GUI-Xplore: Empowering Generalizable GUI Agents with One Exploration

Supplementary Material

A. Overview

Our supplementary includes the following sections:

- **Section B: Data Collection Details.** Details for annotations generation.
- **Section C: Dataset Statistical Details.** Details for dataset samples and statistics.
- **Section D: Experimental Details.** Details for experimental settings.
- **Section E: Additional Experiment.** Details for additional evaluation results.
- **Section F: Additional Discussion.** Further discussion for boarder impacts and limitations.

B. Data Collection Details

The raw exploratory information was collected using annotation tools, including screen recordings of the exploration process, screenshots of the visited pages with view hierarchy (VH), operational data logs, and a GUI transition graph generated through rule-based clustering. Based on this raw data, question-answer pairs for downstream tasks were subsequently created.

Raw Data Processing. To filter invalid data generated during the automated exploration process, we employed a page-count-based analysis method. Invalid data typically arose from network connectivity issues, login failures, or app-related errors. To ensure temporal alignment, timestamps were cross-referenced with video frame data using pixel-based calibration. Erroneous data identified in this process were discarded. Additionally, operational logs from the Android system were mapped to specific action categories, including click, input, swipe, back, and exit.

Data Enrichment. To provide comprehensive annotations of app functionality, supplementary data were collected from Google Play store descriptions, encompassing textual summaries and user feedback about the app.

Data Generation. Using the collected data, GPT was utilized to generate two types of outputs:

- Global Functional Summaries: Information scraped from Google Play was provided to GPT as input, with carefully designed prompts guiding the model to summarize the overall functionality of the app.
- Page-Level Functional Descriptions: For any given page, the shortest navigation path from the homepage to the target page (derived from the GUI transition graph) was recursively processed. Page descriptions and operational details along the path were interleaved to form prompts, prompting GPT to generate functional descriptions for the target page and associated virtual tasks.

Question-Answer Pair Generation. Five distinct types of question-answer pairs were systematically generated based on the processed data.

- Application Overview: The global functional description serves as the answer.
- Page Analysis: The specific timestamp serves as the question, while the corresponding page's functional description serves as the answer.
- Application Usage: Virtual tasks derived from the page's operational data serves as the question, while the shortest path from the homepage to the page (as extracted from the GUI transition graph) was used as the answer.
- Action Recall: The page's functional description serves as the question, while the corresponding timestamp was used as the answer. To ensure uniqueness, only leaf nodes from the GUI transition graph were used in this task.
- Action Sequence Verification: Triplets of pages were extracted from the GUI transition graph, and their topological order serves as the answer. Unique reachability sets were generated for all nodes in the graph during data collection, ensuring exhaustive coverage. Operation triples with well-defined topological relationships were carefully annotated to enhance the comprehensiveness of this task.

Distractor Generation. GPT was further employed to generate distractor options for each question. These distractors were designed to closely mimic the format and length of the correct answers, while intentionally containing incorrect or misleading content.

C. Dataset Statistical Details

To ensure the proposed method supports various input modalities, we curated a dataset with diverse modalities, including exploration videos, images, view hierarchies (VH), and operational information. Specific data sample are illustrated in the Figure 1.

The GUI-Xplore dataset includes 312 apps spanning six major categories (Entertainment, Productivity, Health, Shopping, Travel, and News) and 33 subcategories. Figure 2 illustrates the distribution of these categories, emphasizing the dataset's comprehensive coverage across various application types.

Furthermore, GUI-Xplore supports five downstream tasks, including Application Overview, Page Analysis, Application Usage, Action Recall and Action Sequence Verification. A set of question-answering examples for these tasks is provided to illustrate its utility in facilitating different research objectives.

