GUI-Narrator: Detecting and Captioning Computer GUI Actions

Qinchen Wu
Show Lab, National University of Singapore
Singapore, SGP
qinchen.wu62@gmail.com

Difei Gao
Show Lab, National University of Singapore
Singapore, SGP
daniel.difei.gao@gmail.com

Qinghong Lin
Show Lab, National University of Singapore
Singapore, SGP
kevin.qh.lin@gmail.com

Zhuoyu Wu
Show Lab, National University of Singapore
Singapore, SGP
wuzhuoyu11@gmail.com

Mike Zheng Shou †
Show Lab, National University of Singapore
Singapore, SGP
mike.zheng.shou@gmail.com

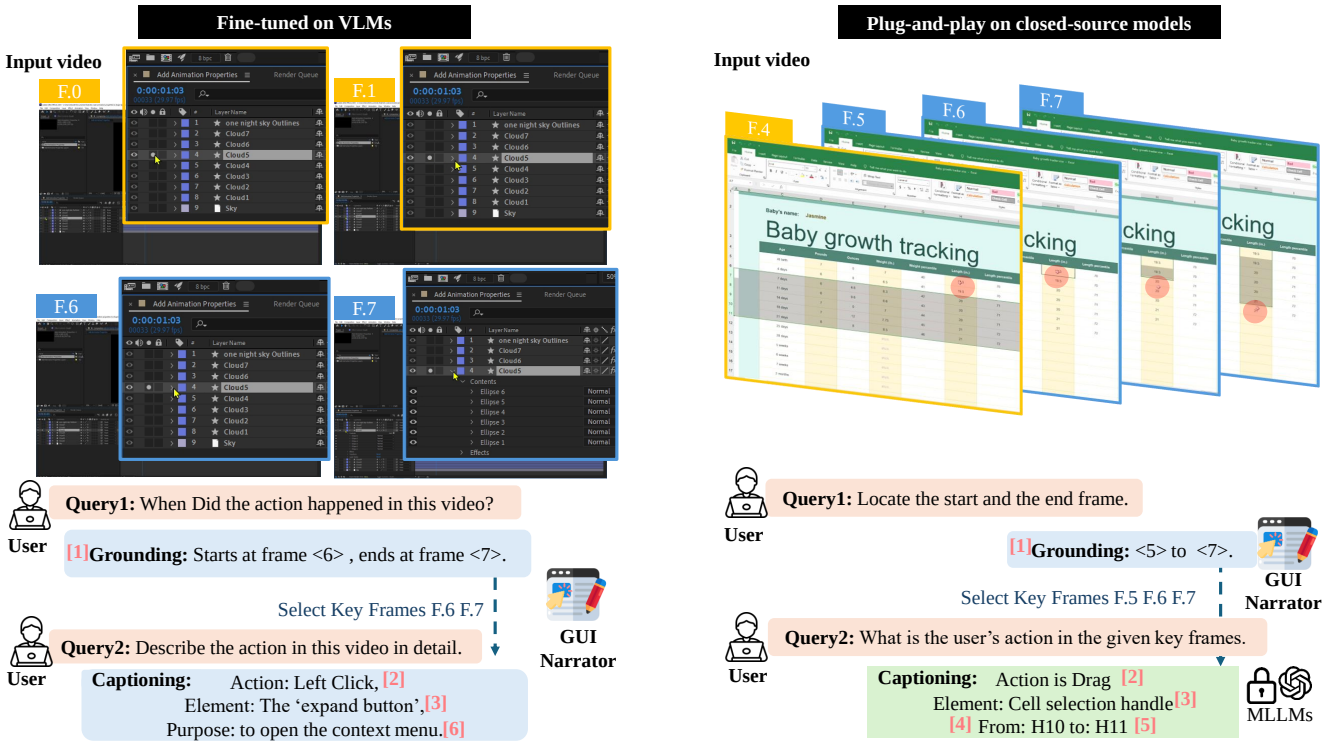


Figure 1: Overview of the framework GUI-Narrator . Left: A two-stage fine-tuned QA model for action frame detection and action captioning for GUI tasks. Right: A plug-and-play implementation on closed-source MLLMs. We numbered the key elements in the response from [1] to [6].

Abstract

Recent advancements in Vision Language Models (VLMs) have demonstrated remarkable capabilities in understanding static screenshots. However, a key aspect of building a robust GUI automation system is understanding dynamic GUI actions such as videos depicting fundamental GUI actions, which enables agents to learn from human demonstrations. This is a non-trivial task that is distinct from natural scene video captioning: (i) GUI screenshots contain more concentrated information than natural scenes due to their high-resolution environment. (ii) Events in GUI videos occurred more

†Corresponding author.

MM '25, Dublin, Ireland

ACM ISBN 979-8-4007-2035-2/2025/10
https://doi.org/10.1145/3746027.3755150

quickly, requiring attention on time-span detection. (iii) Frames in GUI videos with less information increase unnecessary computational costs for captioning. To address these challenges, we propose **Act2Cap** , a new video captioning benchmark specifically designed for GUI action videos, comprising 10,866 diverse video caption pairs containing not only temporal information of keyframes but also detailed narration on action types, elements, location, and purpose. In addition, we propose **GUI-Narrator** , a framework utilizing cursor detection to enhance action interpretation in high-resolution screenshots. Our framework demonstrates improved performance in both open-source models and as a plug-and-play solution for closed-source models while reducing computational costs. The datasets and models are available at https://github.com/showlab/GUI-Narrator.

CCS Concepts

• **Human-centered computing** → **Human computer interaction (HCI)**; • **Information systems** → **Information systems applications**.

Keywords

GUI Agents, Large Language Models, Multimodal, Video Understanding

ACM Reference Format:

Qinchen Wu, Difei Gao, Qinghong Lin, Zhuoyu Wu, and Mike Zheng Shou † . 2025. GUI-Narrator: Detecting and Captioning Computer GUI Actions. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746027.3755150

1 Introduction

GUI (Graphical User Interfaces) Automation holds significant importance as it streamlines and optimizes user interactions with GUIs, drastically improving the efficiency of digital tasks, including information seeking, online shopping, and software copilot. Existing research for this topic can be broadly categorized into two tasks: **UI-Element Grounding:** [3, 14, 17, 24, 25] propose new pre-training models to better understand and ground GUI elements [26, 33, 38, 50]. **Action Planning** [21, 22, 29, 40] focus on building AI agents for action generation. However, one crucial capability remains unaddressed for building a powerful GUI automation system: **Action Narration**, comprehending the specific GUI actions depicted in screenshot recordings. Existing studies rely on static screenshots and annotator-labeled actions, overlooking the potential trajectory through multiple frames or online tutorial videos. Extracting action traces from these data not only enables a more effective reproduction of user behavior but also provides insights into user engagement with applications. Such capabilities significantly extend the functionality of GUI automation systems, making them more versatile and intelligent in understanding real-world tasks.

