GUIOdyssey: A Comprehensive Dataset for Cross-App GUI Navigation on Mobile Devices

Supplementary Material

7. Ethical Discussion

Privacy. We use temporary accounts and virtual usernames to register various apps and ensure no personal information is entered. The dataset contains no authentic personal information.

Ethical Consent in Data Collection. A formal consent process is implemented, wherein participants explicitly agree to the inclusion of their human-annotated data in the dataset. All data are collected with informed consent and in full compliance with ethical guidelines.

Security Concerns. The development of intelligent agents trained on datasets like this offers significant potential for automating tasks and enhancing accessibility. However, it also raises important ethical and security concerns. Sensitive operations, such as financial transactions or privacy management, pose vulnerabilities without robust safeguards. Additionally, malicious actors could exploit these agents to bypass security protocols or manipulate applications for unethical purposes. To mitigate these risks, it is crucial to implement secure model designs, privacy-preserving techniques, and establish clear ethical guidelines. Addressing these challenges will help ensure the responsible deployment of such technology while maximizing its societal benefits.

8. Details of GUIOdyssey

8.1. Description of Task Categories

The specific details of the six task categories are as follows:

General Tool. This category encompasses tasks that involve navigating through system-wide operations such as managing system settings or notifications for apps. An instruction example of a general tool task is "Adjust the notification settings for the YouTube app on your phone using Settings, then proceed to open YouTube".

Information Management. Information management tasks involve searching for information and recording it for future use. This might include looking up information on search engines, reading articles on news apps, checking facts on educational or reference apps, and then saving or organizing this information in note-taking apps.

Web Shopping. Shopping tasks encompass a range of activities related to purchasing products online. Users may start by searching for a product on one app, comparing prices on different e-commerce platforms, checking reviews and ratings on review apps or websites, and finally making a purchase.

Media Entertainment. Media entertainment tasks are about activities involving video and music streaming apps. Users may browse for new content on video platforms like YouTube or Netflix, stream music on services like Spotify or Apple Music, and switch between different media apps to manage playlists or download content.

Social Sharing. This task involves activities where users share content across different social media platforms. This could include taking photos or videos with the camera app, editing them using a photo or video editing app, and then sharing them on multiple social media platforms like Instagram, Facebook, Twitter, or TikTok.

Multi-Apps. Multiple-app tasks involve more complex operations that require three or more apps to complete. For example, cooking food with an online recipe might involve finding the recipe of the food, recording the recipe to a note-taking app, and buying the ingredients online(Fig. 1).

8.2. Action Set

Our recording system utilizes Android Studio to simulate GUI navigation and virtualize various devices. We use the Android Debug Bridge (ADB) to retrieve device information and status, such as the coordinates of click events, and to monitor a wide range of functional keys. The details of the action set in our Android emulator are presented in Table 5.

8.3. Fine-grained Episode Annotation Generation

Fine-grained episode annotations consist of two components: low-level instructions and semantic annotations. Examples of the fine-grained annotations can be found in Fig. 7.

Low-Level Instruction. For each step within an episode, we provide GPT-4o with the high-level instruction corresponding to the episode, along with the action and screenshot associated with the current step. Additionally, for actions such as `CLICK` and `LONG PRESS`, we supply an additional image featuring a bounding box to indicate the click coordinates. All images are configured with the fidelity parameter set to 'high'. The prompt utilized is provided in Fig. 11.

Semantic Annotation. We use GPT-4o to generate semantic annotations in an alternating and iterative manner, following the sequential order of steps within each episode. Specifically, the process begins by providing the current episode's high-level instruction along with the actions and decision rationale from previous steps, prompting GPT-4o

Table 5. The argument and functionality of different actions in GUIOdyssey. 'pos1' and 'pos2' denote the position (x, y).

Action	Argument	Functionality
`CLICK`	[pos1]	click the on-screen position
`LONG PRESS`	[pos1]	press the screen for a long time to copy texts or download images
`SCROLL`	[pos1, pos2]	scroll the screen from position 1 to position 2
`TYPE`	text	type text with keyboard
`COMPLETE`	-	the sign that the instruction has been completed
`IMPOSSIBLE`	-	the sign that the instruction cannot be completed
`HOME`	-	go to the home screen
`BACK`	-	go to the previous screen
`RECENT`	-	go to the previous App

to generate the contextual information for the current step. Subsequently, using the generated contextual information, the high-level instruction, the screenshot image, and the action corresponding to the current step, GPT-4o is prompted step-by-step to generate the screen description and decision rationale for the current step. This iterative process continues until all semantic annotations for each step within the episode are completed in sequence. Similarly, for actions such as `CLICK` and `LONG PRESS`, we supply an additional image with a bounding box indicating the click coordinates. All images are configured with the fidelity parameter set to 'high' to ensure precision. The prompts used for generating these annotations are provided in Fig. 12 and Fig. 13.