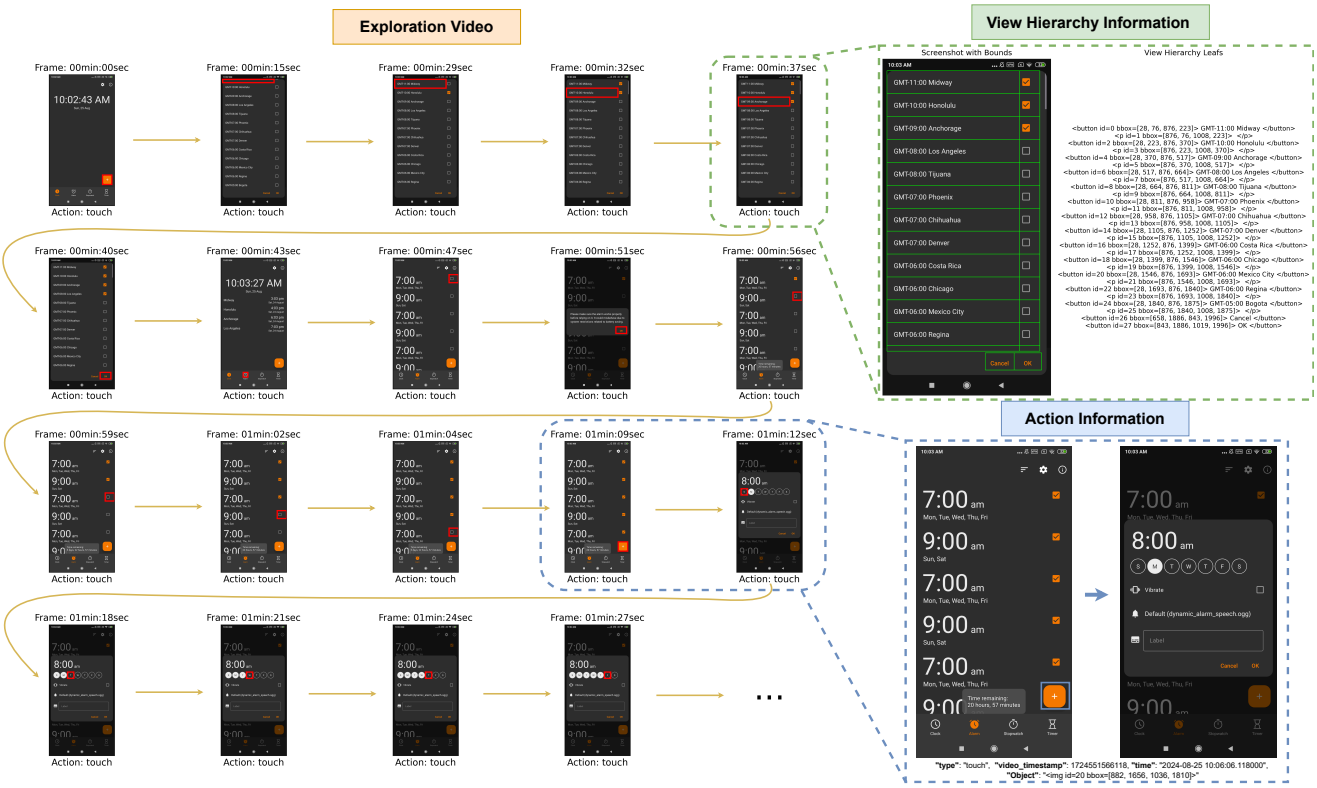

Figure 1. **GUI-Xplore Data Sample.** The samples include exploration video, screenshot, view hierarchy and action information.

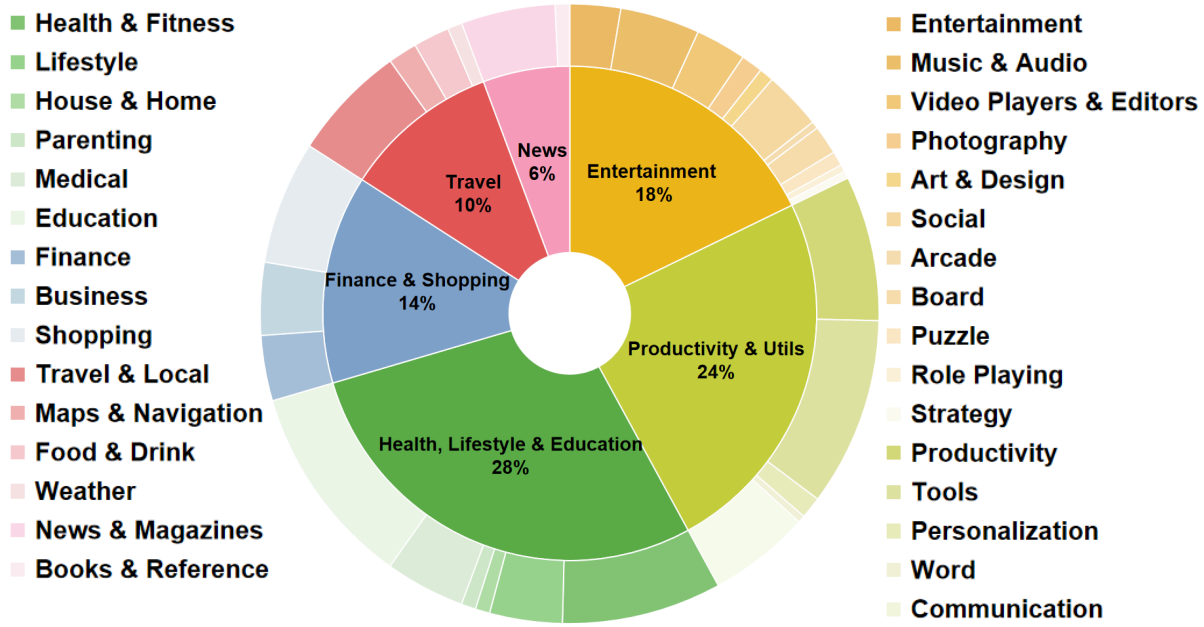

Figure 2. **Application Category Distribution.** The dataset encompasses 312 apps spanning six major categories and 33 subcategories