Understanding actions in natural scenes (i.e., video captioning) [18, 28, 46] is a well-established task in multimedia and machine learning. However, the GUI domain introduces a distinct set of challenges that deviate significantly from those encountered in natural settings. First, GUI screenshots contain much denser information than natural scenes, with numerous interactive elements packed into small areas. Second, actions within GUIs are subtler and occur more rapidly, demanding higher temporal resolution to capture transient interactions and filter keyframes [10, 49]. Third, precise spatial localization is crucial [7, 23] as the exact positioning of the cursor and other elements can dictate the meaning and outcome of an action. Given these specialized demands, it becomes essential to develop a tailored benchmark for GUI video captioning.

To address the unique demands of GUI video captioning, we have meticulously crafted a GUI video caption benchmark, named Act2Cap , encompassing 10,866 video-captioning pairs. Each video in the dataset meticulously records a distinct GUI action, such as left-clicks, right-clicks, double-clicks, drags, and typing, spanning a variety of software environments. This diversity in software types includes Adobe Premiere Pro, Adobe After Effects, Adobe Illustrators, Office, and Web Tools, reflecting the broad applicability of GUI automation across different platforms. The models tasked with analyzing these videos must not only accurately identify the type of action performed but also the specific interface elements involved, such as buttons, menus, or text fields. **Our fine-grained action caption cover:** `[Keyframes, Action type, Element, Start, End Position, Purpose of the action]`. Our benchmark consists of 10,866 video-action pairs. We selected 820 pairs from human-labeled annotations with keyframes well distributed across all the frame indices for testing. The remaining are used for training.

To address the challenges of GUI video captioning, we developed a simple yet effective framework, GUI-Narrator , tailored for this domain. Our approach leverages the cursor's location as a focal point of interaction within GUIs. Specifically, we train a lightweight detection model to accurately locate various cursor formats in high-resolution screenshots. Using the detected cursor positions, the model identifies keyframes where interactions occur in the video. In the stage of action narration, the framework processes images between and including the detected keyframes represented by larger numbers of visual tokens to generate more informative action captions.

Our main contributions are as follows:

1. We curated a fine-grained action trajectory dataset comprising 10K video-caption pairs, spanning a diverse range of professional software layouts.

2. We conducted extensive evaluations of both open-source and closed-source models on our newly developed benchmark to assess their capabilities and limitations in GUI video captioning. Our findings reveal that the best-performing model, GPT-4o [27], achieved only 30.4% accuracy, highlighting the challenges posed by dense information within GUI interaction and the high inference cost of encoding high-resolution screenshots.

3. Our results demonstrate the effectiveness of our framework by improved performance on lightweight open-source models. Furthermore, integrating our strategies to closed-sourced models can improve performance while reducing inference tokens to $\times 0.38$. This suggests that our approach provides a scalable solution that enhances the adaptability and accuracy of models in handling complex GUI environments.

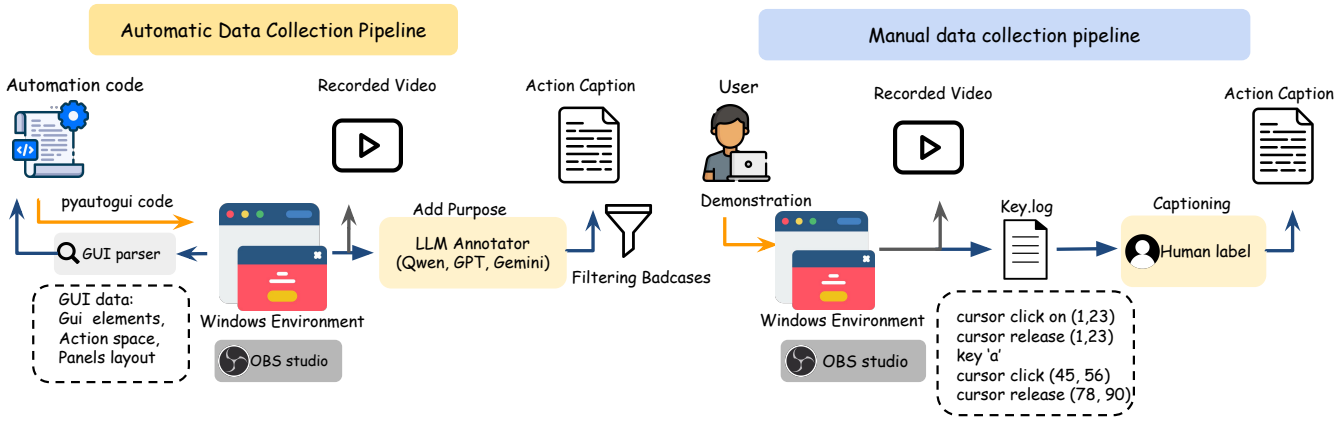


Figure 2: Two data collection pipelines. We use LLMs to generate the purpose for each action collected by automation scripts based on the recorded screenshots and the cropped region around the cursor. These synthetic data are employed for training. For the manually collected captions, they are double-checked and set aside for testing.

2 Related Work

GUI Benchmarks. In the field of GUI agents for task automation, current benchmarks can be broadly categorized into two distinct types: UI-grounded and UI automation benchmarks. UI-grounded benchmarks [8, 9, 35] assess the ability of agents to understand and interpret the graphical user interface based on visual cues and natural language commands, focusing on the agent's proficiency in mapping these inputs to specific UI elements and actions. On the other hand, UI automation benchmarks [11, 16, 48] are designed to evaluate the performance of these agents in executing complex sequences of actions [5] within an interface, thereby measuring their capability to automate tasks effectively across different applications and platforms. Differing from previous works [14, 45, 47], ours introduces the Act2Cap benchmark, which aims for models to understand GUI recordings. This approach serves as a complement to existing GUI agents, enabling models to grasp the meaning of user demonstrations. It enhances the model's comprehension of the functions of UI elements and, in the future, could assist models in learning automation knowledge directly from user demonstrations. We compare existing benchmarks and data with Act2Cap in Table1.

