable parameters.

3.3 GUI Planning

GUI planning captures both next actions for forward planning and historical actions for backtracking, generating an action sequence to complete the task.

3.3.1 Backtracking Planning

As mentioned in the previous Section, the training of GUI agents' planning stage is usually modeled as a partially observable Markov decision process. It provides the model with past behaviors and state perceptions, only requiring the model to perform forward-planning and predict the probabilities of future actions based on these. However, this approach overlooks the model's ability to reflect on and backtrack historical decisions. That is, the model is aware of the state it has reached but is unaware of how it arrived there. As a result, the model is unable to establish connections between the past, present, and future, making it difficult to generate consistent action predictions.

To address this limitation, we extend the interaction between GUI agents and their environment by incorporating backtracking. Specifically, at each time step t, ScaleTrack not only predicts the next action under the current overall goal but also predicts the historical actions that led to the current state. This can be formulated as follows:

$$a_{n-1}, a_n \leftarrow (T, (o_1, a_1), ..., (o_{n-1}), o_n) \tag{4}$$

In the forward-planning aspect, similar to traditional methods, ScaleTrack takes the current state and task instructions as input and generates the next action probability distribution. This enables the agents to determine the most likely next action to perform in the current state, thereby achieving step-by-step task execution.

In terms of backtracking, ScaleTrack introduces a reverse prediction mechanism. Based on the current state and instructions, it predicts the actions that might have occurred before the current state. By doing so, the agents can gain a clearer understanding of the path they took to reach the current state, enabling them to assess the rationality of previous actions and adjust the subsequent planning accordingly.

3.3.2 Chain-of-Thought

To enhance the model's reasoning ability when dealing with complex tasks and provide interpretability of the model output, following (Xu et al., 2024), we retain the thought process during future trajectory prediction. Specifically, in addition to predicting actions, ScaleTrack also predicts the rationale behind each action. The model's output format is as follows:

```
<think> Thought </think>
<tool_call> Actions </tool_call>
```

3.3.3 Training Paradigm

We train from the grounding model to predict a_n and a_{n-1} for backtracking planning in the n-th step. Specifically, for the n-th step, it receives the user task T, the observation o_n and the historical actions $\{h_0, h_1, ..., h_{n-2}\}$ to predict actions $\mathcal{A} =< a_{n-1}, a_n >$, which can be formulated as:

$$\mathcal{A} = VLM_\theta(T, o_n, \{h_0, h_1, ..., h_{n-2}\}), \tag{5}$$

Similarly, the outputs are optimized using a generative loss in an auto-regressive manner:

$$\mathcal{L}_{\text{planning}}(\theta) = -\log p\left(\mathcal{A} \mid \mathcal{Y}_p; \theta\right), \tag{6}$$

where $\mathcal{Y}_p$ indicates the ground-truth tokens, and θ indicates the same trainable parameters as in the grounding stage.

4 Experiments

In this section, we conduct experiments on GUI Grounding Evaluation and Offline/Online

Method		Data Source	Mobile		Desktop		Web		Avg
			Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
Agent Framework									
GPT-4o	SeeClick	Public	81.0	59.6	69.6	33.6	43.9	26.2	52.3
	UGround	Public	93.4	76.9	92.8	67.9	88.7	68.9	81.4
Agent Model									
GPT-4o		In-house	20.2	24.9	21.1	23.6	12.2	7.8	18.3
Claude		In-house	-	-	-	-	-	-	83.0
Gemini 2.0		In-house	-	-	-	-	-	-	84.0
UI-TARS		In-house	94.5	85.2	95.9	85.7	90.0	83.5	89.5
UI-TARS-72B		In-house	94.9	82.5	89.7	88.6	88.7	85.0	88.4
Qwen2.5-VL-7B		In-house	-	-	-	-	-	-	84.7
CogAgent		Public	67.0	24.0	74.2	20.0	70.4	28.6	47.4
SeeClick		Public	78.0	52.0	72.2	30.0	55.7	32.5	53.4
UGround		Public	82.8	60.3	82.5	63.6	80.4	70.4	73.3
Aria-UI		Public	92.3	73.8	93.3	64.3	86.5	76.2	82.4
OS-Atlas		Public	93.0	72.9	91.8	62.9	**90.9**	74.3	82.5
Aguvis		Public	**95.6**	77.7	**93.8**	67.1	88.3	75.2	84.4
ScaleTrack		Public	92.3	**85.6**	93.3	**75.0**	90.0	**80.6**	**87.0**

Table 2: Comparison of various planners and grounding methods on ScreenSpot. Bold denotes the best performances of GUI agents based on public data.