1. Application Overview:

```
"Question": "Classify the app and select the functions that best describe its capabilities.",
"Option": {
    "A": "Real-time traffic updates; AI-generated painting tutorials; Virtual wardrobe organization",
    "B": "Timer setup with customizable ringtones and vibration options; Fully open source with customizable colors; Choice of week starting on Sunday or Monday",
    "C": "Virtual reality dating experiences; Augmented reality game integration; Interactive live streaming fitness sessions",
    "D": "Live streaming meditation sessions; Live art classes with renowned artists",
    "E": "AI-powered gardening tips; Customized daily affirmations"
},
"Answer": "B"
```

2. Page Analysis:

```
"Question": "At the 00min:07sec second timestamp, what does the app screen facilitate?",
"Option": {
    "A": "Selecting a time zone from a list of available options for display settings",
    "B": "Users can access a virtual assistant for additional help and guidance.",
    "C": "Users can participate in virtual escape rooms and puzzle challenges on this screen.",
    "D": "This UI allows users to track their caffeine intake and consumption patterns.",
    "E": "Users can track their daily caloric intake and nutritional balance on this screen."
},
"Answer": "A"
```

3. Application Usage:

```
"Question": "Describe the path to the page for "Selecting a time zone from a list of available options for display settings" from the home screen of the app.",
"Option": {
    "A": "Start Simple Clock; Touch About button; Touch Third party licences button;",
    "B": "Touch Settings button on the clock display with timezones screen;",
    "C": "Start Simple Clock; Touch the clock display; Touch the alarm icon; Touch OK on the alarm warning; Touch the alarm settings; Touch the sound selection.",
    "D": "Touch on Time Zone settings button from the clock display screen;",
    "E": "Start Simple Clock; Touch About button; Touch Frequently asked questions button;"
},
"Answer": "D"
```

4. Action Recall:

```
"Question": "When did the user navigate to the [Selecting a time zone from a list of available options for display settings] interface in the video?",
"Option": {
    "A": "06min:53sec",
    "B": "01min:21sec",
    "C": "10min:29sec",
    "D": "00min:07sec",
    "E": "03min:45sec"
},
"Answer": "D"
```

5. Action Sequence Verification:

```
"Question": "Is the order of page visits in the video aligned with the required flow for [support_information_overview_screen:Providing information on support, FAQs, contact details, contributors, donation options, and social media links.; support_overview_screen: Providing information about the app, including support options, FAQs, contributors, donation options, and social media links.; support_faq_prompt_screen:Providing guidance for users before asking support questions, directing them to check settings and FAQs, and ensuring they are using the latest app version.] tasks?",
"Option": {
    "A": [
        "support_overview_screen",
        "support_faq_prompt_screen",
        "support_information_overview_screen"
    ],
    "B": [
        "support_overview_screen",
        "support_information_overview_screen",
        "support_faq_prompt_screen"
    ],
    "C": [
        "support_faq_prompt_screen",
        "support_information_overview_screen",
        "support_overview_screen"
    ],
    "D": [
        "support_information_overview_screen",
        "support_faq_prompt_screen",
        "support_overview_screen"
    ],
    "E": [
        "support_faq_prompt_screen",
        "support_overview_screen",
        "support_information_overview_screen"
    ]
},
"Answer": "B"
```

D. Experimental Details

For the GUI environment understanding in Xplore-Agent, we utilized QwenVL-7B as the base model. Training data was derived from the page visual hierarchy (VH) and operational information in the training set of the GUI-Xplore

View Hierarchy	Action	Overview	Page	Usage	Recall	SeqVeri	Avg.
-	-	96.88%	82.12%	66.48%	22.60%	28.85%	59.39%
Generated	GT	99.25%	97.25%	68.35%	24.20%	39.72%	65.75%
GT	Generated	99.62%	96.34%	67.79%	23.70%	35.88%	63.97%
GT	GT	99.63%	92.86%	67.40%	24.54%	36.71%	64.23%
Generated	Generated	99.25%	92.86%	68.21%	24.36%	36.54%	64.24%

Table 1. Comparison of different configurations across view hierarchy and action settings.