Video captioning Dataset. The domain of video captioning is supported by a diverse array of datasets that cater to different scenarios. Works like MSVD [42], MSR-VTT [43], and Charades-Ego [32]focus on activities in the real world. Others, like YouCook II [31, 52] provide data for instruction following cater to the culinary domain. Recently, GUI-WORLD [4] has created instruction-following datasets in digital worlds, spanning a wide range of actions in both web and app environments. However, current GUI-based video datasets lack detailed annotations of cursor movements, such as details action types(`double click`) and the action trajectory of `Drag` action, leading to less comprehensive action descriptions. Our proposed benchmark and captions aim to fill this gap and enhance the quality of action narration.

Multimodal Agents. Strong visual encoders and large language models (LLMs) [2] have enabled a wide range of visual-language tasks. These models have shown significant capabilities in areas such as grounding, planning, and execution [12]. There are also efforts being made on video understanding. Unified temporal tokens [15, 37] have been incorporated into the vocabulary to help models ground daily activities in real-world scenarios. In the domain of GUIs, vision-language tasks [8, 36] have focused on parsing GUI screenshots to achieve high granularity with low cost [19], while others employ multi-modal large language models (MLLMs) to assist in step-by-step planning [13, 44], enhancing precision and intelligence for automation. There are also efforts being made on video understanding. Later work [4] have leveraged fine-tuned video LLMs as GUI agents to improve the comprehension of GUI videos. Our approach, is distinct from others by introducing a framework that leverages cursor location and action timestamp to assist with high-resolution, text-heavy, and fast-action recordings.

3 Act2Cap Benchmark

3.1 Task Formulation

We present the pipeline of GUI-Narrator shown in Figure 1. The model generates natural language outputs by performing two key sub-tasks: **Temporal Grounding** to detect keyframes and **Action Narration** to describe the action and its purpose.

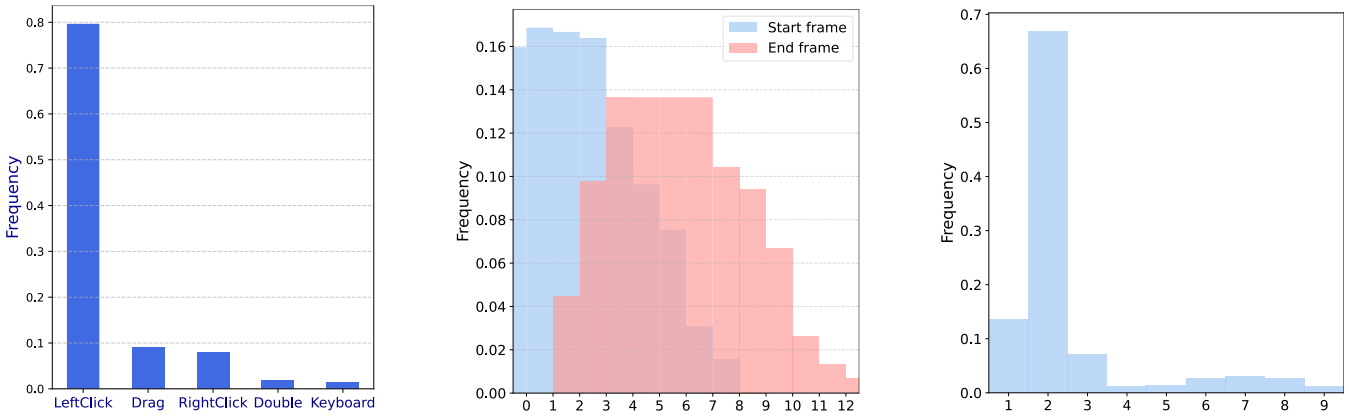
3.2 Data Collections

Our dataset consists of a wide range of GUI actions covering cursor actions (including Left-Click, Right-Click, Double-Click, and Drag) and keyboard Type actions. The data collection under the Windows GUI environment consists of two separate pipelines shown in Figure 2. In addition to our collected data, we included open-source data GUI-WORLD [4] with added annotations to tailor it for fine-grained narration tasks.

Video Collection. The videos in our benchmark consists of: Auto collected videos, manually recorded videos, and existing benchmarks. Initially, our auto-collected videos are generated via an **automatic data collection pipeline**. A GUI parser [11] first analyzes the GUI layout into three main components: 1) Interactive GUI elements; 2) Action spaces such as `Left Click, Right Click,`

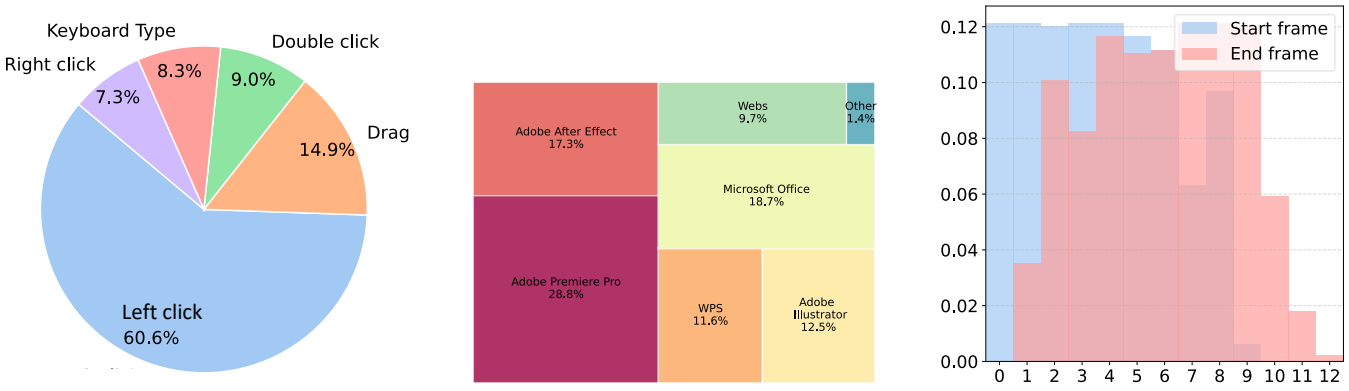
Table 1: Comparison of the existing GUI benchmarks on their Component and Tasks for models in detail.