GUI Agent Evaluation, respectively. We chose Qwen2.5-VL-7B (Bai et al., 2025) as the base model for training and fine-tuned it using the merged grounding data. The training is divided into two steps: Grounding and Planning with backtracking.

4.1 Training Details

Training Data. For training in the grounding stage, we integrated open-source data: OS-Atlas (Wu et al., 2024), Uground (Gou et al., 2024), Aguvis (Xu et al., 2024), and Aria-UI (Yang et al., 2024), and standardized them into the unified format. We provide basic data statistics for training in Table 1. For training in the Planning stage, we selected Aguvis (Xu et al., 2024), a dataset annotated with Observation, Thought, and low-level Instruction as the data source, and performed backtracking transformations. It includes trajectory data from AITW(Rawles et al., 2023), MM-Mind2Web(Deng et al., 2023), GUIAct(Chen et al., 2024a), AMEX(Chai et al., 2024), AndroidControl (Li et al., 2024b) and GUI-Odyssey (Lu et al., 2024).

4.2 Grounding Capability Evaluation

In order to evaluate the impact of data scaling on the agent's grounding ability, we conducted experiments on the ScreenSpot (Cheng et al., 2024) dataset. We first compared the accuracy of our model with other baselines, and then evaluated the changes in the model's grounding ability under different data scales.

ScreenSpot. Table 2 reports the accuracy scores of our proposed ScaleTrack and various baselines, including the open source models such as UGround(Gou et al., 2024), Aria-UI (Yang et al., 2024), OS-Atlas(Wu et al., 2024), Aguvis (Xu et al., 2024), and UI-TARS (Qin et al., 2025), which uses In-house data. We can see from the table that the performance of ScaleTrack clearly surpasses those of the baseline methods that use open-source data and outperforms the previous state-of-the-art(SOTA) model (Xu et al., 2024) by 2.6% in the average Score. Moreover, ScaleTrack gains a more obvious advantage in Icon/Widget subitems that are more difficult to generalize, with improvements of 7.9%, 7.9%, and 5.4%, respectively. The results demonstrate the generalization ability gained by the data scaling. The comprehensive data synthesis strategy used by ScaleTrack achieves better results than any isolated data syn-

Agent Models		Data Source	AndroidControl-Low			AndroidControl-High			GUI Odyssey		
			Type	Grounding	SR	Type	Grounding	SR	Type	Grounding	SR
Agent Framework											
GPT-4o	SeeClick	Public	-	-	52.8	-	-	41.8	-	-	-
	UGround	Public	-	-	62.4	-	-	48.4	-	-	-
Agent Model											
Claude		In-house	74.3	0.0	19.4	63.7	0.0	12.5	60.9	0.0	3.1
GPT-4o		In-house	74.3	0.0	19.4	66.3	0.0	20.8	34.3	0.0	3.3
InternVL-2		In-house	90.9	84.1	80.1	84.1	72.7	66.7	82.1	55.5	51.5
UI-TARS		In-house	98.0	89.3	90.8	83.7	80.5	72.5	94.6	90.1	87.0
UI-TARS-72B		In-house	98.1	89.9	91.3	85.2	81.5	74.7	95.4	91.4	88.6
SeeClick		Public	93.0	73.4	75.0	82.9	62.9	59.1	71.0	52.4	53.9
Aria-UI		Public	-	87.7	67.3	-	43.2	10.2	-	**86.8**	36.5
OS-Atlas		Public	93.6	88.0	85.2	85.2	**78.5**	71.2	84.5	67.8	62.0
Aguvis		Public	-	-	80.5	-	-	61.5	-	-	-
ScaleTrack		Public	**96.6**	**91.4**	**90.2**	**92.1**	77.3	**80.4**	**93.5**	80.2	**80.4**

Table 3: Comparative results on AndroidControl and GUI Odyssey.

thesis criterion, and the superiority of our method also shows the advantage of combining data from different sources.

