SpiritSight Agent: Advanced GUI Agent with One Look

Supplementary Material

9. Extended related work

9.1. Large-scale language models

In recent years, large language models (LLMs) [14, 41, 43–45, 47, 49, 57, 59, 60, 62, 63, 69, 70] have demonstrated remarkable capabilities in the field of Natural Language Processing (NLP), encompassing natural language generation, commonsense knowledge question-answering, code completion, mathematical computation, and logical reasoning. LLM have also demonstrated strong decision-making capabilities, laying the foundation for the emergence of GUI agents.

9.2. Visual large-scale language models

With the development of large language models, numerous works [3, 9, 23, 29, 35, 58, 66, 73] have proposed vision language models (VLMs) to bring the capabilities of language models into the visual domain. CLIP [18] uses contrastive learning to align vision and language features, while BLIP [30] and BLIP-2 [31]build on this by adding a language decoder, enabling the models to perform image-grounded text generation. InternVL [10] attempts to scale the parameters of vision encoder up to 6 billion, significantly enhancing the model's ability to perceive visual input. LLaVA [37] and Sphinx [35] improve the models' understanding and chat abilities through instruction tuning and multitask learning, respectively. Beyond general domains, OCR-Free [25] methods use an encoder-decoder architecture to achieve end-to-end visual document understanding. This demonstrates the significant potential of VLMs in GUI navigation tasks.

9.3. GUI agent benchmarks

GUI agents have seen rapid development in recent years, with many types of benchmarks emerging. MiniWoB [51], MiniWoB++ [36], and WebShop [65] are early classic GUI navigation benchmarks. However, the data in these benchmarks is synthetically generated, which creates a slight gap compared to real-world data. AitW [46] is a large-scale real-world dataset designed for mobile navigation tasks, and Mind2Web [13] is a high-quality benchmark for web navigation that provide evaluation across three scenario: cross-task, cross-website, and cross-domain. ScreenSpot [11] is a functional grounding benchmark that covers mobile, web, and desktop scenarios. GUIAct [7], AMEX [5], AndroidControl [32], and GUI Odyssey [38] are newly released benchmarks designed for web and mobile environments, respectively. They are highly reliable benchmarks as they are annotated by humans and have undergone further validation. In this paper, we evaluate SpiritSight on six benchmarks across different GUI platforms and tasks: Multimodal-Mind2Web [13], ScreenSpot [11], GUIAct [5], AMEX [5], AndroidControl [32], and GUI Odyssey [38]. Overview of these benchmarks is shown in Tab. 5

10. Task formulation

For a given GUI platform, we first obtain an action space $\mathcal{A}$ that contains all operable actions. Given the task description $\mathcal{T}$, the previous actions $\mathcal{H} = \{a_1, a_2, ..., a_{t-1}\}$, the action space $\mathcal{A}$ and the current screenshot o_t, the agent is expected to infer the optimal action a_t^* that maximizes the expected future reward. The inference process is guided by a policy π, as shown below, which maps the current context to a probability distribution over the action space $\mathcal{A}$. Here, a denotes a specific action selected from the action space $\mathcal{A}$.

$$a_t^* \sim \pi(a|\mathcal{T}, \mathcal{H}, \mathcal{A}, o_t), \quad a \in \mathcal{A} \tag{7}$$

We propose a hierarchical decomposition of the policy to manage the complexity of action inference. Initially, we define s as the natural language description (*e.g.* Click on the login button.) of action a (*e.g.* CLICK(132, 243)). We decompose the overall policy π into a step inference policy π_s and an action inference policy π_a as Eq. (8). The step inference policy π_s selects the optimal s based on the current context. Once s is determined, the action inference policy π_a predicts the corresponding action a from the action space $\mathcal{A}$.

$$\pi(a|\mathcal{T}, \mathcal{H}, \mathcal{A}, o_t) = \pi_s(s|\mathcal{T}, \mathcal{H}, o_t) \cdot \pi_a(a|s, \mathcal{A}) \tag{8}$$

Further, we decompose π_a into π_{pos} and π_{attr} as Eq. (9). Here, a_{pos} denotes to the positional aspect of the action, typically the coordinates where the action is performed (*e.g.* (132, 243)), while a_{attr} denotes the non-positional aspects, such as the action type (*e.g.* click) or additional parameters like specific input text.

$$\pi_a(a|s, \mathcal{A}) = \pi_{pos}(a_{pos}|s, \mathcal{A}) \cdot \pi_{attr}(a_{attr}|s, \mathcal{A}) \tag{9}$$

Based on Eq. (8) and Eq. (9) we have

$$\begin{aligned} \pi(a|\mathcal{T}, \mathcal{H}, \mathcal{A}, o_t) = \pi_s(s|\mathcal{T}, \mathcal{H}, o_t) \cdot \\ \pi_{pos}(a_{pos}|s, \mathcal{A}) \cdot \\ \pi_{attr}(a_{attr}|s, \mathcal{A}) \end{aligned} \tag{10}$$