Datasets	Size	Data components	UI Grounding	Action Planning	Action Narration	
					Temporal det.	Actions
ScreenSpot [8]	1.2k	Single Src.	✓	✗	-	-
AgentStudio [51]	227	Test Env.	✓	✓	-	-
OSworld [41]	369	Test Env.	✓	✓	-	-
MIND2WEB [9]	2.3k	Multi Src.	✓	✓	-	-
GUI-Course [5]	10.7M	Multi Src.	✓	✓	-	-
GUI-World [4]	12.3k	Video	✗	✓	✗	High-level
Act2Cap (Ours)	**10k**	**Video**	✗	✗	✓	**Fine-grained**



(a) Distribution of Action Space (b) Freq. of Starting and Ending Frames (c) Freq. of Frame Durations

Figure 3: Statistics of the collected data: We categorized actions into five types, covering nearly all interactions in PC environments. Since 'Left click' is the major action, we down-sampled its instances and up-sampled the others in training to prevent overfitting.



(a) Distribution of Action Space (b) Specific Software Composition (c) Freq. of Starting and Ending Frames

Figure 4: A carefully selected subset from the collected and processed data. Additionally, GUI layouts such as Adobe Illustrator, WPS, and Web-based interfaces are only available in the test set.

`Double Click`, `Drag`, and `Keyboard type`; 3) Coordination of specific panels like the `timeline`, `component` and `effect panels`, which are crucial for determining the starting or ending positions of the `Drag` action. For `Click` actions, the automatic data collection pipeline executes *pyautogui* code based on the selected `[element, action type]` pair and the parsed button locations. For Drag actions, the pipeline generates execution code tailored to `[element, action type, panels]` to direct the cursor in performing drag

operations. To emulate human interaction, the starting time of the cursor is randomized, thereby avoiding the introduction of rigid temporal knowledge regarding the start and end times of the atomic action video.

Although the automatic data collection pipeline efficiently scales up data collection, it struggles with executing more complex, human-like actions such as `'drag the cursor to draw a circle'`. To address these challenging tasks, we introduced a second pipeline known as the **manual data collection**. User-demonstrated videos are recorded and segmented into sub-videos based on the recorded key logs, with one action per video.

For data from existing benchmarks, similar to processing the auto-collected videos, we randomize the frames around the labeled keyframes and extract atomic actions from videos with multiple action steps.

Annotation Generation. (1). Most of the Click actions are generated in an automatic collection pipeline, and annotations are generated based on the predefined action space. For example, given a specific element and action, the annotation would be formatted as `[Specific Action, element]`. The `purposes` of actions are annotated using MLLMs (GPT4-o, Gemini). (2). To regularize narration and identify action space in sampled data from open-source data, we prompt GPT4-o and Gemini to categorize actions and extract elements in provided captions into formatted JSON structures. Later, we filtered out bad cases with meaningless actions. We will provide detailed pipelines in the supplementary materials.

Human Labeled Annotation. Human annotators are tasked by following online tutorials across various domains, including PR, AE, PPT, and WEB. The annotations are strictly defined template `[Specific Action, element, start position, end position, purpose]`. The videos are segmented into atomic actions based on the key logs, capturing both the timestamps of each action and the corresponding mouse click positions. Screenshots are extracted at these timestamps to document the entire trajectory of the action.

Benchmark Settings. We split our Act2Cap into **Training set:** 6.5k data sampled from our collected data and 4k from GUI-WORLD (add detailed annotations and frame sampling). Data statistics are shown in Figure 3. **Testing set:** 820 meticulously chosen pairs that feature balanced distributions of keyframes and a diverse range of software compositions from manual data. Figure 4 demonstrates the distribution of test set benchmark.

4 Task Metrics

Temporal Grounding. We evaluate keyframe detection performance using Recall@IOU and mIOU metrics. Recall@IOU measures the percentage of predictions with an IOU above a specified threshold, reducing sensitivity to minor deviations in predicted action boundaries and providing a more sensitive assessment.

Dense Narration Caption. Evaluating caption performance at the token level fails to capture the overall semantic meaning. To address this limitation, we utilize GPT-4o-08-06 as an evaluator to assess whether each element aligns with the semantic meaning on an element-wise basis, assigning a score of 0 for mismatches and 1 for matches. Because the elements contained in Click, Keyboard Type and Drag narrations are different, we categorize actions into three types for evaluation: Click actions (including Left-Click, Right-Click and Double-Click), Drag actions, and Keyboard Type actions, as shown in Table2 • **Click actions**: These are instantaneous actions evaluated based on the name of `element`, the type of cursor `click` and `purpose` for click. • **Drag actions**: These require consideration of 4 factors: `the start position`, `end position`, `GUI element` involved, and the intended `purpose` of the drag. • **Keyboard Type actions**: These are evaluated based on the `elements` (i.e., keys) typed or pressed and the `purpose`, which can include adding text, executing hotkeys, or using shortcuts.

For each evaluated data point, the evaluation generates a list of five elements for `Drag` actions and three elements for `Click` and `Keyboard Type` actions. To enhance the robustness of this evaluation strategy, we predefine knowledge for the LLM evaluator, treating 'buttons' and 'icons', as well as 'folders' and 'files', as having equivalent semantic meanings. The final evaluation result is the average score of each element in the output, normalized to a range of 0 to 1.

Table 2: Elements to be assessed for each action type in Act2Cap . Each element is evaluated as 0/1.

Action type	Specific action	Start	End	UI element	Purpose
Click	✓	-	-	✓	✓
Keyboard Type	✓	-	-	✓	✓
Drag	✓	✓	✓	✓	✓

5 Method

Overview. We propose a two-stage baseline model built upon Qwen2VL-2B for atomic step narration in GUI, as depicted in Figure5. We use LM loss [20] as the training objective.

In the first stage, the input video is uniformly sampled into 5 to 12 frames, followed by a dynamic cropping strategy that focuses on the region of interest around the cursor, leveraging a trained cursor detection model. This stage extracts keyframes representing the GUI screenshots before and after each atomic action. In the second stage, the frames between the extracted keyframes are used as visual input to generate an action narration including `[Action type]`, `[Element]`, `[Start and End position]`, and `[Purpose]`.

5.1 Spatial Detection

Given the video frame list of $[V_1, V_2, \ldots, V_N]$, where V_i denotes the i_{th} of sampled frame, the output will be a cropped frame sequence $[V_1^{cropped}, V_2^{cropped}, \ldots, V_N^{cropped}]$, N is the number sampled frame.

Cursor Location Detection. The current baseline approach, which uses the entire image as input, hinders vision models from effectively focusing on the action region. To address the difficulty, we trained a cursor detection model based on YOLO-v8 [34] under our collected dataset.