8.4. Examples

An example of episodes in our GUIOdyssey is shown in Fig. 6, while examples of semantic annotations can be found in Fig. 7. An example of an annotation for a task that could not be successfully completed and ends with the `IMPOSSIBLE` action can be found in Fig. 8 and Fig. 9.

As mentioned in Sec. 5.1, we use SAM2 [37] to assist in evaluating whether the model's output actions are correct. Fig. 10 provides examples of bounding boxes for clicked elements obtained through SAM2 segmentation.

8.5. Data Format

Each field of annotation is as follows.

episode_id: the unique identifier of this episode.

device_info: the detailed information of the virtual device from which the episode was collected, including the device model, screen resolution, and other device-related details.

task_info: the detailed information of the task from which the episode was collected, including the task category, the app used, the high-level instruction, and other task-related details.

step_length: the total number of steps in this episode.

steps: a list of steps in this episode. Each step in the list includes the file path of the screenshot, executed action and its corresponding parameters (*e.g.*, the coordinates for a click action), the low-level instruction, the semantic annotation, the bounding box obtained from SAM2 segmentation, and additional recorded information such as the overall scroll trajectory for scroll actions and annotator notes.

9. Experiment Details

9.1. Detailed description of four different setups.

The following details the four different setups in GUIOdyssey.

i) Train-Random & Test-Random. We randomly partitioned all the episodes in the dataset into training and testing sets using a ratio of 80% to 20% as the standard approach to divide the dataset. It can assess the in-domain performance of OdysseyAgent.

ii) Train-Task & Test-Task. In this setup, We proportionally sampled meta-tasks from six categories, maintaining approximately a $6:1$ ratio for the training and test sets. The tasks in the test set differ significantly from those in the training set. This partitioning method allows for a robust assessment of an agent's generalization capabilities across diverse tasks.

iii) Train-Device & Test-Device. To evaluate an agent's generalizability across different and unseen devices, we selected episodes annotated on the Tablet, which differs significantly from other devices, as the test set. We obtained $1,381$ episodes as the test set and $6,953$ episodes as the training set.

iv) Train-App & Test-App. This split is aimed at evaluating the agent's performance on unseen Apps and App combinations. First, we calculated the frequency of app usage in the dataset and categorized the apps into 25 classes

Table 6. The impact of different semantic annotations on OdysseyAgent across four different splits. We use high-level instructions for both training and evaluation. Performance is assessed using AMS and SR as metrics. SD, CI, and DR denote screen description, contextual information, and decision rationale, respectively.

	Semantic Annotation			**Test-Random**		**Test-Task**		**Test-Device**		**Test-App**		**Overall**	
	SD	CI	DR	AMS	SR	AMS	SR	AMS	SR	AMS	SR	AMS	SR
(1)	✗	✗	✗	75.79	9.38	54.36	0.09	61.20	1.88	63.03	7.70	63.60	4.76
(2)	✓	✗	✗	75.18	8.94	54.06	0.00	64.41	2.03	64.91	8.47	64.64	4.86
(3)	✗	✓	✗	75.42	10.04	55.71	0.00	62.52	3.19	64.24	5.30	64.47	4.63
(4)	✗	✗	✓	77.71	11.44	55.60	**0.26**	65.88	4.63	65.74	7.96	66.23	6.07
(5)	✗	✓	✓	77.23	11.16	56.93	0.18	63.87	2.24	66.32	7.87	66.09	5.36
(6)	✓	✗	✓	77.24	10.88	**57.15**	0.00	63.55	2.17	**67.04**	**9.67**	66.24	5.68
(7)	✓	✓	✗	76.58	10.14	57.13	**0.26**	64.48	3.91	66.27	7.96	66.11	5.57
(8)	✓	✓	✓	**78.24**	**11.62**	56.19	**0.26**	**66.63**	**5.07**	65.89	8.81	**66.74**	**6.44**

(*e.g.*, Video, Music) based on their characteristics. Then, we selected a few apps with the lowest occurrence from each class to form the test app set. Subsequently, we partitioned the episodes that utilized the app in the test app set into the Test-App set, maintaining an approximately 85% to 15% ratio between the training set and the test set.