It is easy for vision-based agents to learn the step inference policy π_s, as recent VLMs excel at image understanding

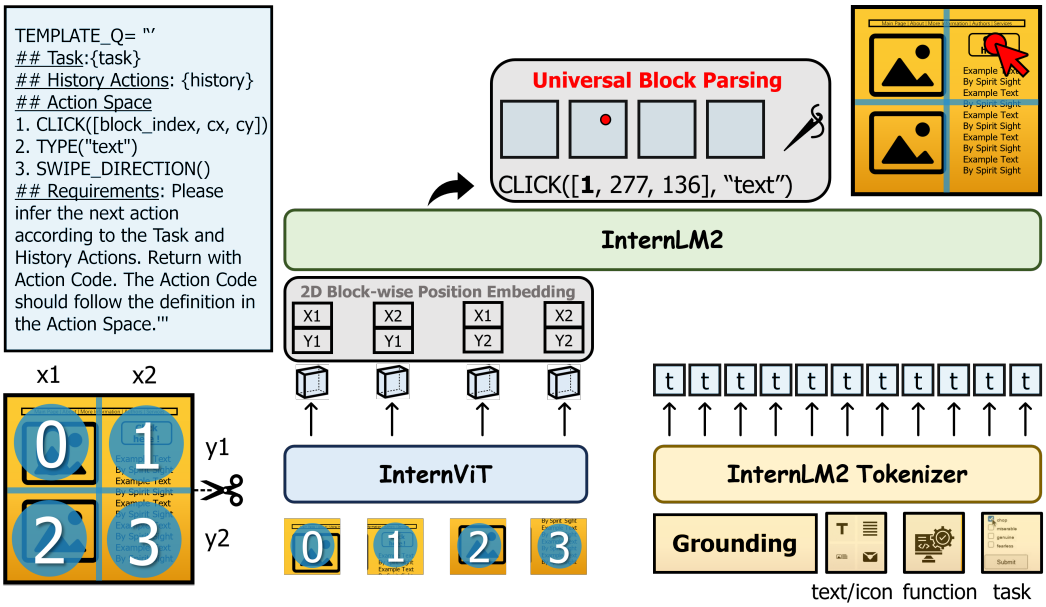


Figure 7. The overall architecture of SpiritSight. SpiritSight is pre-trained on GUI-Lasagne, a large-scale, multi-level, high quality GUI dataset. The UBP method solves the ambiguity in Dynamic High-Resolution input during model training.

and reasoning. Learning the non-positional inference policy π_{attr} is also manageable, since the non-positional aspects of an action can be directly inferred from the natural language step. For example, an action like "*INPUT('Copenhagen')*" can be directly inferred from a step such as "*Input 'Copenhagen' into the arrival input box*". The primary challenge lies in learning the positional sub-policy π_{pos} as discussed in Sec. 2. To address this challenge, we construct a large scale dataset focused primarily on grounding tasks to facilitate learning accurate positional actions.

11. Overall architecture

We build our model based on the pre-trained InternVL2, a family of advanced and open-sourced VLMs. We chose InternVL for the following reasons: (1) The large-scale and high-performance vision encoder is more capable to handle the text-rich GUI environment. (2) The dynamic resolution strategy largely preserves the details of the input screenshots, allowing for enhanced perception of fine-grained text and icon information. We take the advantage of large-scale InternViT with a large-scale GUI dataset described in Sec. 3. We further propose a Universal Block Parsing (UBP) method to resolve the ambiguity problem brought by dynamic resolution in Sec. 4.

The architecture of SpiritSight is depicted in Fig. 7. To begin with, the input image is the GUI screenshot. According to the dynamic resolution algorithm of InternVL, an appropriate ratio of input image is decided. Then, the image is divided into several blocks, each with a unique index, in preparation for the post-processing phase of our UBP method. These image blocks will be flattened into sequences before being sent to the vision encoder, which results in the loss of their 2D spatial relation. To address this problem, we introduce 2D Block-wise Position Embedding (2D-BPE) method, which maintains the blocks' 2D spatial relation by adding a row embedding and column embedding to each block. Afterwards, the embedded image features, along with the task objective, the action space and the history actions are processed by the InternLM2 decoder to infer the action code. Finally, the action and corresponding coordinate for operation is obtained by the UBP parser.

12. Experiments

12.1. Data preprocessing

The Mind2Web [13] data has very skewed aspect ratio distribution. We follow the preprocessing procedure used by SeeClick to make it compatible with more general scenarios. The images are randomly cropped around the objects at a resolution of 1280×720, therefore the final resolution does not result in a highly skewed aspect ratio.

12.2. Implementation details for benchmark experiments

We use InternVL2 (2B, 8B and 26B) [9, 25] as the base models and train them with two stages: continual pre-training and fine-tuning.