Cropping Location Creation. After detecting the frames, we apply a strategy to crop the frames centered around the detected cursor position $c = (x, y)$. The crop center is determined as the average of the x and y coordinates across N sampled frames. A minimum region of 512×512 pixels is cropped based on the determined center. If the cursor's movement exceeds the 512×512 pixel

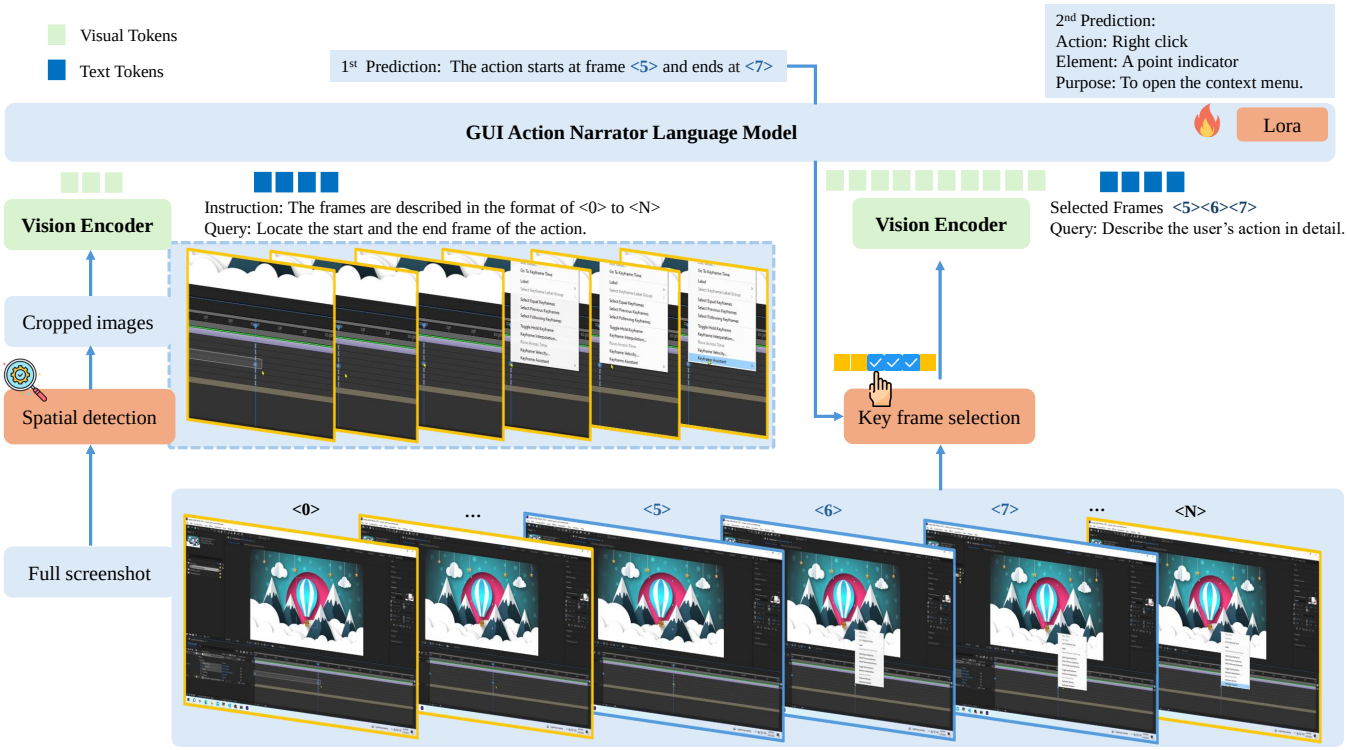


Figure 5: Two-stage prediction with GUI-Narrator : Initially, a spatial detection model crops the screenshots from the full-resolution images. The objective of the first stage is to identify keyframes within the cropped screenshot sequences. During the second prediction stage, a keyframe selection module is employed to choose frames from the original full-resolution screenshot sequences.

region, an additional buffer of $\delta = 128$ pixels is added to expand the cropping area, ensuring the cursor's actions are fully captured.

5.2 Keyframe Detection

Visual Inputs. The visual inputs of the VLM model are cropped frames $[V_1^{cropped}, V_2^{cropped}, ..., V_N^{cropped}]$ from the sampled screenshot sequence. We utilize the pre-trained vision encoder from Qwen2VL to encode the input image sequence. For training efficiency, we limit the visual token representing each image (e.g.160). As a result, the visual tokens input of the VLM can be calculated to at most $160 \times N$.

Instructional Text Input. Compared to traditional vision-only models, VLMs can process text inputs, enhancing Temporal Grounding. The model will be provided with an instruction that denotes the input frames sequentially from `<0>` to `<N>`, where N is the total frame count, along with a query like `"When did the action happen in this video? Tell me the start frame and the end frame."`. The model outputs text with `<s>` and `<e>` marking the start and end keyframes indices.

5.3 Action Captioning

Consequent to the stage of keyframe detection, we implement two alternative solutions shown in Figure 6.**i) Fixed-length:** Given the start index `s` and end index `e`, the selected frames are defined as $[V_{\texttt{<s>}}, V_{\texttt{<(s+e)//2>}}, V_{\texttt{<e>}}]$. **ii)Variable-length:** All frames between the indices `s` and `e` are selected to form the visual input sequence: $[V_{\texttt{<s>}}, V_{\texttt{<s+1>}}, ..., V_{\texttt{<e-1>}}, V_{\texttt{<e>}}]$. The text input is designed to guide the model to recognize the entire screenshot and regularize its output format. A template of the text query is: `"Narrate the action in detail. Follow the template: Action, Element, Source, Destination, Purpose."`

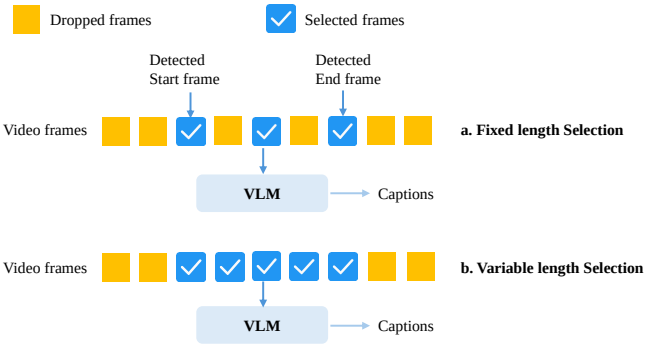


Figure 6: Two keyframe selection strategies.