9.2. Training Details.

To train OdysseyAgent, we employ the AdamW optimizer with a learning rate of $2e{-}5$ and utilize a cosine learning rate schedule. We set β_1 and β_2 to 0.9 and 0.95, respectively, and use a weight decay of 0.1. Additionally, we utilize a global batch size of 128 and implement DeepSpeed ZERO2-style data parallelism. During training, OdysseyAgent treats each action step as an individual training sample. The input consists of the task instruction, the current screenshot, and the previous 4 actions and screenshots (*i.e.*, $\delta = 4$), while the output corresponds to the action for the current step. By default, OdysseyAgent is trained separately on Train-Random/Task/Device/App for one epoch, excluding the semantic annotation component. When training includes semantic annotations, these annotations are converted into single-turn QA pairs, which serve as additional training samples (*i.e.*, semantic annotations are introduced only during training-time). Any training configuration that incorporates semantic annotations is explicitly noted. The entire training process requires approximately 32 A100 hours to complete.

9.3. Prompt for Evaluation.

We utilize the prompt shown in Fig. 14 to evaluate the performance of GPT-4V, GPT-4o, Claude3.5-sonnet, and InternVL2-Pro. For SphAgent and CogAgent, we tested them following their officially recommended methods [9, 23].

10. More Experiments

10.1. History Resampler vs. Multi-Image Training.

We evaluate different approaches for processing historical screenshot images. Qwen-VL supports multi-image input by interleaving image and text tokens, but this leads to a high token overhead (*e.g.*, 1024 tokens for four historical steps). Our history resampler compresses this to 256 tokens, greatly improving efficiency. As shown in Table 7, both approaches achieve comparable performance, but the history resampler significantly enhances training and inference efficiency.

Table 7. The average AMS for HL and LL instructions across 4 splits, along with the number of historical screenshot tokens, inference metrics (Time to First Token (TTFT) and Tokens per Second (TPS)), and training GPU hours.

strategy	HL	LL	Token Count	TTFT ↓	TPS ↑	GPU Hours
history resampler	63.60	82.44	256	0.71	20.27	32
multi-image	65.04	82.34	1024	0.98	17.05	48

10.2. The effect of different semantic annotations.

We assess the impact of different semantic annotations in GUIOdyssey (*i.e.*, screen description, contextual information and decision rationale) on model performance in both in-domain and out-of-domain settings. The results are presented in Table 6. A comparison of experiments (1)–(4) shows that all three components contribute positively, but engaging in detailed reasoning before making decisions is more important than understanding current screen information or summarizing historical processes in cross-app tasks. Experiments (5)–(8) further indicate that using two or more types of semantic annotations generally outperforms using a single annotation type. Specifically, using all semantic annotations yields the best results and improves AMS by 3.14

and SR by 35% compared to training without any semantic annotations. These findings suggest that teaching the model to understand the reasoning behind each action—similar to how humans observe, understand, review completed steps, and reason thoroughly before deciding—can be beneficial for improving performance in both in-domain and out-of-domain cross-app tasks.

10.3. Transferability of instructions at different levels of granularity.

As shown in Table 8, models trained on high-level instructions exhibit significantly better transferability across different levels of instruction granularity compared to those trained on low-level instructions. Furthermore, training on both instruction granularities outperforms training on a single granularity, a phenomenon similar to what has been observed in single-app tasks [26].

10.4. Transferability across different devices.

We utilize our GUIOdyssey dataset to conduct additional experiments to evaluate the generalization capabilities of OdysseyAgent beyond the initial experimental setup. we test the OdysseyAgent's adaptability by using data from one device as the test set while training on data from the remaining five devices. The results of these experiments are presented in the Table 9, demonstrating the model's performance across different devices. The model exhibits the weakest transferability on tablet devices, which we attribute to the significant differences in user interface layouts between tablets and smartphones. Furthermore, the model's transferability on small phones and foldable devices is also suboptimal. We surmise that the disparity in screen resolution compared to other phone models may contribute to this underperformance.

Table 8. The results for OdysseyAgent trained and tested on Train-Random/Test-Random with both high-level and low-level instructions are presented, with AMS as the evaluation metric. HL and LL denote high-level and low-level instructions, respectively.