Pre-training Stage. We train all the GUI-Lasagne datasets mentioned in Sec. 3 simultaneously. Different

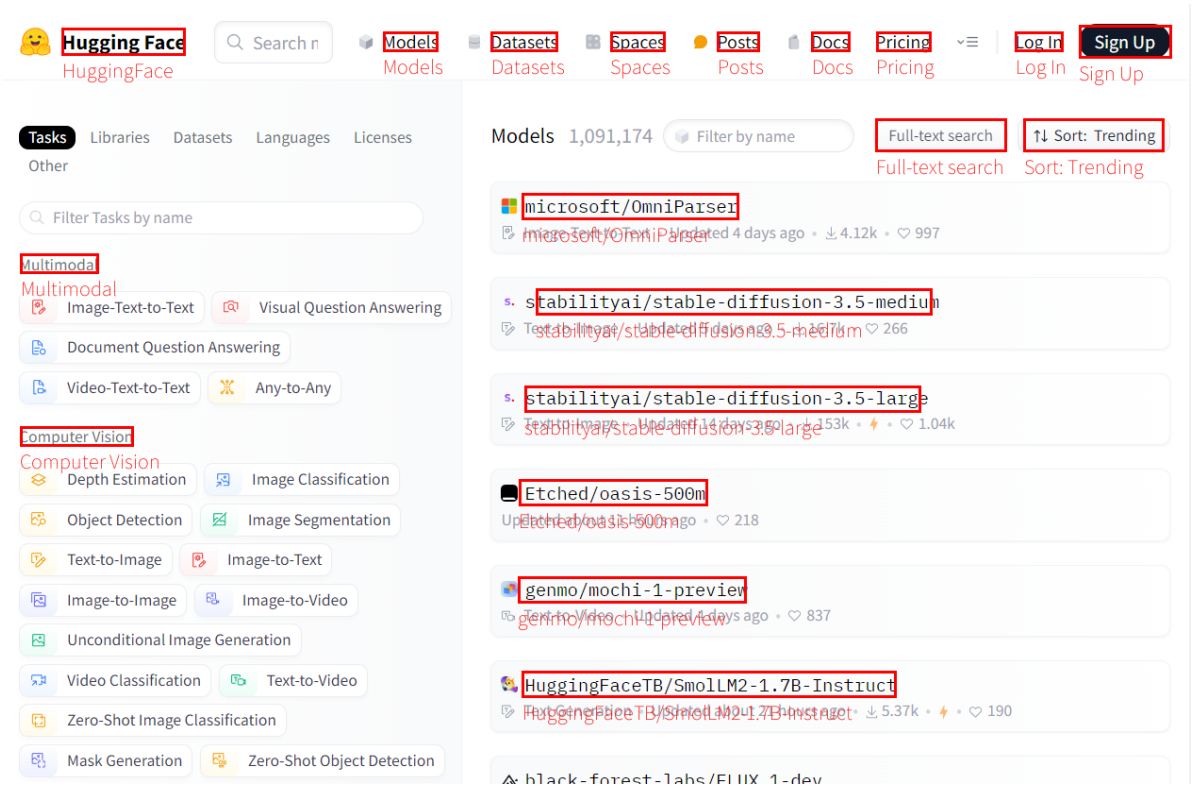


Figure 8. Visualization results of SpiritSight-2B on our custom text2bbox test set. The red boxes represent the generated results and the text next to it represent the text prompt.

prompts are designed for different training tasks to avoid task confusion. See Sec. 14 for the detailed prompt content. We unfreeze the vision encoder, decoder, and MLP layer of InternVL. The learning rate is set to 1e-4/1e-4/5e-5 for InternVL-2B/8B/26B, respectively, and the batch size is set to 1024. We get **SpiritSight-Base** models after pre-training.

Fine-Tuning Stage. We fine-tune SpiritSight-Base models in several downstream tasks separately. We define a distinct action space for each task to prevent action confusion. The max number of history actions is set to 5 to prevent excessive overload. For ScreenSpot benchmark, we follow the data proportions from [11], using part of the level-1 and level-2 data of GUI-Lasagne, and the data from [12, 33, 54] to train the entire model. For other GUI navigation benchmarks, we first train the entire model for 1 epoch using the level-3 data of GUI-Lasagne and the training data corresponding to each benchmark, then fine-tune the model for 1 epoch on the benchmark-specific training data using LoRA [21]. While training the entire model, the learning rate is set to the same as pre-training, and the batch size is 1024. During fine-tuning, the learning rate is set to 5e-5, the batch size is 64, with the alpha of vision encoder and LLM decoder set to 32 and 64, respectively.

