UIPro: Unleashing Superior Interaction Capability For GUI Agents

Hongxin Li[1,2,3,7]   Jingran Su[5]   Jingfan Chen[5]   Zheng Ju[1,2,3]   Yuntao Chen[4✉]   Qing Li[5]
Zhaoxiang Zhang[1,2,3,6✉]

[1]University of Chinese Academy of Sciences (UCAS)
[2]New Laboratory of Pattern Recognition (NLPR), CASIA
[3]State Key Laboratory of Multimodal Artificial Intelligence Systems (MAIS), CASIA
[4]Hong Kong Institute of Science & Innovation, CASIA
[5]PolyU   [6]Shanghai Artificial Intelligence Laboratory   [7]StepFun
Code: https://github.com/ZJULiHongxin/UIPro

Abstract

*Building autonomous agents that perceive and operate graphical user interfaces (GUIs) like humans has long been a vision in the field of artificial intelligence. Central to these agents is the capability for GUI interaction, which involves GUI understanding and planning capabilities. Existing methods have tried developing GUI agents based on the multi-modal comprehension ability of vision-language models (VLMs). However, the limited scenario, insufficient size, and heterogeneous action spaces hinder the progress of building generalist GUI agents. To resolve these issues, this paper proposes **UIPro**, a novel generalist GUI agent trained with extensive multi-platform and multi-task GUI interaction data, coupled with a unified action space. We first curate a comprehensive dataset encompassing 20.6 million GUI understanding tasks to pre-train UIPro, granting it a strong GUI grounding capability, which is key to downstream GUI agent tasks. Subsequently, we establish a unified action space to harmonize heterogeneous GUI agent task datasets and produce a merged dataset to foster the action prediction ability of UIPro via continued fine-tuning. Experimental results demonstrate UIPro's superior performance across multiple GUI task benchmarks on various platforms, highlighting the effectiveness of our approach.*

1. Introduction

The concept of autonomous GUI agents capable of clicking, typing, and scrolling on behalf of humans as personal assistants is an enticing prospect (Fig. 1). Imagine a GUI agent navigating the Internet to perform daily tasks such as using search engines and managing emails, as well as more
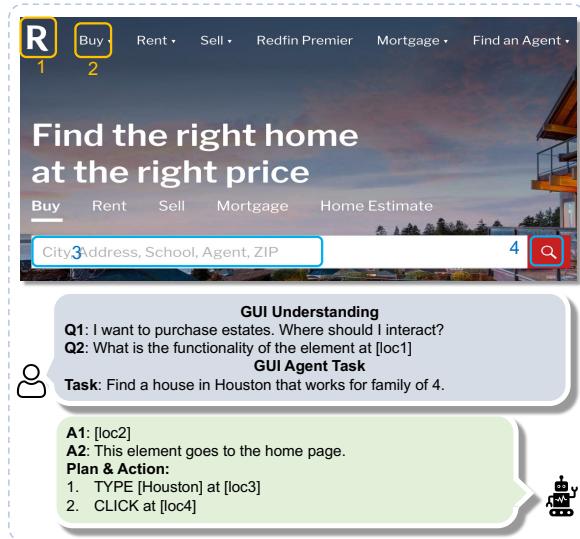

Figure 1. Examples of GUI interaction by a GUI agent.

complex activities like comparing prices across e-commerce platforms and collecting the latest news on stock markets.

To achieve proficient GUI interaction, an agent relies on two fundamental capabilities. Firstly, a foundational visual understanding of GUI elements is essential. This includes the ability to recognize and interpret various components like buttons, text fields, and images, which are integral to navigating and manipulating GUIs effectively. The second capability involves both planning and executing tasks in alignment with user goals, requiring the agent to efficiently translate plans into actionable steps.

Although existing methods [10, 12, 20, 54] have sought to enhance GUI interaction by leveraging the multi-modal comprehension abilities of vision-language models (VLMs), significant challenges persist: (a) **Data scale.** Effective de-

---
✉ Equally advising corresponding authors. E-mails: zhaoxiang.zhang@ia.ac.cn, chenyuntao08@gmail.com.

velopment of generalist GUI agents demands large-scale data since the complete advantages of large-scale training typically do not manifest at smaller scales [6, 47]. Unfortunately, current GUI interaction datasets [2, 7, 14, 25, 29, 48] often lack sufficient size and scenario diversity, hindering the development of generalist GUI agents. (b) **Training recipe** is also critical, as the recent advancements in large language and vision models have largely depended on sophisticated strategies for data curation [28, 35]. Existing GUI interaction datasets typically adopt various formats (e.g., heterogeneous action spaces), which complicates data curation and reduces the effectiveness of training generalist GUI agents.

In this paper, we tackle the aforementioned issues by developing **UIPro**, a generalist GUI agent trained with massive multi-platform multi-task GUI interaction data and unified action spaces. Concretely, to lay the foundation of GUI interaction, we first curate a large-scale GUI-understanding-oriented dataset by annotating and curating data from multiple GUI platforms, including web browsers, Android phones, and iPads of various types and resolutions. We also spot the astonishingly high noise level in the raw GUI sources and release a systematic denoising procedure to reduce the burden of data cleaning in future research. After cleaning, we curate **20.6M** GUI understanding tasks associated with **2.5M** unique screenshots, contributing the largest GUI understanding data collection to the community. This dataset is used to pre-train UIPro to develop a strong GUI understanding, especially element grounding capability, which is the key to downstream GUI agent tasks.

To cultivate UIPro's action prediction ability after pre-training, we propose to better combine multiple heterogeneous GUI agent task datasets with a unified action space. This unified space defines an action superset that accommodates different action definitions used in various data sources. Specifically, we resolve action name conflicts, utilize action arguments suitable for various downstream test environments, and finally format all action ground truths in a unified JSON format. As different platforms usually share similar design principles for human-computer interaction, the unified action space that abstracts platform distinctions helps to develop adaptable GUI agents.

After fine-tuning with unified agent task data, UIPro achieves superior performance on multiple GUI interaction benchmarks on different platforms. Ablation studies also justify that the 20.6M GUI understanding data and the proposed unified action space both notably benefit GUI interaction capabilities.

The main contributions of our work are three-fold:

1. We curate a massive and clean GUI understanding dataset with 20.6M task samples on multiple platforms to imbue strong GUI grounding capability into GUI agents.

2. We propose to unify the action space of heterogeneous GUI agent task datasets. This unified action space can better integrate multiple data sources to enhance the GUI interaction capability of GUI agents.

3. Undergoing training with the curated GUI understanding dataset and unified agent task dataset, we build UIPro, an advanced GUI agent that achieves superior agent task performance on multiple platforms.

2. Related Works

2.1. Existing GUI Datasets

Compared with natural image datasets [39, 40], GUI datasets have received less attention. Some works have developed datasets for mobile GUIs [3, 7, 9, 24, 25, 45] centered on the RICO dataset [13], which contains 72K Android app screenshots. Notable examples include Widget Captioning [25], which examines the captions and linguistic features of UI elements, and RICOSCA [24], which maps single-step instructions to corresponding UI elements. Mobile-Views [17] utilizes chip clusters to parallelize GUI data collection, providing a huge dataset covering 20k Apps. Some works focus solely on web scenarios: WebUI [48] crawls 400k rendered webpages associated with automatically extracted metadata used for enhancing GUI visual understanding. GUICourse [10] renders URLs from the *C4* [37] and generates element-level understanding, GUI Q&A, and action prediction tasks for building GUI agents. To expand diversity, [2] and [20] curate huge datasets at the scale of hundreds of millions, but these datasets have not been publicized. Despite the smaller scale, several works [12, 18, 30, 49] have collected multi-platform datasets, helping to push the boundary of GUI agents. Our work also contributes a new large-scale GUI understanding dataset by cleaning, annotating, and curating GUI data on multiple platforms.