Testing Instructions	Training Instructions		
	HL	LL	HL + LL
HL	75.79	29.39	78.96
LL	71.26	86.88	88.84

10.5. Whether cross-App tasks benefit single-App tasks.

We further investigate whether cross-app tasks benefit single-app performance by evaluating the impact of different training data compositions under controlled conditions. Specifically, we randomly sample 50k training samples each from GUIOdyssey, AITW, and AndroidControl

Table 9. Performance Evaluation of OdysseyAgent Across Different Devices. Each Device serves as a test set while the remaining five devices are used as training sets.

Evaluation Device	**Resolution**	**AMS**	**SR**
Pixel 7 Pro	$1,440 \times 3,120$	75.91	7.44
Pixel 8 Pro	$1,344 \times 2,992$	74.67	6.05
Small Phone	$720 \times 1,280$	71.68	3.77
Medium Phone	$1,080 \times 2,400$	73.05	5.45
Pixel Fold	$2,208 \times 1,840$	67.67	4.48
Pixel Tablet	$2,560 \times 1,600$	61.20	1.88

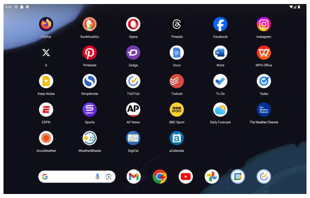
(denoted as Ody50k, AITW50k, and AC50k, respectively) and evaluate their performance on AndroidControl, which provides both in-domain and out-of-domain scenarios. As shown in Table 10, we find that incorporating cross-app data from GUIOdyssey consistently enhances performance in most single-app scenarios, whereas adding AITW data surprisingly yields limited improvements or even performance degradation. This suggests that the more complex cross-app tasks in GUIOdyssey can benefit single-app tasks.

Table 10. Effectiveness of Different Training Data on the AndroidControl. The evaluation metrics are the action matching score (AMS).

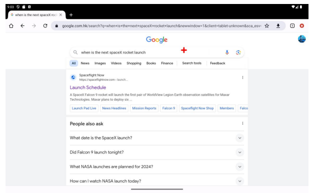
Training Data	IDD	category_unseen	app_unseen	task_unseen	Overall
AC50k	60.43	54.46	50.00	72.10	59.25
AC50k + AITW50k	60.69	**55.26**	45.19	68.84	57.50
AC50k + Ody50k	**61.48**	54.61	**50.96**	**72.46**	**59.88**

instruction: Utilize Chrome to research the key property of a Triangle and compile your findings into a concise document using Google Docs.

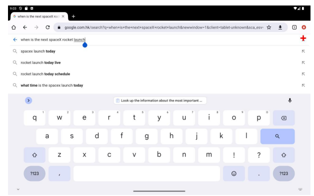
device_name: Pixel Tablet
category: Information_Management
app: ["Chrome", "Google Docs"]
step_length: 16



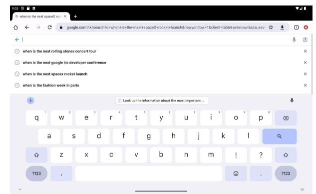
"Open the Chrome browser."
CLICK: [525, 912]



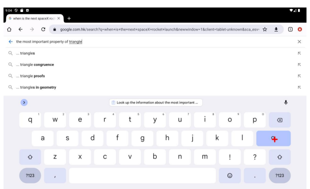
"Tap the search bar."
CLICK: [589, 244]



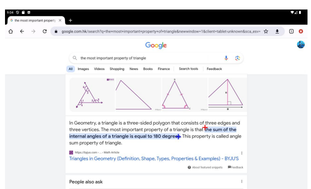
"Clear the current search query."
CLICK: [976, 172]



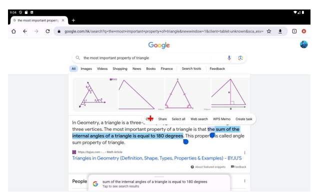
"Type 'the most important property of triangle' in the search bar."
TEXT: the most important property of triangle



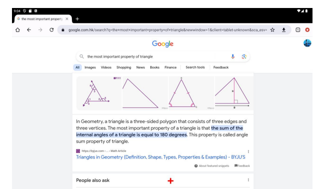
"Initiate the search query."
CLICK: [895, 696]



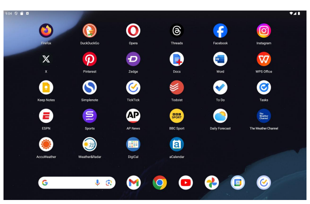
"Scroll to copy the related results."
SCROLL: [661, 622], [573, 671]