12.3. Implementation details for ablation study

Recognition and Grounding as Priors for GUI Navigation. To verify the significance of the three levels of GUI-Lasagne data, we progressively removed level-3, level-2, and level-1 data from the training set during the pre-training stage and evaluate models on Multimodal-Mind2Web benchmark. During the fine-tuning stage, we train the SpiritSight-Base model only on the Multimodal-Mind2Web training data using LoRA, without training the whole models on level-3 data of GUI-Lasagne, as level-3 data is excluded from the pre-training datasets. The results are shown by the blue line in Fig. 6a.

We also conducted ablation experiments to evaluate the effectiveness of data cleaning and CoT construction strategies on the level-3 data, as shown by the orange line in Fig. 6a. We use the same setting as in the benchmark experiments, except with different versions of level-3 data (original version, data cleaning version and CoT version).

Better Grounding Ability from UBP. To verify the effectiveness of UBP on grounding task, we use LoRA for resource efficiency to fine-tune InternVL in 4 different settings (original as baseline, 2D-BPE, UBP, and 2D-BPE&UBP), respectively. First, We fine-tune InternVL1.5 (26B) on 10% of our GUI-Lasagne dataset. The alpha is

Benchmarks	Platforms	Task	Metric	# Test Samples	History?
ScreenSpot[11]	Web, PC, Mobile	Functional Grounding	ClickAcc	1,272	×
Mind2Web[13]	Web	Navigation	Ele.Acc, Op.F1, Step SR	6,418	✓
AMEX[5]	Mobile	Navigation	AMS	5,284	✓
GUI-Odyssey[38]	Mobile	Navigation	AMS	29,426	✓
AndroidControl-High[32]	Mobile	Navigation	Step Accuracy	8,444	✓
AndroidControl-Low[32]	Mobile	Functional Grounding	Step Accuracy	8,444	×
GUIAct-Multi[7]	Web	Navigation	StepSR	1,065	✓
GUIAct-Single[7]	Web	Functional Grounding	StepSR	1,410	×

Table 5. Statistics of GUI benchmarks we include in this paper.

Training data	task	website	domain
SeeClick (1M)	25.5	16.4	20.8
GUI-Lasagne* (1M)	**28.0**	**17.7**	**21.8**

Table 6. The comparison of SeeClick training data and our GUI-Lasagne data on Qwen-VL on Mind2Web benchmark.

	Mind2Web	AndroidCtrl	GUIAct
w/o UBP	25.3	60.4	37.8
w/ UBP	**30.8**	**62.1**	**40.6**

Table 7. The comparison of the results of the baseline and UBP on different benchmarks.

set to 64. Then, we fine-tune the model on the Multimodal-Mind2Web training data. The alpha is set to 16 and 32 for vision encoder and LLM decoder, respectively.

Scaling Effects on Dataset and Model Size. We explore the impact of pre-training dataset and model size on SpiritSight using Multimodal-Mind2Web benchmark. The training strategies are same as in the benchmark experiments, except with different scale of pre-training data.

Effective Transfer to other languages. We split the training and test sets of GUIAct(web-multi) dataset into English and Chinese partitions, respectively. We fine-tune SpiritSight-Base (8B) on two sets of data: the entire training set (English&Chinese) and the English-only training set. Other training strategies are same as in the benchmark experiments.

12.4. Metrics

We use the metrics proposed in the corresponding benchmarks as shown in Tab. 5. Although they have different names, these metrics are similar: GUI grounding tasks consistently measure the hit rate of predicted bounding boxes, while GUI navigation tasks focus on single-step accuracy.

Click Accuracy. The proportion of test samples where the predicted location falls in the ground truth element bounding box.

Element Accuracy (Ele.Acc). Comparing the selected element with all acceptable elements. For vision-based methods, it is the same as Click Accuracy.

Operation F1 (Op.F1). Token-level F1 score for the predicted operation.

Step Success Rate (Step SR) & Step Accuracy. The proportion of successful steps. A step is regarded as successful only if both the selected element and the predicted operation are correct.

Action Matching Score (AMS). The proportion of predicted actions that match the ground-truth actions. Two actions can match if their action types are equal. For dual-point taps, they are considered equal if they fall within a 14% screen distance from each other. Alternatively, if the tap actions occur within the same detected bounding boxes, where the bounding boxes are augmented to 240% of their total size, they are considered equal. Finally, two dual-point scrolls are considered equal if they have the same primary scroll axis (vertical or horizontal).