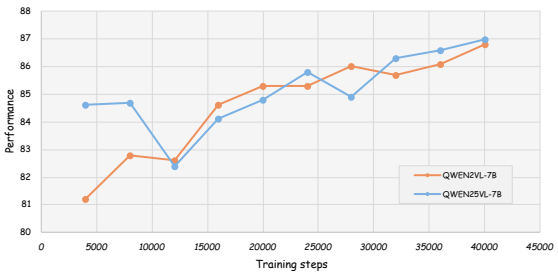

Figure 4: Scaling curves over different base models.

The Effect of Grounding Data Scaling. To further analyze the effectiveness of grounding data scaling, we plot the accuracy scores of ScaleTrack on ScreenSpot in the different training steps. As illustrated in Figure 4, as the data scales up, the average accuracy score fluctuates, but overall it will gradually improve. The result suggests great potential for continuously expanding grounding data to improve performance. Especially when the initial capability of the base model is weak (*e.g.*, Qwen2VL-7B), the GUI grounding capability of the model improves more significantly as the amount of training data increases.

4.3 Offline Agent Capability Evaluation

AndroidControl. We evaluate ScaleTrack on mobile devices using Android automation dataset AndroidControl (Li et al., 2024b), which encompasses 15,000 demonstrations from human raters performing a diverse variety of tasks on 833 different apps spanning 40 app categories on Android devices. Following (Li et al., 2024b; Xu et al., 2024), we randomly sample 800 steps to create a subset. We report the action type accuracy, grounding accuracy, and step success rate on out-of-domain data within both high-level and low-level tasks. As depicted in Table 3, ScaleTrack surpassed strong baselines that use available data on both Low and High levels, achieving the step success rate of 94.5% for the Low level and 82% for the High level. Notably, the data source of the ScaleTrack's planning stage is the same as that of Aguvis, and no additional planning data annotations are added, which proves the effectiveness and scalability of our backtracking strategy.

GUI-Odyssey. GUI-Odyssey (Lu et al., 2024) is a comprehensive dataset for evaluating cross-app navigation agents. It consists of 7,735 episodes from 6 mobile devices. Following (Xu et al., 2024), we randomly sample 500 episodes to create a subset and report the action type accuracy, grounding accuracy, and step success rate. Table 3 reports that ScaleTrack achieved the best performance (80.4%) among the public data models on step success rate.

4.4 Online Agent Capability Evaluation

To better test the performance of ScaleTrack in real-world environments, we also tested Scale-

Agent Models	AndroidControl-Low			AndroidControl-High			GUI Odyssey		
	Type	Grounding	SR	Type	Grounding	SR	Type	Grounding	SR
ScaleTrack	96.6	91.4	90.2	92.1	77.3	80.4	93.5	80.2	80.4
w/o backtracking	94.0	90.7	88.0	86.7	79.5	75.9	87.3	75.9	71.7
diff	-2.6	-0.7	-2.2	-4.1	2.2	-4.5	-6.2	-4.3	-8.7

Table 4: Ablation results of ScaleTrack.

Planner	Grounding	AndroidWorld$_{SR}$
GPT-4-Turbo	UGround	31.0
GPT-4o	UGround	32.8
GPT-4o	AGUVIS	37.1
GPT-4o	Aria-UI	39.7
GPT-4o	ScaleTrack	**46.0**

Table 5: Task Success Rates (SR) on AndroidWorld.

Track on the real-time interaction benchmarks. We use AndroidWorld (Rawles et al., 2024) and MobileMiniWob(Rawles et al., 2023) for online mobile agent evaluation in an Android emulator environment. We use GPT-4o as the planner and ScaleTrack to locate elements and instructions.

AndroidWorld. (Rawles et al., 2024) contains a highly reproducible benchmark of 116 handcrafted tasks across 20 apps and calculates the final state success rate on the device by checking the final system state. In our experiments, we employ GPT-4o as the planner and ScaleTrack as the grounder. As shown in Table 5, ScaleTrack achieves the highest average task success rate of 46.0%, outperforming the baseline models, which further highlights that data-scaling and backtracking help the model handle diverse element descriptions and instructions in real-world environments.

Planner	Grounding	MobileMiniWob$_{SR}$
GPT-4-Turbo	Choice	59.7
Gemini 1.5 Pro	SoM	40.3
GPT-4o	AGUVIS	55.0
GPT-4o	Aria-UI	60.4
GPT-4o	ScaleTrack	**62.0**

Table 6: Task Success Rates (SR) on MobileMiniWob.