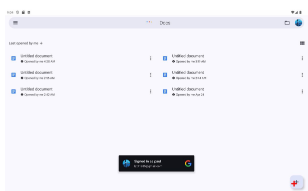
"Copy the highlighted text."
CLICK: [471, 576], [471, 576]



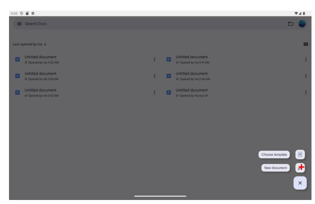
"Return to the home screen."
CLICK: HOME



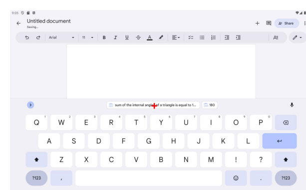
"Open the Google Docs app."
CLICK: [580, 272]



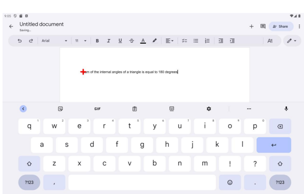
"Create a new document in Google Docs."
CLICK: [954, 919]



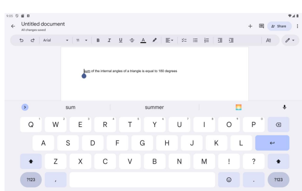
"Create a new document."
CLICK: [967, 826]



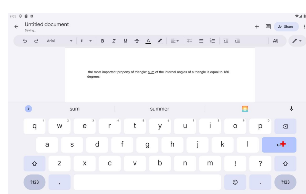
"Select the copied text from the keyboard."
CLICK: [477, 506]



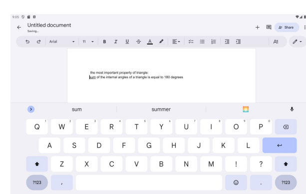
"Select the text at the beginning of the document."
CLICK: [265, 308]



"Type 'the most important property of a triangle:' into the document."
TEXT: the most important property of triangle:



"Press the Enter key."
CLICK: [904, 686]

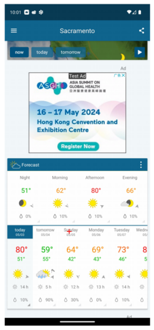


"Task completed."
COMPLETE

Figure 6. An example of episodes in our GUIOdyssey.

High-Level Instruction: Use Chrome to search for a new hiking trail, then check the weekend weather forecast using Weather & Radar. Finally, invite katsunaksu to join the adventure via Tumblr.

step: 13



CLICK: [444, 707]

low_level_instruction: Select the weather forecast for Sunday.

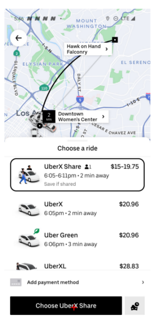
description: This is a screenshot of a weather forecast application displaying the weather for Sacramento. It shows detailed weather information for different parts of the day and upcoming days.

intention: I am selecting the weather forecast for Sunday to verify the conditions for the planned hiking trip. This action allows me to ensure the weather will be suitable for outdoor activities on that day, aiding in the planning process.

Context: So far, a search for new hiking trails has been conducted using the Chrome browser, and detailed information about popular trails in California has been gathered. The Weather & Radar app was then used to check the weekend weather forecast for Sacramento, CA, providing necessary weather insights for planning the hiking adventure.

High-Level Instruction: Using Waze Navigation & Live Traffic, locate a nearby zoo, then request a ride using the Uber app.

step: 10



CLICK: [476, 927]

low_level_instruction: Select the 'Choose UberX Share' button.

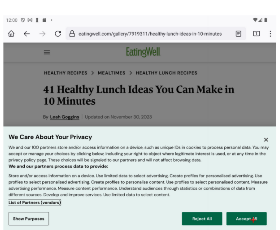
description: This is a screenshot of the Uber app showing ride options to a destination labeled 'Hawk on Hand Falconry.' The user is presented with various ride choices including UberX Share, UberX, Uber Green, and UberXL, along with their estimated arrival times and prices.

intention: I am selecting the 'Choose UberX Share' option to confirm my ride to 'Hawk on Hand Falconry.' This action is appropriate because it finalizes my transportation arrangement, ensuring a cost-effective and timely arrival at my destination.