12.5. Comparison between GUI-Lasagne and existing dataset

In Sec. 6.3, we compare our GUI-Lasagne data and SeeClick training data [11] by training InternVL-2 (8B) on the two datasets respectively. To make the experiment more comprehensive, we also compare these two datasets on Qwen-VL [3], which is the backbone of SeeClick. To make a more fair comparison, we create a subset randomly sampled from GUI-Lasagne, matching the scale of the original SeeClick training data (1M). As shown in Sec. 12.2, our GUI-Lasagne data demonstrates high quality, as using a similar scale of data leads to improvements over previous methods. It also shows that GUI-Lasagne generalizes well across different architectures, including Qwen-VL and InternVL. Experiments with other existing methods' train-

	Dataset	Platform	# Samples	# Tokens	# Elements	# Screenshots
DocVQA	Ureader-Instruction	General	489,150	23,356,600	489,150	118,355
	GUIChat	Web	50,832	14,789,523	50,832	17,979
	WebSRC	Web	23,742	162,177	23,742	5,462
	ScreenQA	Mobile	11,781	29,897	11,781	66,261†
	RICO screen-captioning	Mobile	15,743	121,854	15,743	66,261†
Level1	Web bbox2dom	Web	862,505	281,389,127	862,505	755,499*
	Web text2bbox	Web	736,826	202,821,759	11,959,607	755,499*
	Web bbox2text	Web	267,955	48,760,131	5,118,237	755,499*
	AITW text2bbox	Mobile	1,058,638	309,594,824	27,993,054	1,276,752‡
	RICO widget-captioning	Mobile	28,818	2,059,165	179,144	66,261†
Level2	Web function2bbox	Web	906,087	156,273,895	9,710,488	755,499*
	AITW function2bbox	Mobile	620,736	18,542,781	620,736	1,276,752‡
	RICO-SCA	Mobile	18,148	2,485,239	145,517	66,261†
Level3	AITW w/ CoT	Mobile	639,535	53,039,652	639,535	1,276,752‡
Total			5,730,496	1,113,426,624	57,820,071	2,240,308

Table 8. Statistics of our GUI-Lasagne, a GUI continual pre-training dataset for our SpiritSight-Base Model. In the '# Screenshots' column, several datasets share the same suite of screenshot images, so numbers marked with the same superscript notation are counted only once.

ing data are not included, since they have not been fully released yet.

12.6. Ablation study with UBP

We extend the ablations for UBP mentioned in Sec. 6.4 using different training datasets including Mind2Web, AndroidControl, and GUIAct. The results are shown in Sec. 12.4, where UBP consistently outperforms the baseline across different benchmarks. The results demonstrate the effectiveness of our proposed UBP method.

12.7. Grounding abilities for GUI visual appearances

To evaluate the foundational ability of SpiritSight to ground visual appearance, we construct a small benchmark for text2bbox task. We random select a small number of URLs from those mentioned in Sec. 3.1. These URLs are not used in constructing the pre-training data. Following the method described in Sec. 3.1, we construct a test set with 3,700 text2bbox pairs. We adopt the hit rate as metric, defined to be the proportion of test samples where the model predicted location falls within the ground-truth bounding boxes. Ultimately, SpiritSight-2B achieves a **96.1%** hit rate on this benchmark, demonstrating its strong capability in fundamental grounding tasks. Fig. 8 shows the visualization of the predicted bounding boxes from SpiritSight-2B.

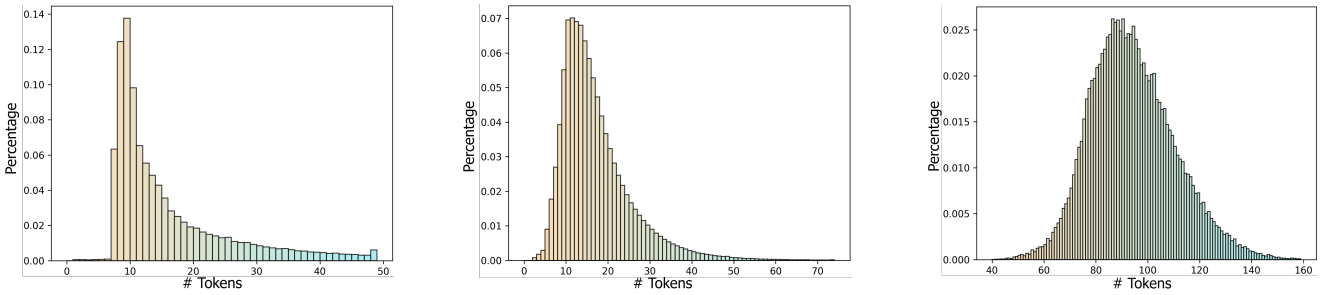
13. Data collection

In this section, we present a cost-effective data collection strategy designed to construct a multi-level, large-scale and high-quality GUI dataset, called **GUI-Lasagne**. This dataset helps equip our models with robust abilities in GUI understanding, grounding, and navigation. The statistics of GUI-Lasagne are shown in Tab. 8 and Fig. 9.

13.1. Level one: visual-text alignment

We collected website URLs from two sources: the CommonCrawl [16] dataset and website ranking sources. We used the URLs from website rankings as a supplement to CommonCrawl due to its compromised quality, which includes a large proportion of blank pages, sparse-texted pages, and dead links. We then developed a data collection tool using playwright library to get real-world web data from the collected URLs.