2.2. GUI Agents

Existing works [12, 19, 34, 36, 50, 54, 56] have explored GUI agents leveraging the emergent capabilities of LLMs and VLMs to tackle long-horizon tasks. Some studies focus on text-based environments [14, 19, 34, 53, 60], with minimal interaction in more complex, dynamic settings, such as visual GUI environments. Operating GUIs like humans presents significant challenges for autonomous agents due to the need for pixel-level control within a complex action space. To address these challenges, recent studies have emphasized visually grounded, pixel-level GUI agents [4, 12, 18, 20, 36, 41, 49, 50, 54, 56]. For instance, Pixel2Act [41] and WebGUM [16] concentrate on the web domain where they develop models that can predict actions on the HTML elements with visual information. Ferret-UI [54], SeeClick [12], CogAgent [20], and OS-ATLAS [49] curate large-scale data to train open-source multimodal models as GUI agents. Additionally, some works [18, 33] also explore combining proprietary foundation models (e.g., GPT-

Dataset	UI Type	#Samples	#Task Types	Open-source
Wid. Cap. [25]	Mobile	163k	1	✓
RICOSCA [24]	Mobile	295k	1	✓
RefExp [3]	Mobile	390k	1	✓
SeeClick [12]	Web, Mobile	5.3M	6	✓
CogAgent [20]	Web	>247M	5	✗
Ferret-UI [54]	Mobile	250k	11	✗
ScreenAI [2]	Mobile	421M	4	✗
UGround [18]	Web, Mobile	10M	5	✓
OS-ATLAS [49]	Web, Mobile, Desktop	13.6M	4	✓
GUICourse [10]	Web, Mobile	10.8M	3	✓
UIPro (ours)	Web, Mobile	20.6M	**13**	✓

Table 1. Comparing large-scale datasets tailored to fostering GUI understanding ability.

4 [43] and PaLM2 [1]) as task decomposers while training VLMs as low-level action executors. However, the generalizability of these works is often insufficient trajectory data and inconsistent action spaces. To address this, we propose a unified action space to integrate multiple data sources, enhancing GUI interaction capabilities.

3. UIPro: Advanced GUI Agent

This section introduces UIPro, an advanced GUI agent built upon massive GUI interaction data curated by us (Fig. 2).

3.1. Model Architecture

UIPro is designed to perform GUI interaction by receiving user tasks as input and then locating target elements (GUI element grounding) and predicting actions advancing towards task completion (GUI agent task). UIPro adopts the popular architecture used by recent VLMs, such as LLaVA [27]. This architecture combines a pre-trained visual backbone f_ϕ (e.g., ViT [15]) and a large language model (e.g., Llama [44]) to build a model capable of processing both textual and visual inputs. We build UIPro on two base models capable of processing high-resolution GUI screenshot images: **UIPro-SLiME** trained from a tabula rasa SLiME-Gemma-2B [58], and **UIPro-Qwen2VL** fine-tuned from Qwen2-VL-7B-Instruct [46].

3.2. GUI Interaction Data Construction

We curate massive GUI understanding and agent task data to grant UIPro strong GUI interaction performance.

3.2.1. Curating GUI Understanding Data

To imbue GUI understanding, especially *grounding* capability, into UIPro, we first curate a super-large dataset by annotating crawled GUI data and generating diverse GUI grounding tasks. Each grounding task is represented by a *<screenshot, referring expression (RE), coordinates>* triplet. We generate various REs for GUI elements: **(a) Element Description** describes visual appearances, element types, and positions of elements, aiming to establish a fundamental GUI understanding capability. For example, a house-shape element might be described as "a navigating-home button at

the top left". We also extract the displayed texts for pure-text elements and icon classes (e.g., *Home icon* and *Instagram icon*) for iconic ones as their REs. **(b) User Intent** describes how a user intends to interact with a specific element [7], such as "focus on the Password textbox" for a password input field. The aforementioned two types of REs are derived from the element properties in GUI source code, such as HTML or Android view hierarchies. **(c) Contextual Functionality** describes the interactive affordance of GUI elements [22]. For example, an element shaped like an upward arrow might be annotated with "This element enables users to share content with others". This RE type helps models comprehend the functional semantics of elements, complementing the textual and appearance-level semantics provided by the former two types. We generate this RE type following [22].

Using the triplets, we generate massive GUI grounding tasks, including *funcgnd*, *elemgnd*, *textgnd*, *icongnd*, and *intentgnd*) based on the functionality annotations, element descriptions, displayed texts, icon classes, and user intents, respectively. Following [12] and [20], we also produce dual referring tasks (e.g., *funcref*, *elemref*, *OCR*, *iconref*) that prompt the model to generate referring expressions for specific element locations. To further enhance understanding capability, we follow Ferret-UI [54] to also include *widget listing*, *GUI captioning*, and *GUI Q&A* tasks.

To expand scenario diversity, the used GUI sources include multi-resolution webpages from Common Crawl, mobile UIs from various Android device types, and publicized raw GUI collections (details in the Appendix).

3.2.2. Unifying GUI Agent Task Data

After pretraining UIPro with the GUI understanding data, we fine-tune it on downstream GUI agent tasks. Although various GUI trajectory datasets exist, each offers limited training trajectories and employs different action spaces with inconsistent implementations. For instance, AITW [38] defines *swipe* as DUAL_POINT(start, end) while AndroidControl [23] uses scroll(direction). Mind2Web [14] requires a GUI agent to output the text box coordinates for a typing action, while GUIAct [10] and WebLINX [32] do not support this coordinate argument for typing. Moreover, swipe and scroll are not used consistently across the existing datasets. These conflicting definitions complicate combining multiple sources for multi-task fine-tuning.

To handle this problem, we merge different sources of GUI agent tasks by proposing a unified action space, which employs general action definitions as a superset over heterogeneous ones. For example, we define *swipe* as swipe(start, direction, distance) to represent how users start the swipe at a specific point on the screen, move their finger in a certain direction, and cover a certain distance to complete the swipe gesture. This definition accommodates the swipe usages

Figure 2. We develop **UIPro**, an advanced GUI agent capable of interacting with GUIs given user tasks. We first curate a large-scale GUI understanding dataset containing diverse tasks used to pre-train UIPro. Then, we merge heterogeneous GUI agent task data with unified action spaces to fine-tune UIPro, granting UIPro superior planning ability,

of multiple datasets, including AITW [38] and Android-Control [23]. We design three unified action spaces for mobile devices, web browsers, and desktop environments, respectively, representing three different embodiments of UIPro. For instance, the mobile action space incorporates `tap`, `long_press`, `drag`, `input_text`, `navigate_home/back/recent`, `press_enter`, `swipe`, `wait`, `status_complete/infeasible` (Full definitions in Sec. 8.2 of Appendix). This action unification enhances UIPro's interoperability and facilitates the integration of diverse data sources for more effective multi-task learning. As training data in mobile and web scenarios are significantly richer than those in desktop environments (e.g., Windows and MacOS), we mainly merge datasets in mobile and web scenarios to justify the efficacy of the proposed action space unification.

3.2.3. Systematic Denoising Procedure

GUI data may contain noise due to GUI design defects[1]. Training samples generated based on these noisy elements will likely compromise the performance of the trained model, as shown in our experiments. To tackle this issue, we inspect the used GUI data sources, summarize invalid element types, and contribute a denoising procedure useful for the community. The checking items of the procedure include detecting blank elements according to the standard deviation of region color, employing OCR tools[2] to remove invisible elements, and removing elements with invalid bounding boxes (the full procedure is detailed in Sec. 8.3 of Appendix). After checking multiple data sources, we surprisingly found that the noise ratio is unignorable, with one source [9] reaching 29.0%, as shown in Tab. 17. We use this denoising procedure

---

[1]A recent survey (https://webaim.org/projects/million/) revealed that 95.9% of home pages contained accessibility errors, averaging 56.8 errors per page.

[2]https://pypi.org/project/pytesseract/

Grounding
Intent Gnd 5.10M (24.7%)
Other Element Gnd 1.60M (7.7%)
Functionality Gnd 0.38M (1.8%)
Text Gnd 3.90M (18.9%)
Icon Gnd 0.30M (1.5%)

Referring
Text Ref 4.80M (23.2%)
Icon Ref 0.43M (2.1%)
Functionality Ref 0.38M (1.8%)
Other Element Ref 0.10M (0.5%)

Others
Q&A 2.90M (14.0%)
Widget Listing 0.68M (3.3%)
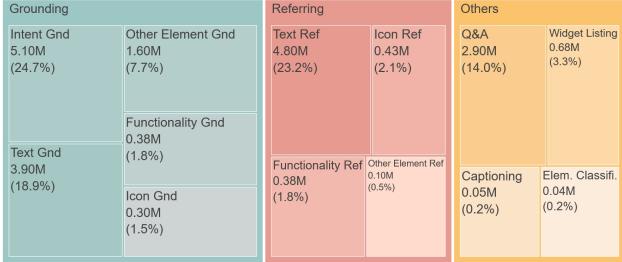Captioning 0.05M (0.2%)
Elem. Classifi. 0.04M (0.2%)

Figure 3. The proportions of the task types in our curated 20.6M-sample GUI understanding dataset introduced in Sec. 3.2.1. Please refer to Sec. 7 in Appendix for data sources and detailed statistics.