Context: So far, the task has involved using the Waze app to locate a nearby zoo by searching for it and confirming the search results. After identifying the desired zoo location, the Uber app was accessed to request a ride, specifying 'Hawk On Hand Falconry' as the destination to ensure a seamless transportation arrangement.

High-Level Instruction: Choose a nutritious lunch plan for tomorrow, then jot it down in Google Keep. After that, use Firefox to find and watch a Tiktok video on how to prepare one of the dishes.

step: 8



CLICK: [916, 901]

low_level_instruction: Accept the privacy terms to continue browsing the webpage.

description: This is a screenshot of a webpage from EatingWell displaying an article titled '41 Healthy Lunch Ideas You Can Make in 10 Minutes.' A privacy notice is visible, offering options to accept or reject data processing terms.

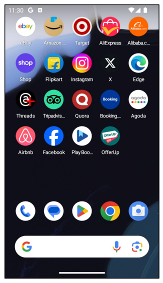
intention: I am choosing to click 'Accept All' to quickly dismiss the privacy notice and proceed with viewing the article. This action allows me to focus on selecting a nutritious lunch plan without interruptions, which is essential for completing my task efficiently.

Context: So far, the task has involved using the Firefox browser to search for a TikTok video on how to prepare a nutritious meal as part of a healthy lunch plan. The search included typing a query, initiating the search, and scrolling through the results to find relevant content. A link titled '41 Healthy Lunch Ideas You Can Make in 10 Minutes' was clicked to explore specific meal options that are both nutritious and quick to prepare.

Figure 7. Examples of fine-grained annotations in GUIOdyssey.

High-Level Instruction: Utilizing Threads and Booking.com, devise a travel itinerary for a trip to New York City, USA, and secure your lodging arrangements.

step: 1



CLICK: [162, 353]

low_level_instruction: Open the Threads app.

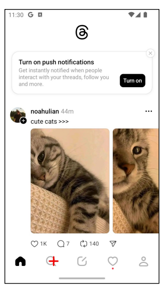
description: This is a screenshot of an Android home screen displaying various app icons, including Threads, Booking.com, and Airbnb.

intention: I am selecting the Threads app to gather ideas and recommendations for my travel itinerary to New York City. This step is crucial for accessing social media content that might offer insights or suggestions for the trip.

Context: Task just started, nothing has been done.

High-Level Instruction: Utilizing Threads and Booking.com, devise a travel itinerary for a trip to New York City, USA, and secure your lodging arrangements.

step: 2



CLICK: [162, 353]

low_level_instruction: Open the search section.

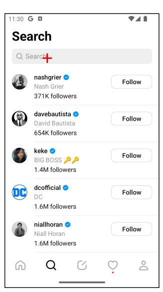
description: This is a screenshot of a social media app showing a post featuring pictures of a cute cat. There is also a prompt to turn on push notifications at the top of the screen.

intention: I am choosing to click on the search icon to explore more content related to travel ideas and recommendations for New York City. This action will help me gather information and inspiration for planning my travel itinerary.

Context: So far, the user has navigated to the main screen of their phone and accessed the Threads app.

High-Level Instruction: Utilizing Threads and Booking.com, devise a travel itinerary for a trip to New York City, USA, and secure your lodging arrangements.

step: 3



CLICK: [274, 163]

low_level_instruction: Tap on the search bar.

description: This is a screenshot of the search page in the Threads app, displaying a list of verified profiles with follower counts and follow buttons.

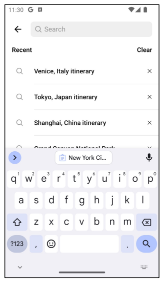
intention: I am selecting the search bar to input keywords related to New York City travel ideas. This action will help me find relevant content and recommendations to enhance my travel itinerary.

Context: So far, the user has navigated to the home screen of their phone and accessed the Threads app. They have also utilized the search function within the app to explore content related to travel ideas and inspiration for their trip.

Figure 8. Example of an annotation for an unsuccessful task, ending with the IMPOSSIBLE action.

High-Level Instruction: Utilizing Threads and Booking.com, devise a travel itinerary for a trip to New York City, USA, and secure your lodging arrangements.

step: 4



TYPE: New York City, USA itinerary

low_level_instruction: Enter 'New York City, USA itinerary' in the search bar.

description: This is a screenshot of a search interface on a mobile app showing recent searches for travel itineraries such as Venice, Italy and Tokyo, Japan. The user is currently typing 'New York City, USA itinerary' in the search bar.

intention: I am entering 'New York City, USA itinerary' in the search bar to gather ideas and recommendations for planning my trip. This action is appropriate as it allows me to find relevant information and suggestions to enhance my travel itinerary for New York City.