Table 3: Average grounding score(GS)and caption score(CS) of five types of actions.* denotes we implement the checkpoint of ShowUI without path merging. Column FT shows the training data and strategy. TG denotes whether the temporal grounding is applied in the captioning. C and T&C denote captioning with all frames and captioning after temporal detection, respectively. GS is calculated by Recall@IOU= 0.5 and CS is derived from the metric mentioned in Section 4.

Base Model	FT?	TG?	Left Click		Right Click		Double Click		Drag		Keyboard		Avg.	
			GS	CS	GS	CS	GS	CS	GS	CS	GS	CS	GS	CS
GPT-4o-mini-07-18	-	-	-	21.8	-	16.9	-	10.5	-	25.6	-	21.5	-	20.9
Gemini-1.5-flash [30]	-	-	-	24.1	-	27.2	-	5.4	-	26.8	-	23.5	-	23.0
Claude-3.5-sonnet [1]	-	-	-	28.7	-	**28.1**	-	9.0	-	28.9	-	**29.8**	-	27.0
GPT-4o-08-06 [27]	-	-	-	**34.4**	-	27.8	-	**10.8**	-	**29.4**	-	26.9	-	**30.4**
InternVL2.5-2B [6]	**C**	-	-	22.9	-	15.4	-	3.6	-	11.5	-	5.3	-	17.4
+ GUI Narrator	**T&C**	✓	24.5	25.1	23.3	11.7	39.1	17.5	31.9	12.9	23.5	3.9	26.7	19.8
Qwen2VL-2B [39]	**C**	-	-	24.4	-	23.8	-	5.3	-	11.8	-	1.4	-	18.8
+ GUI Narrator	**T&C**	✓	24.6	**34.3**	23.0	22.7	35.1	11.7	40.1	**15.4**	26.4	4.8	**28.0**	**26.1**
ShowUI-2B* [19]	**C**	-	-	23.7	-	17.8	-	6.3	-	12.3	-	4.9	-	18.4
+ GUI Narrator	**T&C**	✓	24.1	32.4	23.7	**28.8**	39.1	**17.7**	41.2	15.1	19.4	**5.8**	27.9	26.0

Table 4: Comparison of average caption score(%) between zero-shot performance and plug-and-play strategy. GUI Narrators are used as temporal grounding for closed-source MLLMs. * The tokens of GPT-4o-mini are calculated based on cost during inference.

Base Model	Tokens per action	Reduction	Left Click	Right Click	Double Click	Drag	Keyboard	Avg.
GPT-4o-08-06	11,746	1	**34.4**	27.8	10.8	29.4	26.9	**30.4**
+ GUI Narrator	4,390	×0.37	34.1	27.2	**12.7**	28.0	26.0	30.1
GPT-4o-mini-07-18*	134,959	1	21.8	16.9	10.5	25.6	21.5	20.9
+ GUI Narrator	48,943	×0.36	22.5	17.7	7.2	28.4	23.0	21.7
Claude-3.5-sonnet	13,250	1	28.7	**28.1**	9.0	28.9	**29.8**	27.0
+ GUI Narrator	5,660	×0.43	29.2	27.2	8.1	**29.5**	28.4	27.2

6 Experiments

Implemenation Details. We utilize 4 NVIDIA A5000 GPUs to train our models. At first, we train a cursor detection model using a cursor dataset for grounding. Secondly, we fine-tune the VLM model on the Temporal Grounding and Action Captioning dataset by applying LoRA to LLM, with a LoRA rank of 8 and a LoRA alpha of 32. **Dataset Sampling.** To ensure a balanced distribution of keyframes (start, end frames) and action types, we adopt a resampling strategy during dataset creation. Specifically, we down-sample actions associated with `Left Click` and up-sample actions involving `Drag` and `Keyboard`.

6.1 Quantitative Results

6.1.1 Comparison with State-of-the-arts. In Table 3, we compare the captioning performance of closed and fine-tuned models across five action types. Models without temporal grounding for keyframes detections generate captions using all screenshot frames. Although GPT-4o achieves the highest average score, its performance in `Double click` is comparatively lower than its other action types. We compare two fine-tuning data strategies: **(1)** Captioning with all frames (**C**), using a maximum of 160 visual tokens per frame. **(2)** Captioning with keyframes grounded temporally (**T&C**), using up to 640 visual tokens per frame for fewer than three keyframes, and 240 tokens per frame for more. The key difference between these strategies lies in whether temporal grounding is utilized to narrate actions. Our findings indicate that temporal grounding leads to improvements in average scores for all three open-source models, showing increases of 13.7%, 41.3%, and 38.8%, respectively. Among the five action types, improvements in `Double Click`, `Left Click`, and `Drag` are particularly notable because these actions benefit from training in temporal grounding. However, `Keyboard` actions remain more challenging, as they are less dependent on the cursor's position when typing on the keyboard. Furthermore, we compare the model with close source models that support video inputs. The results reveal that our fine-tuned models surpass these models in `Right click` and `Double click` actions. `Keyboard` actions are still challenging for small-sized models but demonstrate improvement after detecting keyframes.

6.1.2 Plug and play performance. Table 4 shows the caption score of close-sourced models incorporating with GUI-Narrator , which preliminarily detects the keyframes and subsequently employs these selected frames to generate captions with a reduced number

of visual input tokens. Our results demonstrate that this plug-and-play strategy improves caption quality in models such as Claude and GPT-4omini while requiring about ×0.38 of the visual tokens compared to feeding all the frames to generate captions. Although the overall performance in GPT-4o falls slightly short, this strategy enables the model to accurately distinguish `Double click` actions.

6.2 Ablation Studies

In this section, we focus on the following key questions: **(1)** How does the cropping of screenshots impact temporal grounding performance? **(2)** Can different types of text queries enhance keyframe detection? **(3)** Does utilizing detected keyframes as sequential visual inputs or increasing the tokens per image by using a fixed number of frames improve the narration score? To address these questions, we conducted experiments to assess the significance of dynamic cropping, the role of guidance queries, and effective strategies for selecting frames after identifying the key frames.

Impact of Dynamic cropping. In Table 5, we present experiments on the temporal grounding of keyframes with and without dynamic cropping. Grounding using cropped screenshots consistently outperforms grounding with full screenshots, showing an improvement of 5.3%. A 1920×1080 resolution screenshot generates approximately 2644 after applying 14 ×14 patching and merging by 2× 2. This approach is effective because it preserves more information from a 512×512 pixel screenshot by compressing 334 tokens ($\frac{512\times512}{28\times28}$) to 160 tokens, compared to compressing 2644 tokens from a 1920×1080 screenshot to just 160 tokens.