For each URL, we navigate to the webpage and start data collection only after the webpage has fully loaded. We collect both the webpage screenshots and the corresponding DOM tree according to a carefully designed scheme. First, we perform grid sampling on the screen with a step size of 8 pixels. Then, we mark the element objects corresponding to the sampled points. Finally, we apply an HTML pruning algorithm to simplify the HTML code by retaining all the marked elements and their parent nodes. This process excludes elements that are small in size or invisible on the screen. Additionally, we label all the clickable elements by checking their pointer property and registered events. The



(a) GUI-Lasagne Level 1. (b) GUI-Lasagne Level 2. (c) GUI-Lasagne Level 3.

Figure 9. The distribution of token numbers of our GUI-Lasagne dataset for GUI continual pretraining.

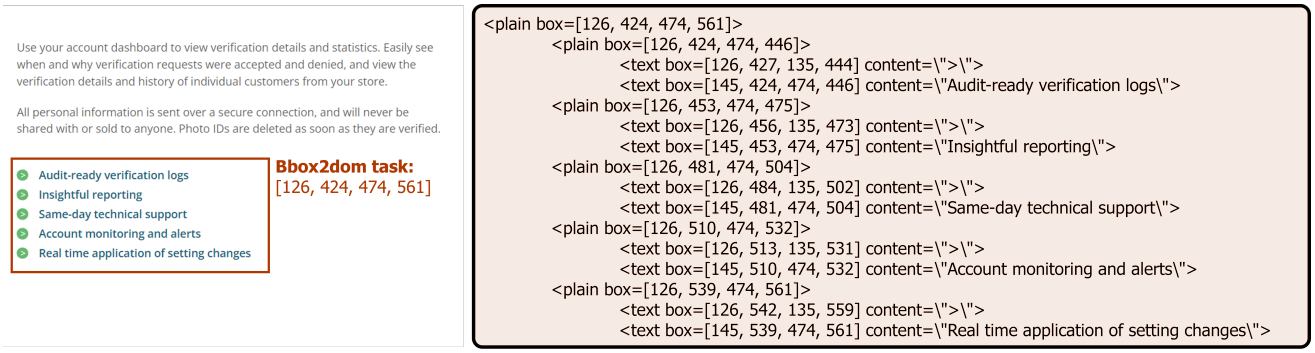


Figure 10. An example of the Bbox2dom task. Left shows a given bounding box on a web page, right shows its corresponding simplified DOM structure.

resulting DOM trees are used to construct the bbox2dom pairs, while the element objects are utilized to create the text2bbox and bbox2text pairs.

After collecting data from the current website, we acquire new pages using two methods: scrolling down or clicking on an element, with these choices being randomly sampled. If clicking on an element is chosen, the target element is also randomly sampled from all clickable elements. We collect a maximum of 30 pages for each URL. We repeat the above mentioned process to achieve an automated data collection. Ultimately, we collect 755K webpage screenshots along with their DOM trees, where English samples account for 3/4 of the data and Chinese samples account for 1/4.

13.2. Level two: visual-text alignment

We leverage InternVL's image understanding capabilities to collect function grounding data. Specifically, we divide each screenshot into a 3x3 grid and describe the approximate location of the target element in text format (*e.g.* in the top-left corner of the image). Additionally, We place a bounding box around the target element in the screenshot to precisely specify its location. To prevent color confusion, we dynamically determine the color of the bounding box. First, we analyze the color data around the target element, then select the most visually prominent color among red, green, and blue as the color of bounding box. By providing InternVL2-26B with the screenshot, the element's text content or icon caption, and the location description, we prompt it to generate the function of the target element. Additionally, we utilize InternLM2.5-20B to enhance the quality and diversity of the generated function descriptions. The two prompts are shown in Sec. 15.2.

A validation is performed by two experienced human annotators[5]. Specifically, we randomly sampled 100 images from the collected data to create a human evaluation set, with the functional description of all labeled element. The annotators are asked to determine whether the functional description is correct. A functional description is considered acceptable if the corresponding element can be uniquely identified in the screenshot based on the description. We calculate the proportion of acceptable functional descriptions out of the total descriptions. Ultimately, the human evaluation achieves an acceptance rate of 90.9%, indicating the effectiveness of our data synthesis strategy. We show some evaluation examples in Fig. 11.

[5]"Experienced" means that the annotators are well trained in labeling and are familiar with GUI tasks.

13.3. Level three: visual GUI navigation

We utilize the public available AitW [46] dataset to construct our GUI navigation training data. AitW is a large-scale mobile navigation dataset where each screenshot is labeled with the corresponding goal, the current step, *etc.* We select the all general, install and web-shopping sets and 1M samples of google-apps set as the source data. We discard the single set as the screenshots are duplicated with others. However, AitW involves a certain amount of incorrectly labeled samples as mentioned by AitZ [72] and AMEX [5].