Methods	Size	General	Install	GoogleApps	Single	WebShop	Overall
GPT-4V-SoM [51]	-	41.7	42.6	49.8	72.8	45.7	50.5
GPT-4V-OmniParser [33]	-	48.3	57.8	51.6	77.4	52.9	57.7
Qwen2-VL [46]	7B	22.0	27.9	24.6	32.7	18.2	25.1
Fuyu-GUI [10]	8B	-	50.9	41.6	45.7	43.8	45.5
SeeClick [12]	10B	54.0	66.4	54.9	63.5	57.6	59.3
OS-ATLAS [49]	7B	57.9	63.4	55.5	79.1	59.7	63.1
UIPro-Qwen2VL (ours)	7B	**64.4**	**74.6**	**67.9**	**79.4**	**67.6**	**70.4**
MiniCPM-GUI [10]	3B	-	62.3	46.5	67.3	57.5	58.4
UIPro-SLiME (ours)	3B	**67.0**	**71.4**	**65.4**	**73.2**	**62.9**	**68.0**

Table 2. **Comparison on the AITW benchmark [38].** UIPro surpasses the compared methods by a large margin and even outperforms the strong GPT-4V-OmniParser [33]. The Step SR metric is reported for the five splits, and the Overall column denotes the Step SR calculated over all splits.

Methods	Size	High-Low			Only High		
		Type ↑	Click ↑	Step SR ↑	Type ↑	Click ↑	Step SR ↑
Qwen2-VL [46]	7B	70.1	31.9	44.2	46.9	15.8	22.1
OS-Atlas [49]	7B	83.7	59.0	62.4	83.7	**59.0**	62.4
UIPro-Qwen2VL (ours)	7B	**96.3**	**84.3**	**85.5**	**83.8**	57.2	**64.0**
Qwen2-VL [46]	2B	34.9	9.8	11.3	22.0	4.1	6.1
ShowUI [26]	2B	47.2	43.3	29.5	45.3	33.0	22.9
UIPro-SLiME (ours)	3B	**98.1**	**44.1**	**61.1**	**84.6**	**34.5**	**46.0**

Table 3. **Comparison on AndroidControl [23].** UIPro outperforms the other methods in step SR on the two settings. Type and Click denote the accuracy (%) of predicting action types and predicting the two actions requiring element localization (i.e., click and long-press), respectively. Evaluation is repeated three times, and the average is reported.

Methods	Size	GUIAct-Web			GUIAct-Mobile		
		Type ↑	Ground ↑	Step SR ↑	Type ↑	Ground ↑	Step SR ↑
GPT-4o	-	77.1	45.0	41.8	63.0	58.2	44.3
Qwen2-VL [46]	7B	63.7	27.9	23.6	51.2	21.5	19.7
UIPro-Qwen2VL (ours)	7B	**85.1**	**82.7**	**69.1**	**84.2**	**78.4**	**67.2**
ShowUI [26]	2B	69.4	61.1	48.8	65.4	50.3	36.5
MiniCPM-GUI [10]	3B	79.4	59.1	60.2	71.7	53.3	44.7
UIPro-SLiME (ours)	3B	**97.8**	**70.1**	**68.2**	**81.2**	**75.3**	**65.2**

Table 4. **Comparison on GUIAct [10].** UIPro achieves notably higher step SR compared to the methods with equal model sizes.

to ensure our dataset is generated from clean elements.

Data Overview We ultimately contribute **20.6M** GUI understanding task samples associated with 3.3M clean elements from 2.5M unique GUI screenshots (see the statistics in Tab. 1). The task proportions are shown in Fig. 3. Note that **67% of the samples are newly annotated by the authors** and 33% are collected and cleaned from existing open-source datasets. For GUI agent task fine-tuning, we produce a 380k-size dataset integrating six sources for mobile tasks and a 145k-size one integrating three sources for web tasks. Data details are listed in Sec. 7 and full dataset statistics are provided in Tables 16 and 18 in Appendix.

4. Experiments

This section evaluates UIPro as an advanced GUI agent through extensive experiments: Sec. 4.2 covers GUI agent tasks, Sec. 4.3 examines GUI grounding, and Sec. 4.4 provides ablation studies and analysis.

4.1. Fine-Tuning Settings

Since UIPro-SLiME begins as a tabula rasa VLM, we initially use LLaVA-Pretrain [27] to train the VL-projector, then use RefCOCO [55] data to fine-tune UIPro-SLiME with the visual encoder frozen, and finally continue fine-tuning it with our GUI understanding samples for one epoch. For UIPro-Qwen2VL, as Qwen2-VL-7B [46] already possesses strong visual understanding capabilities, we fine-tune it for one epoch with a 4.4M subset randomly extracted from the grounding tasks of the full set. The two base models are both fine-tuned with a learning rate of 3e-5. All coordinates are normalized in the range [0, 1000].

In the agent-task fine-tuning stage, we adhere to the training sample formatting outlined in SeeClick [12], which includes the task and action history in the prompt and formats the ground truth action as a `JSON` object recording the action type and arguments. Action definitions are not included in the prompt since UIPro is fine-tuned and tested separately for web and mobile platforms. Our preliminary experiments indicated that excluding the action space leads to more ef-

ficient training. We fine-tune UIPro for six epochs until performance on downstream tasks plateaus. Additional hyperparameters are detailed in the Appendix.

4.2. GUI Agent Task Evaluation
4.2.1. Benchmarks And Compared Methods

We evaluate on three **mobile device control** benchmarks:
- **AITW** [38]: This large Android control dataset covers multiple scenarios, including App Store operations, browser use, and web shopping, across more than 350 apps. We use the same train/test split as in SeeClick [12].
- **AndroidControl** [23]: This Android dataset comprises 15,000 unique tasks over 833 apps, recorded from human demonstrations for training and evaluation. In addition to high-level tasks, AndroidControl provides low-level instructions for each step. We test on two settings: one with only high-level tasks and one with both high-level tasks and low-level instructions. We also follow AndroidCon-

Methods	Size	Cross-Task			Cross-Website			Cross-Domain		
		Elem. Acc.↑	Op. F1↑	Step SR↑	Elem. Acc.↑	Op. F1↑	Step SR↑	Elem. Acc.↑	Op. F1↑	Step SR↑
GPT-4V + SoM [33]	-	-	-	32.7	-	-	23.7	-	-	20.3
OmniParser (GPT-4V) [33]	-	42.4	87.6	39.4	41.0	84.8	36.5	45.5	85.7	42.0
UGround (GPT-4o) [18]	-	47.7	-	-	46.0	-	-	46.6	-	-
Qwen-VL [5]	10B	15.9	86.7	13.3	13.2	83.5	9.2	14.1	84.3	12.0
SeeClick [12]	10B	28.3	87.0	25.5	21.4	80.6	16.4	23.2	84.8	20.8
Fuyu-GUI [10]	8B	19.1	86.1	15.6	13.9	80.7	12.2	14.2	83.1	11.7
UIX-Qwen2VL [30]	7B	43.4	-	38.2	39.2	-	31.0	40.4	-	34.9
OS-ATLAS [49]	7B	39.7	80.0	36.7	39.5	78.1	35.7	40.4	79.5	37.2
UIPro-Qwen2VL (ours)	7B	**52.1**	**89.4**	**48.4**	**47.8**	**85.5**	**43.6**	**51.5**	**86.7**	**45.5**
MiniCPM-GUI [10]	3B	23.8	86.8	20.8	20.3	81.7	17.3	17.9	74.5	14.6
UIPro-SLiME (ours)	3B	**31.8**	**87.2**	**28.7**	**25.0**	**82.9**	**20.7**	**24.7**	**84.6**	**22.0**

Table 5. **Comparison on Multimodal-Mind2Web [14].** UIPro-Qwen2VL-7B outperforms existing methods of equal and larger size and surpasses the methods that use proprietary models. Elem. Acc. denotes the percentage of samples for which the models predict correct target coordinates. Op. F1 denotes the token-level F1 score for the predicted action.

trol [23] to use a random-500 split for testing. Three runs are conducted, and the average performances are reported.
- **GUIAct-Smartphone** [10] also requires agents to complete mobile app operation tasks. It is generated by cleaning and re-annotating a subset of the AITW dataset [38].

These **web browser control** benchmarks are used:

- **GUIAct-Web** [10]: This dataset collects websites, uses large language models (LLMs) to produce information-searching tasks, and employs annotators to label ground truth actions for evaluation.
- **Multimodal-Mind2Web** [14] provides ground truth trajectories from human demonstrations for offline evaluation. It tests GUI agents on three test splits: unseen tasks, websites, and domains.