Context: So far, the user has navigated to the home screen of their mobile phone to efficiently manage travel planning tasks. They have accessed the Threads app and selected the search bar.

High-Level Instruction: Utilizing Threads and Booking.com, devise a travel itinerary for a trip to New York City, USA, and secure your lodging arrangements.

step: 5

CLICK: [162, 353]

low_level_instruction: Initiate the search query.

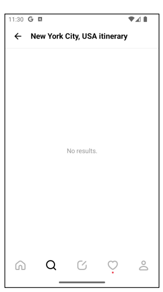
description: This is a screenshot of a mobile search interface where the user has entered 'New York City, USA itinerary' into the search bar. The keyboard is visible, and there is an option to search for the entered query.

intention: I am choosing to tap the search button to initiate a search for 'New York City, USA itinerary' within the Threads app. This action will help me find relevant content and recommendations to enhance my travel planning for the city. By executing this search, I expect to gather useful ideas and suggestions for my itinerary.

Context: So far, the user has navigated to the home screen of their mobile phone and accessed the Threads app to gather ideas for a travel itinerary to New York City. They have utilized the search feature within the app to input relevant keywords, aiming to find content and recommendations to enhance their travel planning for the city.

High-Level Instruction: Utilizing Threads and Booking.com, devise a travel itinerary for a trip to New York City, USA, and secure your lodging arrangements.

step: 6



IMPOSSIBLE

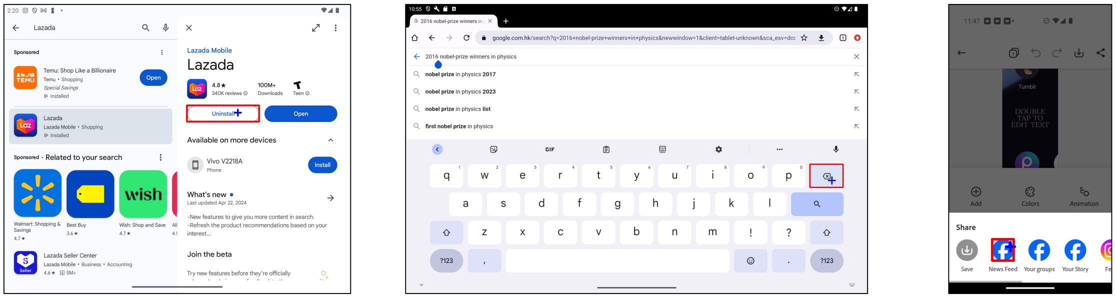
low_level_instruction: Task could not be completed.

description: This is a screenshot of a search result page in the Threads app displaying the message 'No results' for the query 'New York City, USA itinerary'. The screen shows that no content was found matching the search criteria.

intention: The task could not be completed because the search for 'New York City, USA itinerary' in the Threads app yielded no results. This lack of content prevented the gathering of ideas and recommendations necessary for planning the travel itinerary.

Context: So far, the user has navigated to the Threads app on their mobile phone to gather travel ideas and recommendations for a trip to New York City. They have utilized the search function within the app to find relevant content by entering 'New York City, USA itinerary' as the search term.

Figure 9. Example of an annotation for an unsuccessful task, ending with the `IMPOSSIBLE` action.



CLICK: [675, 374]   CLICK: [923, 609]   CLICK: [379, 838]

SAM2_BBOX: [528, 349, 736, 405]   SAM2_BBOX: [875, 553, 947, 633]   SAM2_BBOX: [266, 812, 398, 886]

Figure 10. Examples of bounding boxes for UI elements segmented by SAM2. The actual click locations are indicated by blue '+' symbols, while the red rectangles outline the bounding boxes obtained from the SAM2.

<img>current_screenshot.png</img>
<img>current_screenshot_w_labels.png</img>
Based on the original and marked screenshots of an Android mobile phone, where the marked screenshot is the original screenshot marked with action location, please follow the instructions below:

Low-Level Instruction Identification
- Identify the low-level instruction that the current action represents, such as "Go to the alarm section."

P.S.:
- When following these instructions, use natural language, avoid mentioning technical details (such as action coordinates) or direct use of action tags (such as "PRESS_HOME").