We clean the AitW [46] dataset with GPT-4o and adopt Chain-of-Thought (CoT) [60] to make the judgment more accurate. Specifically, for non-final steps, we prompt GPT-4o with the task description, the current action annotation, two screenshots at the current and the next steps, respectively. GPT-4o is then instructed to first summarize the two screenshots and identify the differences between them, then describe the current step based on these differences, and finally assess the reasonableness of the current action annotation. We filter out steps identified as unreasonable by GPT-4o. For the final step, we prompt GPT-4o with the task description and the current screenshot. It is then instructed to summarize the screenshot and determine whether the task was successfully completed. We filter out the final steps considered successfully completed. Note that we only discard the steps that do not meet the requirements, and do not discard the entire trajectories. The prompts for GPT-4o are shown in Sec. 15.3. The collected data examples are shown in Fig. 11. Ultimately, we obtain 0.63M CoT-style GUI navigation training samples from 1.48M source samples after cleaning.

We also perform a validation for level-3 data by two experienced human annotators. Specifically, we randomly sampled 100 steps that were considered reasonable (the Cleaned Set) and 100 steps discarded by GPT-4o (the Discarded Set). For each step sample, the annotators are provided with the screenshots, the overall task description, and the validity judgments generated by GPT-4o. Then they are asked to determine whether the results of GPT-4o is correct. We report the true positive rate (TPR) for the Cleaned Set and the true negative rate (TNR) for the Discarded Set. The Cleaned Set achieved a TPR of 93.7%, indicating the reliability of our data cleaning procedure. The Discarded Set achieved a TNR of 76.3%. Though the result is not as high, it is unrelated to the quality of our dataset. In the future, we will explore a more efficient data cleaning method to improve the TNR while keeping the TPR approximately unchanged.

14. Training data format

We constructed a large scale dataset for GUI continual pretraining, including text2bbox, bbox2text, bbox2dom, and function2bbox tasks. To make sufficient use of the context length of the model, we pack multiple data pairs in each training sample for text2bbox, bbox2text and function2bbox tasks, and select the box that includes as many elements as possible for bbox2dom task. We use the center point, width and height to represent a bounding box. It is worth noting that, aside from the function2bbox task, we add an additional block index to each bounding box, which is derived from our proposed UBP method. For function2bbox task, we use the original global coordinate system as the bounding boxes are too large to be considered a point and grounding is not the main focus of this task. Additionally, we normalize all coordinate values between 0 and 999 and round them to the nearest integer. Below are the training data templates for each task. Notably, the prompt is randomly selected from a pool during data construction. See Sec. 15.1 for details of the prompt pool.

Data Format for text2bbox Task
user:
<image>
1.{text 1}
2.{text 2}
3.{text 3}
...
Provide the bounding boxes of each given text in a list format.

assistant:
1.{[block-index, cx, cy, w, h]}
2.{[block-index, cx, cy, w, h]}
3.{[block-index, cx, cy, w, h]}
...

Data Format for bbox2text Task
user:
<image>
1.{[block-index, cx, cy, w, h]}
2.{[block-index, cx, cy, w, h]}
3.{[block-index, cx, cy, w, h]}
...
Provide the text content of each given bounding box in a list format.

assistant:
1.{text 1}
2.{text 2}
3.{text 3}
...

Data Format for bbox2dom Task
user:
<image>
I'd like some information about the specific region [cx, cy, w, h] in the image.

assistant:
{DOM_Tree}

Data Format for function2bbox Task
user:
<image>
1.{function description 1}
2.{function description 2}
3.{function description 3}
...
In this image from a webpage, find out where to click for a certain need and provide bbox coordinates in a list format.

assistant
1.{[block-index, cx, cy, w, h]}
2.{[block-index, cx, cy, w, h]}
3.{[block-index, cx, cy, w, h]}
...

15. Prompt templates

15.1. Evaluation inference

Prompt for Evaluation Inference
Task: {task}
History Actions:
{history}
Action Space
{Action Space}
Requirements: Please infer the next action according to the Task and History Actions.
Return with Action Code. The Action Code should follow the definition in the Action Space.

15.2. Level-two function generation

Prompt for Level-two Function Generation
Please infer the purpose of the operation "click on the '{text}' on the {region} of the webpage" based on the webpage.
Please deliver the purpose specifically and clearly, which points to the certain item.
Its direct context includes the following information: {context_text}.
Please make the answer only in English.
Let's think step by step.
Your final answer should be in a new line and included in double quotation like:
The purpose is "xxx".

Prompt for Level-two Function Augmentation
Can you rewrite the original purpose "{purpose}" into a short phrase?
Here are some examples:
{Few-shot example 1}
{Few-shot example 2}
{Few-shot example 3}
Output only the refined purpose, start with 'to', without any explanation.

15.3. Level-three data processing

System Prompt for Level-three Data Processing
You are a mobile operation assistant, the main goal is to help identify whether the mobile navigation operation is correct.