These benchmarks assess agents' ability to predict the next actions given the current GUI screenshot image, user tasks, and action history. The major evaluation metric is **Step Success Rate (Step SR)** [14, 38], where a generated step is considered correct only if the action type and arguments match ground truths. Step SR is calculated as the ratio (%) of successful steps against all steps. More implementations are provided in the Appendix.

For **AITW**, these methods are compared: a) GPT-4V-SoM [51] that detects GUI elements with IconNet [42] and then performs Set-of-Marks prompting [52]. b) Omni-Parser [33] that employs a GUI parser to detect interactable regions to enhance SoM prompting. c) GUI-oriented VLMs, such as SeeClick [12], Fuyu-GUI [10], and OS-ATLAS [49]. For **AndroidControl**, these methods are compared: a) OS-ATLAS [49] fine-tuned based on Intern-VL [11]. b) Qwen2-VL [46]: a general VLM trained with GUI agent capability. c) ShowUI [26] developed on Qwen2-VL-2B [46]. For **GUIAct**, MiniCPM-GUI [10], ShowUI [26], and Qwen2-VL [46] are compared. Likewise, for **Mind2Web**, we also compare with methods equipped with proprietary models and expert VLMs dedicated to GUI agent tasks. As OS-

ATLAS [49] has not been evaluated on Mind2Web, we test this model by adding the Mind2Web action space to its system prompt.

4.2.2. Experimental Results

Tables 2, 3, and 4 show that UIPro achieves superior step SR on the mobile benchmarks. On AITW, UIPro-Qwen2VL leads in performance, surpassing the GPT-4V-OmniParser [33], which employs Omniparser to detect elements as a Set-of-Marks for GPT-4V. Additionally, UIPro-Qwen2VL exceeds OS-ATLAS based on Qwen2-VL-7B by 7.7 in overall step SR. On AndroidControl and GUIAct-Mobile, UIPro maintains high step SR, outperforming methods using proprietary models and GUI-oriented VLMs.

Tables 4 and 5 highlight UIPro's strong action prediction capabilities in web scenarios. UIPro surpasses competitors assisted by proprietary VLMs (in gray) and outperforms GUI-oriented VLMs of equal or larger sizes. The improvements on the Mind2Web splits over previous models, including OS-ATLAS and MiniCPM-GUI, underscore UIPro's ability to generalize to unseen web interfaces.

4.2.3. Error Analysis

Analyzing the correctness of every action type, we found three significant error patterns: (a) Almost hit the target. The predicted point is outside but close to the ground truth bounding box. (b) Difficulty in using long-tailed actions, such as drag and hotkey, mainly due to insufficient training data. (c) The benchmarks, especially AITW [38], often fail to consider alternative solutions, leading to slightly inaccurate evaluations.

4.3. GUI Grounding Evaluation

We assess UIPro's GUI grounding capability after fine-tuning with the GUI understanding data in Sec. 3.2.1, using six benchmarks: **ScreenSpot** [12] and **ScreenSpot-v2** [49] involve mobile, desktop, and web scenarios, requiring mod-

Model	Size	Input Res.	FuncGnd	ScreenSpot	ScreenSpot-v2	MOTIF	RefExp	VWB EG	VWB AG
GPT-4o	-	AnyRes	9.8	17.8	20.4	30.5	21.8	5.6	6.8
Qwen2VL [46]	72B	AnyRes	47.7	71.4	73.2	80.3	77.7	60.5	62.1
Qwen2VL [46]	7B	AnyRes	38.7	66.4	66.9	75.1	64.8	55.9	62.1
CogAgent [20]	18B	1120	29.3	47.4	49.2	46.7	35.0	55.7	59.2
SeeClick [12]	10B	448	19.8	53.4	54.0	11.1	58.1	39.2	27.2
Ferret-UI [54]	8B	AnyRes	1.2	7.1	7.8	15.9	5.5	3.9	1.9
UGround [18]	7B	AnyRes	48.8	74.8	76.5	72.4	73.6	85.2	63.1
OS-ATLAS-Base [49]	7B	AnyRes	52.1	82.5	84.1	78.8	66.5	82.6	69.9
UIPro-Qwen2VL (ours)	7B	AnyRes	**58.8**	**82.5**	**86.9**	**80.6**	**81.9**	**94.9**	**70.9**
Qwen2-VL [54]	2B	AnyRes	7.1	17.9	18.6	28.8	29.2	17.9	17.5
UIPro-SLiME (ours)	3B	AnyRes	**58.3**	**60.7**	**61.1**	**73.3**	**59.0**	**60.0**	**40.8**

Table 6. **Comparison on the GUI element grounding benchmarks.** UIPro achieves impressive grounding accuracy, especially on FuncPred, RefExp, and VWB EG. AnyRes means using an image division strategy to handle images with variable resolutions.

els to locate elements based on brief descriptions. **MO-TIF** [7] prompts models to locate targets based on language intents in mobile apps. **RefExp** [3] locates elements on mobile devices given action intents. **VisualWebBench** [29] provides element and action grounding tasks in web environments. **FuncPred** [22] features challenging tasks requiring models to locate elements specified by functionality descriptions. Task examples are visualized in Fig. 7 of the Appendix.

We report grounding accuracy (%): Acc = $\sum_{i=1}^{N} \mathbf{1} \left(\text{pred}_i \text{ inside GT bbox}_i \right) / N \times 100$ where $\mathbf{1}$ is an indicator function and N the number of test samples. This formula calculates the percentage of samples for which the predicted points fall within the elements' bounding boxes.

Tab. 6 shows that UIPro demonstrates higher accuracy on the GUI grounding benchmarks compared to competing models UIPro-Qwen2VL outperforms the previous leading model, OS-ATLAS [49], with improvements of **15.4** and **12.3** on RefExp and VWB EG, respectively. Notably, OS-ATLAS is fine-tuned with 13.8M element annotations, significantly more than the 4.4M used for UIPro-Qwen2VL. Despite being only one-fifth the size, UIPro-SLiME surpasses CogAgent [20] across benchmarks except for VWB AG. Moreover, UIPro outperforms the other models on FuncPred [59], benefiting from the incorporation of more functionality grounding tasks in the training data (see Sec. 3.2.1). UIPro also achieves higher accuracy on the comprehensive ScreenSpot benchmark, with impressive performance on icon grounding tasks, shown in the Tab. 15.

Overall, the results demonstrate UIPro's strong grounding capability across mobile and web platforms.

4.4. Ablations And Analysis

This subsection examines the effectiveness of the 20.6M GUI understanding data (Sec. 3.2.1), the unified action space (Sec. 3.2.2), scaling effects, and the denoising approach (Sec. 3.2.3). UIPro-SLiME is used for ablation unless other-
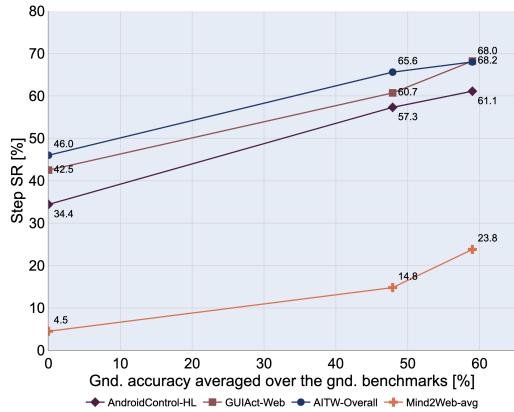

Figure 4. **Correlation between agent task performance and grounding accuracy.** UIPro-SLiME is pre-trained with 0, 5.9M, and 20.6M GUI understanding data, respectively, and then fine-tuned on the GUI agent tasks. We can see that higher grounding accuracy after pre-training leads to higher step SR after fine-tuning.

wise stated.

Impact of GUI Understanding Pre-Training To demonstrate the benefits of the collected 20.6M GUI understanding data for downstream GUI agent task fine-tuning, we compare UIPro with two variants: one pre-trained without any GUI understanding data and another with only a 5.9M subset. The results in Fig. 4 show that without pre-training on GUI understanding data, UIPro exhibits significantly lower Step SR across all benchmarks. Increasing the pre-training data amount not only improves the average GUI grounding accuracy but also increases the downstream agent task performance. These findings suggest that UIPro achieves stronger task performance when fine-tuned on a foundation with higher grounding accuracy, consistent with SeeClick [12].
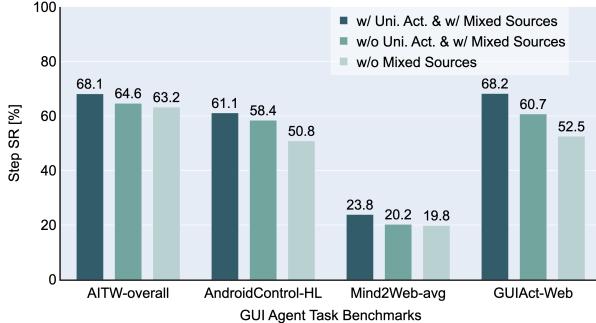
Figure 5. **Ablation study of the unified action spaces.** Fine-tuning UIPro with solely the training split of each benchmark (w/o Mixed Sources) is significantly inferior to the full UIPro. When mixing the data sources without unifying the heterogenous action spaces, UIPro witnesses clear decreases on all the benchmarks.