Table 5: Ablation study on the temporal grounding for keyframe detection.

Base Model	Cropped?	R@0.4	R@0.5	R@0.6	mIOU
InternVL2.5	✗	28.9	25.3	8.4	23.0
	✓	29.3	26.7	9.0	23.9
Qwen2VL	✗	30.2	27.0	9.0	23.3
	✓	32.3	28.0	9.5	24.1

Guidance in Temporal Grounding. In Table 6, we implement three different types of guidance following the query template. **(i).** Both `[Action]` and `[Element]` are used to ground the keyframes in the videos. **(ii).** Either `[Action]` or `[Element]` are provided. **(iii).** No guidance is provided, as mentioned in Section 5.2. Our experimental results demonstrate that the variable `[Action]` has a greater impact on keyframe detection than `[Element]`. When evaluated using the benchmark R@0.6, leveraging both `[Action]` and `[Element]` as textual grounding guidance yields an improvement of approximately 80%. However, in our benchmark scenario, we assume no prior knowledge of the specific action type or element. Consequently, the framework operates under guidance without specific `[Action]` or `[Element]` when generating action narrations.

Impact of keyframe selection. In Table 7, we compare two different types for frame selection shown in Figure 6. Our experimental results show that using a fixed length for selection improves performance in `Left Click` actions. However, it leads to a decrease in performance for `Right Click` and `Drag` actions, which can be attributed to the reduction in the number of frames, challenging the model to differentiate between actions with longer durations and those with shorter durations. As a result, using Variable-length selection reaches the state-of-the-art performance.

Table 6: Random: Denotes the random guess of start and end indices. S&E: Denotes the prediction is always the first and last sampled frame. ZS denotes the zero-shot performance of temporal grounding on a closed-source model.

Base Model	Element	Action	R@0.5	R@0.6
S&E	-	-	11.1	6.9
Random	-	-	18.9	9.3
Claude-3.5-ZS	✓	✓	23.7	13.1
	✓	✗	18.7	12.5
	✗	✓	20.5	11.3
	✗	✗	19.1	11.2
InternVL2.5	✓	✓	30.1	16.7
	✓	✗	27.4	8.6
	✗	✓	29.6	15.8
	✗	✗	26.7	9.0
Qwen2VL	✓	✓	33.2	18.4
	✓	✗	29.3	11.2
	✗	✓	32.2	17.5
	✗	✗	28.0	9.5

Table 7: Ablation study on the frame selection types. F.: Denotes the fix-length, each frame containing up to 760 tokens(or 256×3 for InternVL). V.: Variable-length. The maximum number of tokens is determined by the total number of detected input frames. For sequences with more than 3 frames, the maximum tokens per frame are limited to 240 (or 256 for InternVL).

Base Model	Select Type	LC	RC	DC	DG	KT	Avg.
InternVL2.5	F.	27.2	6.0	8.1	12.2	2.4	19.6
	V.	25.1	11.7	17.5	12.9	3.9	**19.8**
Qwen2VL	F.	35.0	21.3	10.8	9.2	5.3	25.5
	V.	34.3	22.7	11.7	15.4	4.8	**26.1**

7 Conclusion

In this paper, we propose a GUI Video Caption benchmark, featuring 10,866 samples, that addresses unique challenges specific to GUIs, such as denser information and rapid, subtle events. We also propose GUI-Narrator to utilize a cursor as a visual prompt to enhance the interpretation of screenshots spatially and temporally. Despite the task's complexity, challenging even for advanced models like GPT-4o and Claude-3.5-sonnet, GUI-Narrator effectively boosts their performance while reducing the cost of input tokens to ×0.38 through detecting keyframes. This framework also demonstrates an improvement of 41.3% (ShowUI), and 38.8% (Qwen2VL) in open-source vision-language models, highlighting its adaptability and impact in GUI automation.

8 Acknowledgment

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-RP-2022-030).

References

[1] Anthropic. 2024. Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet. Accessed: 2025-02-02.

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).

[3] Rajat Chawla, Adarsh Jha, Muskaan Kumar, Mukunda NS, and Ishaan Bhola. 2024. GUIDE: Graphical User Interface Data for Execution. *arXiv preprint arXiv:2404.16048* (2024).

[4] Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, et al. 2024. GUI-WORLD: A Dataset for GUI-oriented Multimodal LLM-based Agents. *arXiv preprint arXiv:2406.10819* (2024).

[5] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. 2024. GUICourse: From General Vision Language Models to Versatile GUI Agents. *arXiv preprint arXiv:2406.11317* (2024).

[6] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. 2024. Expanding Performance Boundaries of Open-Source Multimodal Models with Model, Data, and Test-Time Scaling. *arXiv preprint arXiv:2412.05271* (2024).

[7] Zhaorun Chen, Zhuokai Zhao, Hongyin Luo, Huaxiu Yao, Bo Li, and Jiawei Zhou. 2024. HALC: Object Hallucination Reduction via Adaptive Focal-Contrast Decoding. arXiv:2403.00425 [cs.CV] https://arxiv.org/abs/2403.00425

[8] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935* (2024).

[9] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a Generalist Agent for the Web, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 28091–28114. https://proceedings.neurips.cc/paper_files/paper/2023/file/5950bf290a1570ea401bf98882128160-Paper-Datasets_and_Benchmarks.pdf

[10] Nikita Dvornik, Isma Hadji, Ran Zhang, Konstantinos G. Derpanis, Animesh Garg, Richard P. Wildes, and Allan D. Jepson. 2023. StepFormer: Self-supervised Step Discovery and Localization in Instructional Videos. arXiv:2304.13265 [cs.CV] https://arxiv.org/abs/2304.13265

[11] Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. 2024. AssistGUI: Task-Oriented PC Graphical User Interface Automation. (2024), 13289–13298.

[12] Difei Gao, Lei Ji, Luowei Zhou, Kevin Qinghong Lin, Joya Chen, Zihan Fan, and Mike Zheng Shou. 2023. Assistgpt: A general multi-modal assistant that can plan, execute, inspect, and learn. *arXiv preprint arXiv:2306.08640* (2023).

[13] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the Digital World as Humans Do: Universal Visual Grounding for GUI Agents. *arXiv preprint arXiv:2410.05243* (2024). https://arxiv.org/abs/2410.05243

[14] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2024. Cogagent: A visual language model for gui agents. (2024), 14281–14290.