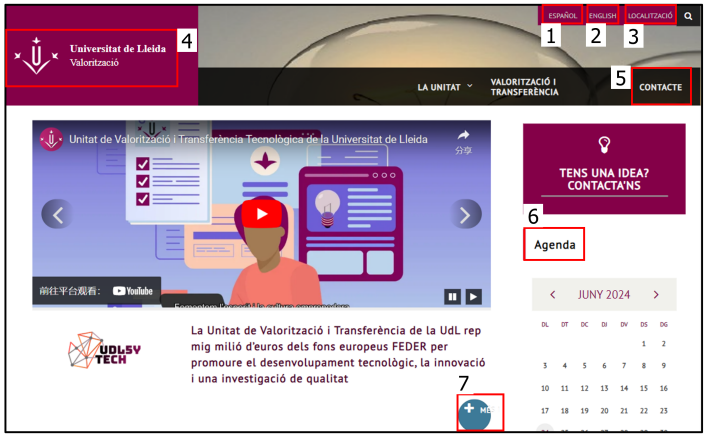
Prompt for Level-three Middle Step Data Processing

Task: {task}
Action History: {history}
The Current Action: {action}
You are completing a mobile task and now in step {step_idx}. Picture 1 shows the current screen with action demonstration and picture 2 shows the screen after performing The Current Action on picture 1. You are also given the Action History before the Current Action.
Return:
1. Summarize picture 1 about its main content and its functionality. Also describe the changes that have occurred in Figure 2 compared to Figure 1. Describe them with necessary details, but not too long.
2. Based on the changes between Figure 1 and Figure 2, estimate the function of the Current Action. Return with format of "The function of the Current Action: xxx"
3. Analyze the rationality of the Current Action based on the Task. Return only the reason.
4. Return the final answer of the rationality of the Current Action with just 'True' or 'False'.
5. Analyze if the Task is successfully completed. Return only the reason.
6. Return the final answer of the complementarity of the Task with just 'True' or 'False'.

Prompt for Level-three Last Step Data Processing

Task: {task}
Action History: {history}
You have just completed a mobile task with a series of actions listed in Action History. The picture shows the final screen of the mobile.
Return:
1. Summarize the picture about its main content and its functionality. Describe it with necessary details, but not too long.
2. Analyze if the task is successfully completed from the perspectives of success and completion separately.
3. Return the final answer of the analysis with just 'True' or 'False'.

1: ✅ To provide daily temperature and forecast with activity and outfit suggestions.
2: ✅ To provide a list of recommended outdoor activities based on the current weather conditions in Victoria, BC.
3: ✅ To recommend appropriate clothing attire based on the current weather conditions.
4: ✅ To get tailored morning run advice
5: ✅ To obtain personalized advice on the optimal timing for walking activities.
6: ✅ To view the weather forecast for the full 72 hours
7: ❌ To access detailed weather for a specific time
8: ✅ To view the hourly weather forecast for Victoria, BC at 4am.
9: ✅ To check the weather forecast for 5am in Victoria, BC
10: ✅ To access more detailed weather information for the specific time slot of "6am".
11: ✅ To view detailed weather information for the 7 a.m. hour in Victoria, BC

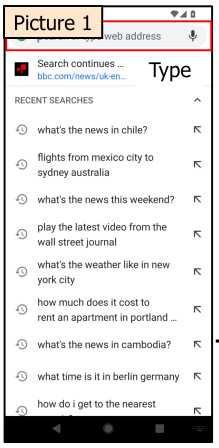


1: ✅ To switch the language of the webpage to Spanish
2: ✅ To switch the language of the webpage from Spanish to English
3: ❌ To change the language of the webpage to Catalan
4: ✅ To navigate to the main page of the Universitat de Lleida Valorització
5: ✅ To access the contact page for the University of Leiden's Valorization & Technology Transfer Office.
6: ❌ To access historical content or archived data from the specified date and time.
7: ✅ To access more information about the 500,000 euros grant from FEDER funds received by the UdL Technology Transfer and Valorization Unit to promote technological development, innovation, and quality research.

"task":
Open the downloads

"action":
Click([0.88, 0.25])

"rationale": False



"task":
What's the news in Malaysia?

"action":
Type("What's the news in Malaysia?")

"rationale": True

Picture 1 is the last screen

Task finished

"task":
What time is it in Berlin?

"complete":
True

Action reason:
The Current Action is not rational based on the Task because the task is to open the downloads, not to change the view mode.

Action reason:
The Current Action is rational because it directly addresses the task of finding out the news in Malaysia by initiating a relevant search query.

Action reason:
The Current Action is rational because it aims to bring back the main content that shows the time in Berlin, which is the information needed to complete the task.

Figure 11. The examples of our collected GUI function and navigation data. The upper two screenshots show the functional annotation generated by InternVL2 and InternLM2.5. The lower three samples show the judgment results and reasons provided by GPT-4o.