Ablations of Unified Action Space To assess the impact of the unified action space during GUI agent task fine-tuning, we mix the GUI agent task sources without unifying the action spaces for fine-tuning. Specifically, we retain the action definitions, including names and arguments, for each data source. We also compare a variant that uses only the training set of each GUI agent benchmark for fine-tuning.

The results in Fig. 5 show that mixing the different data sources achieves higher step SR than fine-tuning solely with the benchmark's training split. Without unifying the action spaces of the data sources, UIPro experiences notable performance declines across the benchmarks. Analysis of cases where UIPro without unified action space failed, while the full UIPro succeeded, revealed two major causes of the decrease: 1) significantly lower action type accuracy due to action definition conflicts; 2) lower accuracy of predicting swiping directions due to inconsistent swipe usage. These results suggest that unifying the action space better unleashes the potential of diverse GUI agent task data sources.

Why Unified Action Space Works. To understand how the unified action space improves performance on agent tasks, we analyze UIPro-7B's improvements on AndroidControl [23] by distinguishing between common and uncommon actions. Table 7 demonstrates that fine-tuning with unified action space data consistently outperforms training exclusively on the benchmark's training set. Crucially, accuracy increases even for the ***Wait*** action—which is unique to AndroidControl—suggesting benefits extend beyond common actions. We attribute this improvement to two mechanisms: (1) cross-task knowledge transfer, where exposure to diverse GUI transitions improves the model's ability to identify appropriate waiting contexts; and (2) training regularization from heterogeneous data sources, which enhances model robustness. These findings indicate that action unification mitigates rather than exacerbates overfitting to common actions.

Variant	Android Control - High				GUIAct		
	Click	Swipe	Navigate Back	Wait	Click	Swipe	Answer
UIPro w/ benchmark training set	45.1	22.6	33.2	43.6	42.8	40.2	12.6
UIPro w/ Uni. Act. Space (full)	**57.2**	**50.9**	**50.7**	**73.4**	**48.1**	**57.0**	**43.0**

Table 7. Compare UIPro-7B fine-tuned with only the AndroidControl training set and with our unified dataset. Unified action benefits common and uncommon actions. The metric is the prediction accuracy for each action type.
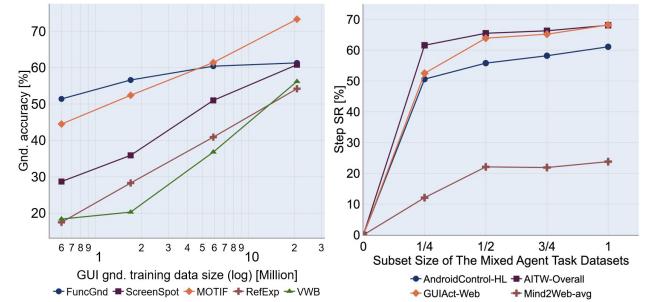

Figure 6. **Scaling effects of the GUI grounding data and mixed agent task data.** Left: increasing the data size of the GUI grounding data consistently improves grounding accuracy. Right: After pre-training on the 20.6M grounding data, fine-tuning UIPro with an increasing amount of the mixed GUI agent task data leads to higher step SR. Note that the step SR is zero as UIPro has not learned the output format required by the agent tasks.

Variants	FuncPred	ScreenSpot	MOTIF	RefExp	VWB EG	VWB AG
w/ Denoise	66.7	56.4	69.2	45.8	52.3	34.0
w/o Denoise	64.3	55.7	66.9	43.5	50.6	32.0

Table 8. **Efficacy of the proposed denoising procedure.** UIPro-SLiME is pre-trained using 6.7M GUI understanding tasks with and without denoising. We can see that denoising contributes to notable grounding accuracy gains over all the benchmarks.

Scaling Effects of GUI Interaction Data. (a) On GUI Grounding. To investigate how UIPro's grounding capability evolves during pre-training, we assess the training checkpoints at four data size checkpoints: 0.6M, 1.7M, 5.9M, and 20.6M. Fig. 6 demonstrates that increasing the data size consistently improves the grounding accuracy. The scaling effect on the challenging FuncPred [59] is weaker as functionality annotations are more costly to acquire. **(b) On GUI Agent Task.** We also examine the scaling effects on mixed agent task data by fine-tuning UIPro-SLiME with $1/4$, $1/2$, $3/4$, and the full dataset. Fig. 6 illustrates clear increasing trends on the four benchmarks.

Efficacy of Denoising To validate the denoising procedure in Sec. 3.2.3, we pre-train UIPro with 6.7M GUI understanding tasks with and without denoising. Tab. 8 highlights the importance and effectiveness of our denoising approach.

8

5. Conclusion

This paper introduces UIPro, a generalist GUI agent with superior GUI interaction capability. By curating extensive multi-platform and multi-task GUI interaction data and the proposed unified action space, UIPro exhibits superior performance on multiple GUI interaction and grounding benchmarks. We hope our curation programs and the cleaned dataset will facilitate further research and development in the GUI agent domain.

6. Acknowledgments

References

[1] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023. 3

[2] Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cǎrbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. Screenai: a vision-language model for ui and infographics understanding. In *IJCAI*, pages 3058–3068, 2024. 2, 3

[3] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. Uibert: Learning generic multimodal representations for ui understanding. In *IJCAI*, 2021. 2, 3, 7

[4] Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *NIPS*, 37:12461–12495, 2024. 2

[5] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. 2023. 6

[6] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. 2

[7] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A. Plummer. A dataset for interactive vision-language navigation with unknown command feasibility. In *ECCV*, 2022. 2, 3, 7, 1, 6

[8] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents, 2024. 1, 2, 6

[9] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xin Xia, Liming Zhu, John Grundy, and Jinshui Wang. Wireframe-based ui design search through image autoencoder. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 29(3):1–31, 2020. 2, 4, 1, 3, 6

[10] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Guicourse: From general vision language models to versatile gui agents, 2024. 1, 2, 3, 5, 6

[11] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, pages 24185–24198, 2024. 6

[12] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *ACL*, pages 9313–9332, 2024. 1, 2, 3, 5, 6, 7

[13] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 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, 2017. 2, 1, 6

[14] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web:

Towards a generalist agent for the web. *NIPS*, 36, 2024. 2, 3, 6

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3

[16] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. In *ICLR*, 2024. 2

[17] Longxi Gao, Li Zhang, Shihe Wang, Shangguang Wang, Yuanchun Li, and Mengwei Xu. Mobileviews: A large-scale mobile gui dataset, 2024. 2, 1, 6

[18] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *ICLR*, 2025. 2, 3, 6, 7, 5

[19] Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *ICLR*, 2024. 2

[20] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents. In *CVPR*, pages 14281–14290, 2024. 1, 2, 3, 7, 6

[21] Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *ECCV*, page 161–178, Berlin, Heidelberg, 2024. Springer-Verlag. 1, 3, 5, 6

[22] Hongxin Li, Jingfan Chen, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. Autogui: Scaling gui grounding with automatic functionality annotations from llms. In *ACL*, 2025. 3, 7, 1

[23] Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. In *NIPS*, pages 92130–92154. Curran Associates, Inc., 2024. 3, 4, 5, 6, 8, 1, 2

[24] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile UI action sequences. In *ACL*, pages 8198–8210, Online, 2020. Association for Computational Linguistics. 2, 3

[25] Y. Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. In *EMNLP*, 2020. 2, 3

[26] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *CVPR*, pages 19498–19508, 2025. 5, 6

[27] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NIPS*, pages 34892–34916. Curran Associates, Inc., 2023. 3, 5

[28] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024. 2

[29] Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. Visualwebbench: How far have multimodal LLMs evolved in web page understanding and grounding? In *First Conference on Language Modeling*, 2024. 2, 7

[30] Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. Harnessing webpage UIs for text-rich visual understanding. In *ICLR*, 2025. 2, 6, 1