MobileMiniWob. (Rawles et al., 2023) containts 92 tasks from MiniWob++ (Zheng et al., 2023). As shown in Table 6, ScaleTrack outperforms existing work in task success rate on MobileMiniWobs when using GPT-4o as the planner, with an average SR of 62.0% on MobileMiniWob. This comparison particularly highlights the effectiveness of the data scaling and backtracking in our GUI agent model.

4.5 Ablation Study

We further studied how ScaleTrack improves the model's planning ability by tracking the history of actions that led to the current state. We compared the model performance before and after backtracking data training on three offline evaluation datasets.

We show the ablation results in Table 4. After training on backtracking data, the accuracy of the model on AndroidControl-Low, AndroidControl-High, and GUI-Odyssey datasets increased by 2.2%, 4.5%, and 8.7% on step success rate, respectively. Furthermore, the recognition accuracy of the model after backtracking training in action type has increased by 2.6%, 4.1%, and 6.2%, respectively. The results demonstrate the action understanding and prediction abilities gained by the backtracking training.

5 Conclusion

In this work, we introduce ScaleTrack for scaling and backtracking automated GUI agents. We find two widespread problems, including isolated data synthesis criterion and unconsidered backtracking capability. To alleviate these problems, we first propose to integrate several data-driven GUI element enhancement methods to scale the training process of GUI grounding and unify a wide range of grounding data into a fixed training template. We then propose a hybrid training strategy to learn forward-planning and backtracking capabilities simultaneously. We conduct extensive experiments on several benchmark datasets, like grounding evaluation, offline and online evaluation, and the results demonstrate the effectiveness of our proposed ScaleTrack.

Limitations

Although our approach has shown promising results, there are several limitations that need to be addressed in future work. First, ScaleTrack establishes connections between past, present, and future by backtracking previous actions, but this comes with increased token output costs. In practical applications where cost and real-time performance are critical, this may become a bottleneck and limit the efficiency of deployment. Second, ScaleTrack is based solely on algorithmic improvements using open-source datasets, without designing new data annotation methods or providing new data sources. In the future, it will be necessary to construct more high-quality grounding and planning data to achieve more data scale. Finally, ScaleTrack focuses on PC and mobile scenarios, and does not cover all GUI agent scenarios such as browser. Future work should expand the ScaleTrack approach to more scenarios and tasks to provide a more comprehensive evaluation of its generalizability.

Ethic Statement

We emphasise that research and application regarding the GUI Agent are conducted in a virtual environment, strictly adhering to privacy protection protocols, and do not involve any leakage of private data or applications.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

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

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, and 1 others. 2024a. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, and 1 others. 2024b. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *Science China Information Sciences*, 67(12):220101.

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*.

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

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

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, and 1 others. 2024. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*.

Siyuan Hu, Mingyu Ouyang, Difei Gao, and Mike Zheng Shou. 2024. The dawn of gui agent: A preliminary case study with claude 3.5 computer use. *arXiv preprint arXiv:2411.10323*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36:39648–39677.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and 1 others. 2024a. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024b. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

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*.

Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, Wenhao Yu, and Dong Yu. 2023. Laser: Llm agent with state-space exploration for web navigation. *arXiv preprint arXiv:2309.08172*.

OpenAI. 2023. Gpt-4v(ision) system card. Technical report, OpenAI.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, and 1 others. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*.

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

Yueqi Song, Frank Xu, Shuyan Zhou, and Graham Neubig. 2024. Beyond browsing: Api-based web agents. *arXiv preprint arXiv:2410.16464*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024a. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Song XiXuan, and 1 others. 2024b. Cogvlm: Visual expert for pretrained language models. *Advances in Neural Information Processing Systems*, 37:121475–121499.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and 1 others. 2024. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*.

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*.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2024. Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*.

Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. 2023. Ferret: Refer and ground anything anywhere at any granularity. *arXiv preprint arXiv:2310.07704*.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. 2025. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23378–23386.

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*.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2023. Synapse: Trajectory-as-exemplar prompting with memory for computer control. *arXiv preprint arXiv:2306.07863*.