Method	Exploration Percentage	Overview	Usage	Avg.
GPT-4V	100%	96.88%	66.48%	81.68%
Xplore-Agent	20%	98.38%	66.10%	82.24%
Xplore-Agent	50%	97.88%	65.17%	81.53%
Xplore-Agent	80%	99.38%	67.04%	83.21%
Xplore-Agent	100%	99.25%	68.21%	83.73%

Table 2. Probing the impact of exploration coverage on task performance across Application Overview and Application Usage tasks.

dataset. We trained the model on NVIDIA A100 GPUs. Our hyperparameters are as follows: learning rate of 3e-5, batch size of 4, 2500 steps for VH generation and 3500 steps for action generation.

For the subsequent graph-guided reasoning module, we employed GPT for GUI clustering and reasoning tasks. The specific version used was *'gpt-4o-mini-2024-07-18'*, which facilitated accurate understanding and task execution.

E. Additional Evaluation Results

E.1. GUI Modeling Ability

Two-stage video understanding methods are inherently limited by the performance of their first-stage visual understanding module. To explore this dependency, we conducted additional experiments evaluating how VH generation and action generation accuracy affect the final reasoning performance of Xplore-Agent. Specifically, we compared model performance under two scenarios: excluding the GUI transition graph and replacing the generated VH and action information with their ground truth counterparts.

The experimental results in table 1 demonstrate that leveraging the GUI transition graph to structure information from exploration videos significantly enhances the reasoning capabilities across diverse downstream tasks. However, the accuracy of the GUI modeling module introduces certain constraints on overall system performance. Interestingly, replacing only the action information with ground truth improved task accuracy in specific scenarios. In contrast, replacing the VH information with ground truth unexpectedly led to performance degradation on certain tasks. This phenomenon may be attributed to the excessive text volume in the ground truth VH, which could introduce confusion during the reasoning phase.

E.2. Exploration Coverage

The exploratory videos in the GUI-Xplore dataset aim to achieve comprehensive coverage of software pages. However, in practical apps, full page coverage is often unattainable. To examine the impact of varying exploration coverage rates on Xplore-Agent's reasoning accuracy, we conducted additional experiments focusing exclusively on tasks unrelated to video completeness, specifically the Application Overview and Application Usage tasks.

The results indicate that incomplete environmental exploration significantly affects Xplore-Agent's reasoning accuracy. In extreme cases, the performance of Xplore-Agent aligns closely with GPT4V without the GUI Transition Graph. Interestingly, the model performance under 50% exploration coverage was lower than both 20% and 80% coverage. We hypothesize this is due to the insufficient information provided at 50% coverage combined with an increased text volume, which may exacerbate confusion in the reasoning process.

F. Additional Discussion

GUI-Xplore is designed to support the evaluation and testing of generalized GUI agents across diverse apps and tasks. Achieving strong generalization in GUI agents, by enabling them to adapt to diverse app environments and user needs, has the potential to significantly enhance user experiences on mobile devices and revolutionize existing human-computer interaction paradigms. However, current

limitations in result generation, deployment efficiency, and privacy preservation pose challenges, which we aim to address in future work.

F.1. Gap Between Q&A and Action Prediction

To ensure consistency across multiple downstream tasks, we unified task evaluation using a question-answering (Q&A) framework. While this approach offers a direct benchmarking of existing GUI agents' performance across tasks and app platforms, it introduces a gap between QA-based evaluation and real-world operational deployment. Ideally, a robust GUI agent should exhibit not only strong QA and task planning capabilities but also precise execution of actions. Our experiments have demonstrated that the exploration-based Xplore-Agent achieves higher zero-shot action generation accuracy compared to state-of-the-art methods. Future work will focus on extending support for action-oriented tasks, advancing the development of GUI agents with enhanced generalization and stability.

F.2. Trade-off Between Accuracy and Efficiency

While our experiments highlight the superior performance of the exploration-based framework in cross-app and cross-task scenarios, the inherent trade-off between accuracy and efficiency arises due to the additional deployment cost of preliminary exploration. We anticipate that this challenge will be mitigated through two main avenues. First, the automation capabilities of GUI agents themselves enable efficient exploration of application environments. Recent advancements in automated and comprehensive application exploration techniques support this trajectory. Second, mobile devices, as the primary medium of human-computer interaction, generate vast amounts of unlabeled user interaction data daily. Given that the exploration-based framework does not require additional annotation during inference, it naturally leverages locally generated user operation records as prior environmental knowledge. This approach not only facilitates environmental understanding but also models user behavior patterns, paving the way for personalized virtual assistants.