[31] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices, 2024. 1, 2, 6

[32] Xing Han Lu, Zdeněk Kasner, and Siva Reddy. WebLINX: Real-world website navigation with multi-turn dialogue. In *ICML*, pages 33007–33056. PMLR, 2024. 3, 1, 6

[33] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent, 2024. 2, 5, 6

[34] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. 2

[35] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NIPS*, pages 27730–27744. Curran Associates, Inc., 2022. 2

[36] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents, 2025. 2

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67, 2020. 2

[38] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control. In *NIPS*, pages 59708–59728. Curran Associates, Inc., 2023. 3, 4, 5, 6, 1, 2

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy,

Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 115:211 – 252, 2014. 2

[40] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *NIPS*, 2022. 2

[41] Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *NIPS*, 36:34354–34370, 2023. 2

[42] Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Jindong Chen, Abhanshu Sharma, and James W. W. Stout. Towards better semantic understanding of mobile interfaces. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5636–5650, Gyeongju, Republic of Korea, 2022. International Committee on Computational Linguistics. 6

[43] OpenAI Team. Gpt-4 technical report, 2024. 3

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 3

[45] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning. *The 34th Annual ACM Symposium on User Interface Software and Technology*, 2021. 2

[46] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024. 3, 5, 6, 7

[47] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *TMLR*, 2022. Survey Certification. 2

[48] Jason Wu, Siyan Wang, Siman Shen, Yi-Hao Peng, Jeffrey Nichols, and Jeffrey P Bigham. Webui: A dataset for enhancing visual ui understanding with web semantics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2023. Association for Computing Machinery. 2, 1, 6

[49] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. OS-ATLAS: Foundation action model for generalist GUI agents. In *ICLR*, 2025. 2, 3, 5, 6, 7

[50] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong.

Aguvis: Unified pure vision agents for autonomous GUI interaction. In *ICML*, 2025. 2

[51] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023. 5, 6

[52] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023. 6

[53] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *NIPS*, 35:20744–20757, 2022. 2

[54] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *ECCV*, page 240–255, Berlin, Heidelberg, 2024. Springer-Verlag. 1, 2, 3, 7

[55] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *ECCV*, pages 69–85. Springer, 2016. 5

[56] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023. 2

[57] Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 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, 2024. Association for Computational Linguistics. 1, 2, 6

[58] Yi-Fan Zhang, Qingsong Wen, Chaoyou Fu, Xue Wang, Zhang Zhang, Liang Wang, and Rong Jin. Beyond llava-hd: Diving into high-resolution large multimodal models, 2024. 3

[59] Zhuosheng Zhang and Aston Zhang. 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, 2024. Association for Computational Linguistics. 7, 8

[60] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *ICLR*, 2023. 2

UIPro: Unleashing Superior Interaction Capability For GUI Agents

Supplementary Material

7. Details of UIPro Datasets

We list the data sources used in the 20.6M GUI understanding dataset (Sec. 3.2.1) in Tab. 16 and those in the unified GUI agent task (Sec. 3.2.2) in Tab. 18.

GUI understanding data:

Common Crawl: Following [22], we select the pages from the top-200 domains in Common Crawl and design an in-house web crawler that interacts with elements rendered on the web page and collects interaction trajectories. Subsequently, we use the AutoGUI [22] pipeline to generate functionality grounding and referring tasks.

Android Emulator: We set up virtual Android phones on Android Emulator and collect interaction trajectories on GUIs, including the home page, drop-down panel, settings page, and Apps drawer. Likewise, we use the AutoGUI [22] pipeline to generate functionality grounding and referring tasks.

RICO [13]: We use the element annotations prepared by SeeClick [12] and generate element grounding and referring tasks using RICO element descriptions as referring expressions.

MobileViews [17]: This dataset provides massive GUIs recorded on 20k mobile apps.We generate text localization, OCR, intent grounding, and widget listing tasks using the GUI metadata of this dataset.

WAE [9]: This dataset also provides large-scale GUI metadata, which are originally used for assisting GUI design. Similar to MobileViews, we generate grounding and referring tasks from GUI metadata. As this data source provides accurate element properties. We also generate tasks specific to iconic elements.

WebUI [48]: This dataset also provides large-scale GUI metadata. We generate GUI understanding tasks as we do for WAE.

MultiUI [30]: This dataset provides massive GUI-related Q&A tasks, which are cleaned and incorporated into our dataset to enhance the model's capability of understanding various aspects of GUIs.

GUIEnv [10]: This dataset contains only text localization and OCR tasks, which are both cleaned and incorporated into our dataset

SeeClick-Web [12]: This dataset contains only text localization and OCR tasks in web scenarios, which are cleaned and incorporated into our dataset.

OmniAct [21]: This dataset contains element annotations on web and desktop scenarios, which are used to generate intent grounding tasks.

MOTIF [7]: This dataset contains action trajectories collected on mobile apps. We convert the actions into intent-grounding tasks.

GUI agent task data:

AITW [38]: AITW is released by Google Research, providing massive Android app interaction trajectories. We use the train/test splits provided by SeeClick [12] and incorporate the cleaned training samples into our unified agent task data.

AITZ [57]: This dataset cleans a subset of AITW [38] and uses a proprietary LLM to generate high-quality reasoning and action annotations. Given a step in AITZ, We generate a sample with a reasoning process and one without reasoning to leverage the GUI knowledge entailed in the reasoning content.

AMEX [8]: This dataset expands the General split of the AITW [38] to provide more detailed annotations for each interaction step. The quality of this dataset is high, so we directly reformat their samples and incorporate them into our unified dataset.

AndroidControl [23]: This dataset boasts massive, high-quality interaction trajectories collected over one year by Google Research. As this dataset has not provided the bounding box of target elements, we find the smallest box enclosing target points using the GUI metadata provided.

GUIOdyssey [31]: This dataset provides cross-app interaction trajectories, which can diversify our unified dataset.

WebLINX [32]: This dataset provides dialogue-format human-agent interaction trajectories. We remove all non-action steps and use the action-related steps to generate action prediction samples.

OmniAct-Desktop [21]: This dataset is used in the experiments to assess the transferability of UIPro. As each action plan provided in OmniAct-Desktop is associated with only the starting screenshot, we extract the first action in the plan to generate training and test data.

8. Implementation Details of UIPro

8.1. Training Parameters

The hyper-parameters of training UIPro-SliME and UIPro-Qwen2VL are shown in Tab. 9 and Tab. 10. All experiments are conducted with 8 L20 GPUs, each with 48GB of memory. Pre-training UIPro with the 20.6M GUI understanding data for one epoch took approximately 96 hours on the 8 L20 GPUs; Fine-tuning UIPro with the 380k unified agent task data for the mobile embodiment took approximately 9 hours; Fine-tuning UIPro with the 144.9k unified agent task data for the web embodiment took approximately 3 hours.

Table 9. The training hyper-parameters used for fine-tuning UIPro-SLiME.

Hyper-Parameter	Value
Epoch	1
Global batch size	128
#GPUs	8
Learning rate for all stages	3e-5
weight decay	0.0
ADAM Beta2	0.95
Warm-up ratio	0.03
LR scheduler	Cosine
Model max length	2048
Frozen module	ViT
DeepSpeed	ZeRO-2
Data type	BFloat16

Table 10. The training hyper-parameters used for fine-tuning UIPro-Qwen2VL.

Hyper-Parameter	Value
Epoch	1
Global batch size	128
#GPUs	8
Learning rate for all stages	3e-5
LoRA Rank	128
LoRA Alpha	256
weight decay	0.0
ADAM Beta2	0.95
Warm-up ratio	0.03
LR scheduler	Cosine
Model max length	4096
Frozen module	ViT
DeepSpeed	ZeRO-2
Data type	BFloat16

8.2. Unified Action Spaces

We unify the heterogeneous action definitions from the used data sources and generate three unified action spaces for mobile, web, and desktop scenarios, respectively. Please refer to Tab. 11, Tab. 12, and Tab. 13.

Inconsistency mainly exhibits in swipe, scroll, drag/move, and status (used to signal task completion/impossibility), with substantially different parameter definitions on AITW [38], AndroidControl [23], GUIOdyssey [31], and Mind2Web [14].

8.3. Denoising Procedure

We inspect the used data sources listed in Tab. 16 and Tab. 18, categorize typical noise in the raw GUI data, and develop the following denoising procedure:

1. **Checking the validity of element bounding box coordinates.** An element will be discarded if it is outside of the GUI screenshot or its area is zero.