A Training Details

Training Settings. We choose Qwen2.5-VL (Bai et al., 2025) as the base model for training and employ the AdamW optimizer with a learning rate of $1e{-}5$ and employ a cosine learning rate scheduler with a warm-up ratio of 0.03 steps. We utilize a global batch size of 128 in the grounding stage and 64 in the planning stage and employ DeepSpeed ZERO3-style data parallelism. We train ScaleTrack following a two-stage procedure. First, all grounding data is used to train ScaleTrack's basic GUI grounding capability. Then, based on the model trained in the grounding stage, planning data with forward-planning and historical backtracking are input into the model to further enhance the planning ability of the model. We train ScaleTrack on a cluster of 8 nodes with V100-80G GPUs.

B ScaleTrack Unified Design

Details of Action Space in ScaleTrack. In this section, we will introduce the unified action space of our training framework, ScaleTrack. As shown in Tables 7, our action space encompasses common actions in both web and mobile scenarios, ensuring the generalizability of the agent model across different environments while maintaining flexibility in specific contexts. Specifically, for mobile, the actions include: 'key', 'click', 'long_press', 'swipe', 'type', 'system_button', 'open', 'wait', and 'terminate'. For computer, the actions include: "key", "type", 'mouse_move', 'left_click', 'left_click_drag', 'right_click', 'middle_click', 'double_click', 'scroll', 'wait', 'terminate', 'answer' .

Action functions in mobile environments as an example Following (Bai et al., 2025), we use prompts in JSON format to inform the model of the available functions and their corresponding parameters, which facilitates extensibility. We provide the following action functions for ScaleTrack in the mobile environment, along with their corresponding descriptions.

C Data Curation of ScaleTrack Data Collection

Detailed Source Dataset Statistics. The statistics of our Stage 1 grounding training data are presented in Table 1 of the main text. The Stage 2 planning data consists of three components:

- AGUVIS data processed with backtrack augmentation(see Figure 6). We add constraints to the user prompt: derive the previous action based on the previous UI screenshot and the current UI screenshot, and output it in the form of low-level instructions.

- AndroidControl data supplemented with low-level instruction descriptions in the user prompts(see Figure 7). Adding the low-level instructions of the current action to the user prompt can significantly improve the model's low-level prediction ability.

- GUI Odyssey data enhanced by incorporating historical screenshots in the action history(see Figure 8). In each step of the historical actions within the user prompt, add historical screenshots and represent the historical actions in a high-level form, thereby enhancing the model's understanding of the task and the accuracy of planning.

Data Type	Action Space (JSON format)
Web	{"name": "computer_use", "arguments": {"action": "key", "keys": ["ctrl", "a"]}} {"name": "computer_use", "arguments": {"action": "left_click", "coordinate": [x, y]}} {"name": "computer_use", "arguments": {"action": "type", "text": "text"}} {"name": "computer_use", "arguments": {"action": "answer", "text": "text"}} {"name": "computer_use", "arguments": {"action": "terminate", "status": ["success"]}} {"name": "computer_use", "arguments": {"action": "wait", "time": "time"}} (Total 15 action types...)
Mobile	{"name": "mobile_use", "arguments": {"action": "system_button", "button": "enter"}} {"name": "mobile_use", "arguments": {"action": "click", "coordinate": [x, y]}} {"name": "mobile_use", "arguments": {"action": "type", "text": "text"}} {"name": "mobile_use", "arguments": {"action": "swipe", "coordinate": [x, y], "coordinate2": [x, y]}} {"name": "mobile_use", "arguments": {"action": "type", "text": "text"}} {"name": "mobile_use", "arguments": {"action": "status", "button": "success"}} {"name": "mobile_use", "arguments": {"action": "wait", "time": "time"}} (Total 11 action types...)

Table 7: Action space specifications for Web and Mobile environments

Action Functions for ScaleTrack in the Mobile Environment

```
You are a helpful assistant.
Tool
You may call one or more functions to assist with the user query.
You are provided with function signatures within XML tags:
{"type": "function", "function": {"name_for_human": "mobile_use", "name": "mobile_use",
"description": "Use a touchscreen to interact with a mobile device, and take screenshots.
This is an interface to a mobile device with touchscreen. You can perform actions like
clicking, typing, swiping, etc.
Some applications may take time to start or process actions, so you may need to wait and
take successive screenshots to see the results of your actions.
The screen's resolution is 644x1400.
Make sure to click any buttons, links, icons, etc with the cursor tip in the center of the
element. Don't click boxes on their edges unless asked.", "parameters": {"properties":
{"action": {"description": "The action to perform. The available actions are:
key: Perform a key event on the mobile device. This supports adb's keyevent
syntax. Examples: "volume_up", "volume_down", "power", "camera", "clear".
click: Click the point on the screen with coordinate (x, y).
long_press: Press the point on the screen with coordinate (x, y) for specified seconds.
swipe: Swipe from the starting point with coordinate (x, y) to the end point with
coordinates2 (x2, y2).
type: Input the specified text into the activated input box.
system_button: Press the system button.
open: Open an app on the device.
wait: Wait specified seconds for the change to happen.
terminate: Terminate the current task and report its completion status."
...

For each function call, return a json object with function name and arguments within XML
tags:
{"name": , "arguments": }
```