Output Format:
The output should be in JSON format as follows:
{{"instruction": "low-level instruction that the current action represents in the desired format"}}

Please return the result in pure JSON format, without any json tags like ```json ```.

Figure 11. Prompts for generating low-level instruction.

You are completing the task: {task} on a mobile phone, and the actions you have performed along with their respective intentions are listed in chronological order as:
{intentions}

Please follow the instructions below:
- Summarize the completed progress of the task based on the actions performed and their intentions in 2-3 sentences.
- Ensure your summary is purely focused on completed actions, and avoid provide further insight.
- Create a logically connected summary, rather than simply listing each action in detail.
- Avoid using any time-sequence phrases, such as 'after completing,' 'upon finishing,' or similar expressions.
- Use a completed-action tone, describing the progress as if each step has already occurred.
- Use an objective tone and describe concisely from an impersonal perspective.
- Format the context as follows: "So far, [summary of what has been accomplished]."

P.S.:
- When following these instructions, use natural language, avoid mentioning technical details (such as action coordinates) or direct use of action tags (such as "PRESS_HOME").

Output Format:
The output should be in JSON format as follows:
{{
"context": "a 2-3 sentence summary of task progress up to this point in the desired format"
}}
Please return the result in pure JSON format, without any json tags like ```json ```.

Figure 12. Prompts for generating contextual information.

```
<img>current_screenshot.png</img>
<img>current_screenshot_w_labels.png</img>
Based on the original and marked screenshots of an Android mobile phone, where the marked screenshot is the original screenshot marked with action location, please follow the instructions below:

1. Screenshot Description:
- Analyze and describe the overall content of the current screenshot.
- Provide a concise 2-3 sentence summary of the screenshot content.
- Format the description as follows: "This is a screenshot of [summary of the screenshot content]."

2. Intention Recognition:
- You are viewing the current screenshot while completing the task: {task} on a mobile phone, and you have chosen to perform the action: {action}. Analyze the reasoning behind this action choice.
- The progress made on the task before this action is: {context}.
- Focus on why this action is appropriate within the current context, using present tense as if actively solving the problem.
- Explain your intention in the first person.
- Format the intention in 2-3 sentences as follows: "To [goal or purpose], I choose to [action to take]. This allows me to [result or benefit]."

P.S.:
- When following these instructions, use natural language, avoid mentioning technical details (such as action coordinates) or direct use of action tags (such as "PRESS_HOME").

Output Format:
The output should be in JSON format as follows:
{{
"description": "2-3 sentences summarizing the current screenshot content in the desired format",
"intention": "2-3 sentences explaining reasoning for choosing this action in the desired format”
}}

Please return the result in pure JSON format, without any json tags like ```json ```.
```

Figure 13. Prompts for generating screen description and decision rationale.

Prompt for evaluating closed-source proprietary LVLMs
<img>current_screenshot.png</img> Given a device screenshot and an instruction, please provide the corresponding action. Available Actions: CLICK: <coordinate> LONG_PRESS: <coordinate> TYPE: <text> SCROLL: UP SCROLL: DOWN SCROLL: LEFT SCROLL: RIGHT PRESS_BACK PRESS_HOME PRESS_RECENT IMPOSSIBLE COMPLETE All <coordinates> are in the form (x, y), representing the coordinates to click or long press. The coordinate of the top-left corner is (0, 0), and the coordinate of the bottom-right corner is (1000, 1000). The instuction is: {instuction} The historical actions are: {history_actions} Based on the screenshots and the available actions, provide the next step directly.
Prompt for evaluating closed-source proprietary LVLMs with OmniParser
<img>current_screenshot.png</img> <img>current_screenshot_w_labels.png</img> Given two device screenshots and an instruction, provide the corresponding action. The first image is the original screenshot, and the second is the same screenshot with numeric tags on different interface elements. If the action requires clicking or pressing, choose the closest numeric tag that aligns with your intended location. Here are the Available Actions: CLICK: <element_idx chosen from the second screen> LONG_PRESS: <element_idx chosen from the second screen> TYPE: <text> SCROLL: UP SCROLL: DOWN SCROLL: LEFT SCROLL: RIGHT PRESS_BACK PRESS_HOME PRESS_RECENT IMPOSSIBLE COMPLETE The instuction is: {instuction} The historical actions are: {history_actions} Based on the screenshots and the available actions, provide the next step directly.

Figure 14. The prompt for evaluating closed-source proprietary Large Vision Language Models (LVLMs).