2. **Removing oversized elements.** As the grounding and referring tasks in our dataset focus on elements that are leaf nodes in the GUI layout hierarchy, we also remove oversized elements that are likely parent nodes. Localizing or referring to these oversized elements probably leads to ambiguity, as the element center often falls in leaf node elements. We remove elements whose area ratios are greater than 0.65. We choose 0.65 as this threshold can achieve an empirically good tradeoff between retaining meaningful elements and removing as many noisy ones as possible.

3. **Removing extremely tiny elements.** According to Google Accessibility Help[3], an element should be large enough for reliable interaction, with a width and height of at least 48dp. Considering that low-density GUI screenshots probably exist in the used GUI data sources, the smallest size of an element is limited to $48 \times (60/160) = 18$pixels. We remove elements whose smaller size is less than this threshold.

4. **Removing blank elements.** It is a common case that the GUI rendering is incomplete due to massive content loading and rendering program bugs. An example is the Mind2Web dataset [14], which contains many incomplete web page HTML scripts. Generating GUI interaction samples from these blank elements is likely to confuse the trained models. To remove these blank elements, we calculate the color deviation using `np.std` for the element region and remove elements whose color deviation is less than 5 (also an empirical value to achieve a good tradeoff between retaining meaningful elements and removing noisy ones as many as possible).

5. **Removing duplicate boxes.** It is also a common case that multiple elements are in the same bounding box. For example, an image button may contain an image element and a button element. We retain only one of the duplicate elements to ensure the dataset's diversity.

6. **Removing invisible textual elements.** Invisible elements, e.g., a hidden menu, will confuse the model. To remove these elements, we utilize `pytesseract` (an efficient OCR toll) to detect the texts for textual elements and remove the elements for which the similarity score between the OCR outputs and their text properties is less than 22.

7. **Denoising for agent task data.** Apart from the noise in GUI grounding data, the used GUI agent task data also contains noise. We list the most occurring noise type in the GUI agent task dataset: 1) AITZ [57]: the action mentioned in the agent's reasoning content does not match the actually taken action. 2) AITW [38]: Apart from the noise recognized by other works [8, 57], we also find that in some cases, one action is repeated multiple times, leading to redundant training samples. 3) AndroidControl [8]: No bounding box associated with the interacted element. 4) Mind2Web [14]:

---

[3]https://support.google.com/accessibility/android/answer/7101858?hl=en

Action Name	Usage	Definition
click	Click on an element. The target_x and target_y denote the x and y coordinates of the target.	{"action_type": "click", "target": ({target_x},{target_y})}
long_press	long-press an element for a duration.	{"action_type": "long_Press", "target": ({target_x},{target_y})}
swipe	Simulate a real swipe action to change viewports. The start_x and start_y denote the x and y coordinates of the swipe starting point. The direction has four options: up, down, left, and right. The distance has three options: short, medium, and long.	{"action_type": "swipe", "start": ({start_x},{start_y}), "direction": "{direction}", "distance": "{distance}"}
input_text	Type texts in to an input box.	{"action_type": "input_text", "text": "{text}"}
drag	Press a finger on a target, move the finger to a destination, and finally list the finger.	{"action_type": "drag", "start": (x1,y1), "end": (x2,y2)}
enter	This action inherits from AndroidControl and Android World. It simulates pressing Enter on a keyboard.	{"action_type": "enter"}
navigate_back	Navigate to the previous GUI.	{"action_type": "navigate_back"}
navigate_home	Navigate to the home page on the mobile phone.	{"action_type": "navigate_home"}
navigate_recent	Open the window showing recently used apps.	{"action_type": "navigate_recent"}
wait	Wait for content loading. This is also inherited from AndroidControl and Android World.	{"action_type": "wait"}
status	Signal the termination of a task and return an answer if required. The goal_status has two options: "successful" denoting task completion and "infeasible" denoting an impossible task. The answer is used to contain the answer generated by the model.	{"action_type": "status", "goal_status": "{goal_status}", "answer": "{answer}"}

Table 11. The unified action space for mobile scenarios.

Action Name	Usage	Definition
click	Click on an element. The target_x and target_y denote the x and y coordinates of the target.	{"action_type": "click", "target": ({target_x},{target_y})}
scroll	Simulate a scroll action to change viewports. The direction has four options: up, down, left, and right. The distance has three options: short, medium, and long.	{"action_type": "scroll", "direction": "{direction}", "distance": "{distance}"}
input_text	Type texts into an input box.	{"action_type": "input_text", "text": "{text}"}
drag	Press a finger on a target, move the finger to a destination, and finally lift the finger.	{"action_type": "drag", "start": (x1,y1), "end": (x2,y2)}
move_to	This action simulates moving the mouse and can be used to move the pointer to a location or hover on an element.	{"action_type": "move_to", "start": (x1,y1), "end": (x2,y2)}
navigate_back	Navigate to the previous webpage.	{"action_type": "navigate_back"}
navigate_forward	Undo navigate_back.	{"action_type": "navigate_home"}
go_to	Go to a certain URL.	{"action_type": "go_to", "url": "(a certain url)"}
search_google	This action simulates directly typing a search query in the address bar and pressing Enter.	{"action_type": "search_google", "query": "(search query)"}
press_key	This action simulates pressing a key down and then releasing it. Example keys include 'enter', 'shift', arrow keys, or function keys.	{"action_type": "press_key", "key": "(key name)"}
hotkey	Press a key combination. The key_comb examples include Ctrl-S or Ctrl-Shift-1 with multiple keys combined with '-'.	{"action_type": "hotkey", "key_comb": "(key combination)"}
new_tab	Create a new tab in the web browser.	{"action_type": "new_tab"}
switch_tab	Switch to a tab specified by its index.	{"action_type": "switch_tab", "tab": "(tab index)"}
close_tab	Close the focused tab.	{"action_type": "close_tab"}
status	Signal the termination of a task and return an answer if required. The usage of its parameters is the same as mobile.	{"action_type": "status", "goal_status": "{goal_status}", "answer": "{answer}"}

Table 12. The unified action space for web scenarios.

blank interacted elements.

Our denoising procedure is designed for the data sources used. Nevertheless, one can integrate more rules and adjust the empirical thresholds to extend our procedure to more data sources.

The noise ratios for several data sources are listed in Tab. 17. We find that the noise ratios are unignorable, with 29.0% of WAE [9] elements being noisy.

9. Additional Experiments

9.1. Transfer to Desktop Environments

Transfer UIPro to OmniAct-Desktop Tasks Although UIPro is primarily pre-trained with web and mobile data, we test whether this pre-training can facilitate agent task fine-tuning on out-of-distribution desktop environments, such as Windows and Linux. We fine-tune the pre-trained UIPro with the training split of OmniAct-desktop [21] and evaluate it on the test split. Tab. 14 shows that increasing the pre-training data size consistently improves step SR, with UIPro even surpassing OS-ATLAS[49], which is pre-trained with extensive desktop-domain data and fine-tuned with the same OmniAct data.

9.2. Detailed Performance on ScreenSpot

The grounding accuracy on the three platform splits of ScreenSpot [12] is shown in Tab. 15. The results show that UIPro obtains the highest grounding accuracy and excels at icon grounding.

Action Name	Usage	Definition
click	Click on an element. The target_x and target_y denote the x and y coordinates of the target.	{"action_type": "click", "target": ({target_x},{target_y})}
right_click	Right-click on an element.	{"action_type": "right_click", "target": ({target_x},{target_y})}
double_click	Double-click on an element.	{"action_type": "double_click", "target": ({target_x},{target_y})}
scroll	Simulate a scroll action to change viewports. The direction has four options: up, down, left, and right. The distance has three options: short, medium, and long.	{"action_type": "scroll", "direction": "{direction}", "distance": "{distance}"}
input_text	Type texts in to an input box.	{"action_type": "input_text", "text": "{text}"}
drag	Press a finger on a target, move the finger to a destination, and finnaly list the finger.	{"action_type": "drag", "start": (x1,y1), "end": (x2,y2)}
move_to	This action simulates moving the mouse and can be used to move the pointer to a location or hover on an element.	{"action_type": "move_to", "start": (x1,y1), "end": (x2,y2)}
press_key	This action simulates pressing a key down and then releasing it. Example keys include 'enter', 'shift', arrow keys, or function keys.	{"action_type": "press_key", "key": "(key name)"}
hotkey	Press a key combination. The key_comb examples include Ctrl-S or Ctrl-Shift-1 with multiple keys combined with '-'.	{"action_type": "hotkey", "key_comb": "(key combination)"}
status	Signal the termination of a task and return an answer if required. The usage of its parameters is the same as mobile.	{"action_type": "status", "goal_status": "{goal_status}", "answer": "{answer}"}