Resolution

Figure 5: System prompt in mobile environments.

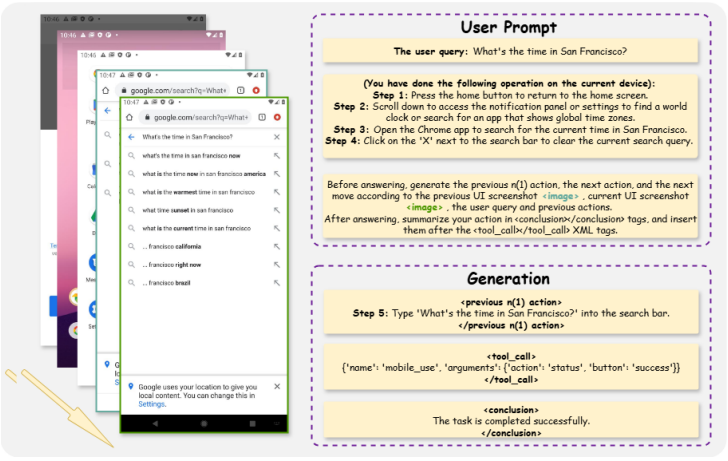

Figure 6: AGUVIS data processed with backtrack augmentation.

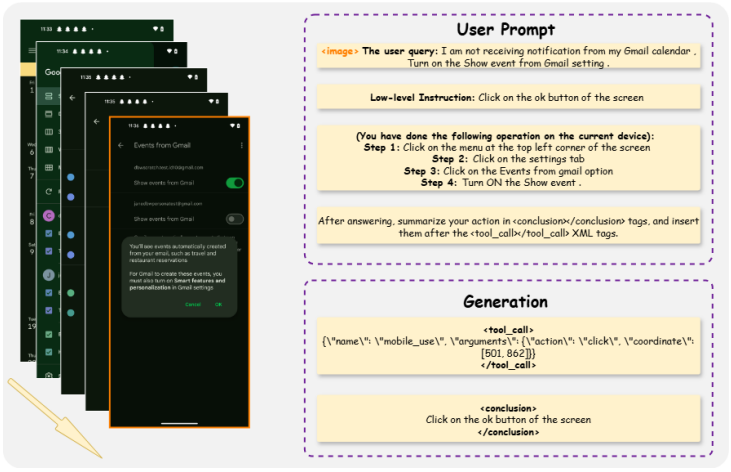

Figure 7: AndroidControl data supplemented with low-level instruction descriptions in the user prompts.

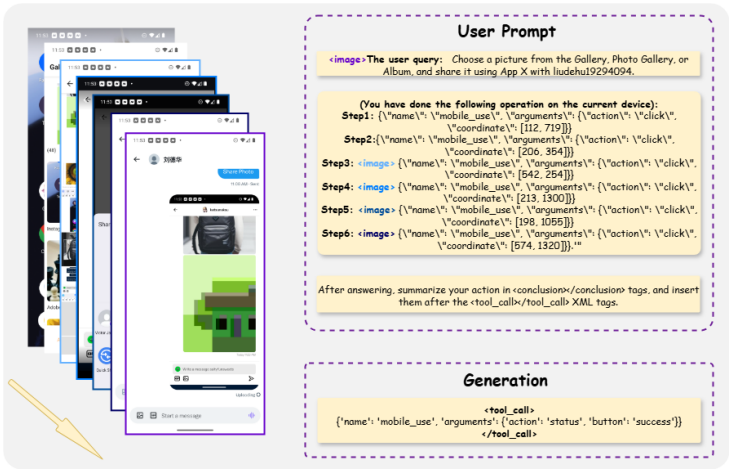

Figure 8: GUI Odyssey data enhanced by incorporating historical screenshots in the action history.