[15] De-An Huang, Shijia Liao, Subhashree Radhakrishnan, Hongxu Yin, Pavlo Molchanov, Zhiding Yu, and Jan Kautz. 2025. Lita: Language instructed temporal-localization assistant. In *European Conference on Computer Vision*. Springer, 202–218.

[16] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024. Language models can solve computer tasks. *Advances in Neural Information Processing Systems* 36 (2024).

[17] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. VisualWebArena: EVALUATING MULTIMODAL. *arXiv preprint arXiv:2401.13649* (2024).

[18] Kevin Lin, Linjie Li, Chung-Ching Lin, Faisal Ahmed, Zhe Gan, Zicheng Liu, Yumao Lu, and Lijuan Wang. 2022. Swinbert: End-to-end transformers with sparse attention for video captioning. (2022), 17949–17958.

[19] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. ShowUI: One Vision-Language-Action Model for GUI Visual Agent. *arXiv preprint arXiv:2411.17465* (2024).

[20] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in neural information processing systems* 36 (2024).

[21] Zhe Liu, Chunyang Chen, Junjie Wang, Xing Che, Yuekai Huang, Jun Hu, and Qing Wang. 2023. Fill in the blank: Context-aware automated text input generation for mobile gui testing. (2023), 1355–1367.

[22] Zhe Liu, Chunyang Chen, Junjie Wang, Mengzhuo Chen, Boyu Wu, Xing Che, Dandan Wang, and Qing Wang. 2023. Chatting with gpt-3 for zero-shot human-like mobile automated gui testing. *arXiv preprint arXiv:2305.09434* (2023).

[23] Zuyan Liu, Yuhao Dong, Yongming Rao, Jie Zhou, and Jiwen Lu. 2024. Chain-of-Spot: Interactive Reasoning Improves Large Vision-Language Models. arXiv:2403.12966 [cs.CV] https://arxiv.org/abs/2403.12966

[24] Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. Comprehensive CognitiAGENTS ON REALISTIC VISUAL WEB TASKSve LLM Agent for Smartphone GUI Automation. *arXiv preprint arXiv:2402.11941* (2024).

[25] Michel Nass, Emil Alegroth, and Robert Feldt. 2023. Improving web element localization by using a large language model. *arXiv preprint arXiv:2310.02046* (2023).

[26] Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. 2024. ScreenAgent: A Vision Language Model-driven Computer Control Agent. *arXiv preprint arXiv:2402.07945* (2024).

[27] OpenAI. 2024. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/ Accessed: 2024-05-21.

[28] Boxiao Pan, Haoye Cai, De-An Huang, Kuan-Hui Lee, Adrien Gaidon, Ehsan Adeli, and Juan Carlos Niebles. 2020. Spatio-temporal graph for video captioning with knowledge distillation. (2020), 10870–10879.

[29] Lihang Pan, Bowen Wang, Chun Yu, Yuxuan Chen, Xiangyu Zhang, and Yuanchun Shi. 2023. AutoTask: Executing Arbitrary Voice Commands by Exploring and Learning from Mobile GUI. *arXiv preprint arXiv:2312.16062* (2023).

[30] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530* (2024).

[31] Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *Pattern Recognition: 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings 36*. Springer, 184–195.

[32] Gunnar A Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. 2018. Charades-ego: A large-scale dataset of paired third and first person videos. *arXiv preprint arXiv:1804.09626* (2018).

[33] Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022. META-GUI: towards multi-modal conversational agents on mobile GUI. *arXiv preprint arXiv:2205.11029* (2022).

[34] Ultralytics. 2024. YOLOv8 Documentation. https://docs.ultralytics.com/models/yolov8/. Accessed: 2025-02-12.

[35] Sagar Gubbi Venkatesh, Partha Talukdar, and Srini Narayanan. 2023. UGIF: UI Grounded Instruction Following. (2023). arXiv:2211.07615 [cs.CL]

[36] Jianqiang Wan, Sibo Song, Wenwen Yu, Yuliang Liu, Wenqing Cheng, Fei Huang, Xiang Bai, Cong Yao, and Zhibo Yang. 2024. OmniParser: A Unified Framework for Text Spotting, Key Information Extraction and Table Recognition. arXiv:2403.19128 [cs.CV] https://arxiv.org/abs/2403.19128

[37] Haibo Wang, Zhiyang Xu, Yu Cheng, Shizhe Diao, Yufan Zhou, Yixin Cao, Qifan Wang, Weifeng Ge, and Lifu Huang. 2024. Grounded-videollm: Sharpening fine-grained temporal grounding in video large language models. *arXiv preprint arXiv:2410.03290* (2024).

[38] Jiayin Wang, Weizhi Ma, Peijie Sun, Min Zhang, and Jian-Yun Nie. 2024. Understanding User Experience in Large Language Model Interactions. *arXiv preprint arXiv:2401.08329* (2024).

[39] 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. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv preprint arXiv:2409.12191* (2024).

[40] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. AutoDroid: LLM-powered Task Automation in Android. (2024).

[41] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. *arXiv preprint arXiv:2404.07972* (2024).

[42] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. 2017. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the 25th ACM international conference on Multimedia*. 1645–1653.

[43] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5288–5296.

[44] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2024. Aguvis: Unified Pure Vision Agents for Autonomous GUI Interaction. (2024). https://arxiv.org/abs/2412.04454

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

[46] Antoine Yang, Arsha Nagrani, Paul Hongsuck Seo, Antoine Miech, Jordi Pont-Tuset, Ivan Laptev, Josef Sivic, and Cordelia Schmid. 2023. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. (2023), 10714–10726.

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

[48] Zhuosheng Zhan and Aston Zhang. 2023. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436* (2023).

[49] Chenlin Zhang, Jianxin Wu, and Yin Li. 2022. ActionFormer: Localizing Moments of Actions with Transformers. arXiv:2202.07925 [cs.CV] https://arxiv.org/abs/2202.07925

[50] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024. Android in the Zoo: Chain-of-Action-Thought for GUI Agents. *arXiv preprint arXiv:2403.02713* (2024).

[51] Longtao Zheng, Zhiyuan Huang, Zhenghai Xue, Xinrun Wang, Bo An, and Shuicheng Yan. 2024. AgentStudio: A Toolkit for Building General Virtual Agents. *arXiv preprint arXiv:2403.17918* (2024).

[52] Luowei Zhou, Chenliang Xu, and Jason Corso. 2018. Towards automatic learning of procedures from web instructional videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.