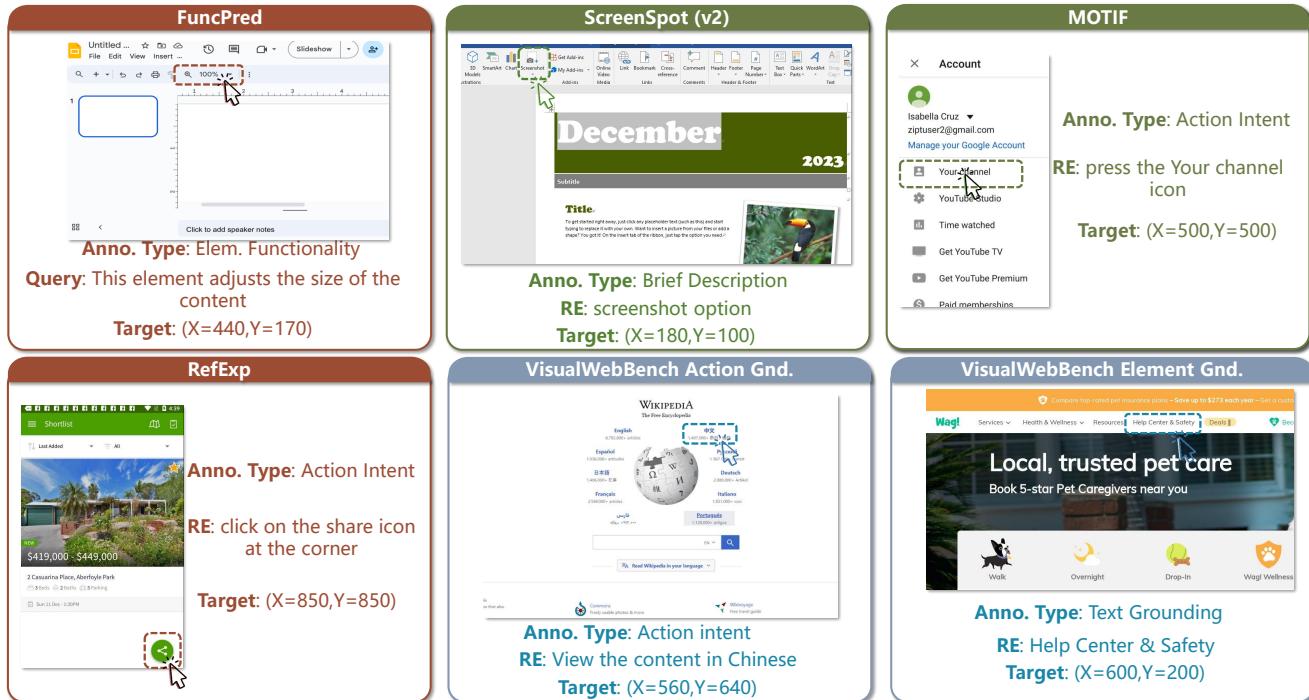Table 13. The unified action space for desktop scenarios.

Figure 7. Examples of the grounding tasks provided by the used GUI element grounding benchmarks in Sec.4.3.

Methods	Size	Step SR
DetAct (GPT-4V) [21]	-	17.0
UGround (GPT-4o) [18]	-	33.4
OS-ATLAS-Base w/ SFT [49]	7B	74.6
UIPro-Qwen2VL	7B	**77.9**
MiniCPM-GUI [10]	3B	11.3
UIPro-SLiME w/o GUI und. PT	3B	15.4
UIPro-SLiME w/ 5.9M GUI und. PT	3B	18.9
UIPro-SLiME w/ 20.6M GUI und. PT	3B	**25.1**

Table 14. **Evaluating UIPro on the desktop software tasks of OmniAct [21]**. Although training samples from desktop domains are scarce in our collected data, the two UIPro models still perform well. Pre-training UIPro-SLiME with more of our GUI understanding data before fine-tuning it on the downstream OmniAct tasks leads to better step SR. OS-ATLAS-Base w/ SFT means OS-ATLAS-Base [49] is finetuned with the OmniAct training split.

Methods	Model Size	Mobile Text	Mobile Icon	Desktop Text	Desktop Icon	Web Text	Web Icon	Avg.
GPT-4o	-	24.9	24.0	15.5	21.4	12.2	7.3	17.8
Qwen2-VL [46]	7B	78.0	62.9	64.4	50.0	72.2	61.2	66.4
CogAgent [20]	18B	68.5	21.0	73.7	17.9	70.4	33.5	49.8
SeeClick [12]	10B	77.3	52.4	68.6	30.7	59.1	30.6	53.4
UGround [18]	7B	82.8	61.6	84.0	65.1	81.3	71.8	74.8
OS-ATLAS-Base [49]	7B	93.0	72.9	**91.8**	62.9	**90.9**	74.3	82.5
UIPro-Qwen2VL (ours)	7B	**93.1**	**74.7**	89.2	**70.0**	85.7	**74.8**	**82.7**
UIPro-SLiME (ours)	3B	84.3	45.4	70.0	42.7	76.8	29.3	60.8

Table 15. Grounding accuracy on the subsets of the ScreenSpot benchmark [12]. UIPro leads in the overall performance, and achieves the highest accuracy on icon grounding tasks.

GUI Source	#Images	#Tasks	Text Gnd.	Text Ref.	Icon Gnd.	Icon Ref.	Other Element Gnd.	Other Element Ref.	Functionality Gnd.	Functionality Ref.	Intent Gnd.	Elem. Class.	Widget Listing	QA	Captioning
Common Crawl[4][†]	35.8k	1.3M	134.2k	157.7k	-	-	-	-	314.5k	314.5k	314.5k	-	48.1k	-	-
Android Emulator[†]	29.6k	629.5k	127.5k	198.1k	-	-	-	-	38.4k	38.4k	189.6k	-	37.7k	-	-
RICO [13][*]	34.4k	1.1M	-	-	-	-	939.5k	101.4k	-	-	-	-	-	-	47.2k
MobileViews [17][†]	64.6k	1.6M	497.2k	544.3k	-	-	-	-	-	-	495.3k	-	64.0k	-	-
WAE [9][†]	372.1k	7.5M	2.1M	2.4M	241.7k	313.5k	-	-	-	-	2.1M	-	345.5k	-	-
AndroidControl [23][†]	23.4k	1.1M	186.3k	262.5k	-	-	-	-	23.4k	23.4k	577.0k	-	23.4k	-	-
WebUI [48][†]	162.3k	485.2k	-	-	59.6k	113.7k	-	-	-	-	110.7k	40.9k	160.3k	-	-
MultiUI [30][*]	1.4M	5.2M	-	371.5k	-	-	684.7k	-	-	-	1.2M	-	-	2.9M	-
GUIEnv [10][*]	70.5k	680.7k	328.3k	352.3k	-	-	-	-	-	-	-	-	-	-	-
SeeClick-Web [12][*]	270.8k	1.1M	541.5k	541.5k	-	-	-	-	-	-	-	-	-	-	-
OmniAct [21][*]	176	19.1k	-	-	-	-	-	-	-	-	19.1k	-	-	-	-
MOTIF [7][*]	55	7.9k	-	-	-	-	-	-	-	-	7.9k	-	-	-	-
TOTAL	2.5M	20.6M	3.9M	4.8M	301.3k	427.3k	1.6M	101.4k	376.3k	376.3k	5.0M	40.9k	679.1k	2.9M	47.2k

Table 16. **Data sources and statistics of UI-Pro GUI understanding dataset.** Approximately two thirds of the tasks are newly generate by the authors while one third is cleaned and included from existing dataset. †means that the authors generate fine-tuning samples from unlabeled raw GUI data. * means that the authors clean the samples provided by the original datasets.

GUI Source	%Invalid Elem
Common Crawl	2.5
Android Emulator	0.4
MobileViews [17]	24.5
WAE [9]	29.0
AndroidControl [23]	11.5
WebUI [48]	14.4
MultiUI [30]	3.0
SeeClick-Web [12]	0.2

Table 17. The percentage of invalid elements detected for the used data source by the proposed denoising procedure.

Scenario	Data Source	#Used Steps	Total
Mobile	AITW [38]	37.6k	380.0k
	AITZ [57]	25.9k	
	AMEX [8]	38.7k	
	AndroidControl [23]	124.0k	
	GUIAct-smartphone [10]	64.3k	
	GUIOdyssey [31]	107.7k	
Web	Mind2Web [14]	7.7k	144.9k
	GUIAct-web [10]	109.3k	
	WebLINX [32]	22.0k	

Table 18. The number of samples included in the merged GUI agen task fine-tuning data from each data source.