# GhostUI: Unveiling Hidden Interactions in Mobile UI


Minkyu Kweon
Seoul National University
Seoul, Republic of Korea
mk@hcil.snu.ac.kr

Seokhyeon Park
Seoul National University
Seoul, Republic of Korea
shpark@hcil.snu.ac.kr

Soohyun Lee
Seoul National University
Seoul, Republic of Korea
shlee@hcil.snu.ac.kr

You Been Lee
Seoul National University
Seoul, Republic of Korea
yblee2001@snu.ac.kr

Jeongmin Rhee
Seoul National University
Seoul, Republic of Korea
jmrhee@hcil.snu.ac.kr

Jinwook Seo*
Seoul National University
Seoul, Republic of Korea
jseo@snu.ac.kr


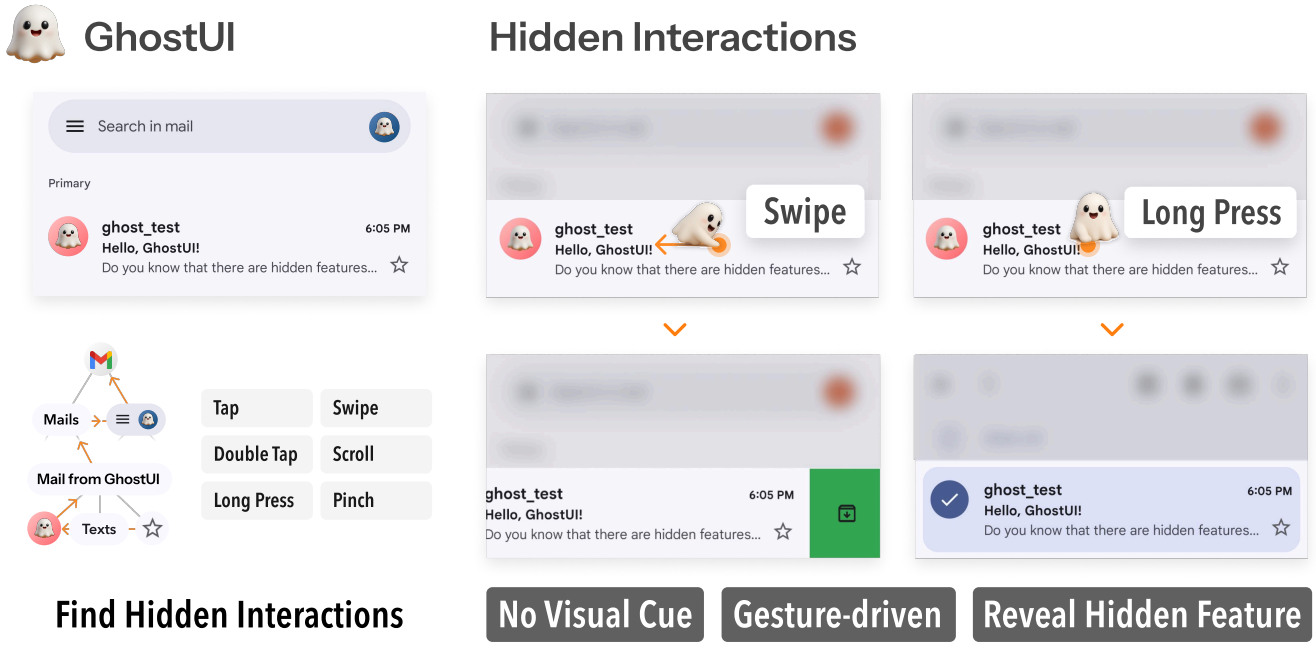


**Figure 1: GhostUI provides a dataset and framework for discovering *hidden interactions* in mobile UIs—interactions that lack visible cues but are triggered by gestures such as swipe, long press, or double tap. The dataset systematically documents these concealed interactions through automated probing of real-world mobile applications. *Hidden interactions* are characterized by (1) the absence of visual affordances, (2) gesture-driven activation, and (3) the revelation of previously inaccessible features.**


## Abstract

Modern mobile applications rely on *hidden interactions*—gestures without visual cues like long presses and swipes—to provide functionality without cluttering interfaces. While experienced users may discover these interactions through prior use or onboarding tutorials, their implicit nature makes them difficult for most users to uncover. Similarly, mobile agents—systems designed to automate tasks on mobile user interfaces, powered by vision language models (VLMs)—struggle to detect veiled interactions or determine actions for completing tasks. To address this challenge, we present GhostUI, a new dataset designed to enable the detection of *hidden interactions* in mobile applications. GhostUI provides before-and-after screenshots, simplified view hierarchies, gesture metadata, and task descriptions, allowing VLMs to better recognize concealed gestures and anticipate post-interaction states. Quantitative evaluations with VLMs show that models fine-tuned on GhostUI outperform baseline VLMs, particularly in predicting *hidden interactions* and inferring post-interaction screens, underscoring GhostUI's potential as a foundation for advancing mobile task automation.



*Corresponding Author




*CHI '26, Barcelona, Spain*



ACM ISBN 979-8-4007-2278-3/2026/04
https://doi.org/10.1145/3772318.3790283



## CCS Concepts

• **Human-centered computing** → **User interface toolkits**; **User interface design**; • **Computing methodologies** → **Mobile agents**.

## Keywords

Mobile User Interface, Hidden Interaction, Vision Language Model, Mobile Agent, UI Task Automation

**ACM Reference Format:**

Minkyu Kweon, Seokhyeon Park, Soohyun Lee, You Been Lee, Jeongmin Rhee, and Jinwook Seo. 2026. GhostUI: Unveiling Hidden Interactions in Mobile UI. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/3772318.3790283

## 1 Introduction

Modern mobile applications support a wide variety of interaction techniques to maximize the utility of limited screen space [46]. While directly visible user interface (UI) elements such as buttons and menus provide clear affordances, many applications also rely on interactions without explicit visual cues—such as gestures performed on seemingly static elements. We refer to these as *hidden interactions*: gestures like long presses or swipes that lack any visible indication that they are possible. *Hidden interactions* serve two primary purposes: *i)* providing shortcuts to frequently used functionality *ii)* revealing hidden UI elements, such as menus or options, that would otherwise consume valuable screen space [10]. These interactions are common in modern mobile experiences. For example, in Apple's iMessage conversation list, a two-finger downward pan gesture anywhere on the message list enables quick selection of multiple messages without entering edit mode. Similarly, in Instagram's direct message screen, long pressing on an individual message reveals hidden options like deletion or relocation—features that are typically inaccessible on Android without using this gesture.

The emergence of powerful vision language models (VLMs) [17, 75] has led to a surge in mobile agents [67] designed to automate UI tasks through natural language instructions, opening new possibilities for user interaction. While humans can discover and learn *hidden interactions* through exploration, onboarding tutorials, or prior experience with similar applications, VLM-powered agents face a fundamental challenge: they can only perceive what is visually present in a screenshot. This constraint makes it difficult for such systems to perform tasks that rely on *hidden interactions*, where the necessary actions are not visually apparent in the interface [39]. The challenge is further complicated by the many-to-many relationship between UI elements and gestures. A single UI element may respond differently to various gestures, and the same gesture may trigger different outcomes across different elements. For instance, in the YouTube video player, a single tap reveals the playback controls, a double tap skips forward or backward, and a long press enables 2x speed playback—all on the same UI area. Conversely, the same double tap gesture produces different outcomes depending on the target element: in Instagram, double tapping a post likes the content, while double tapping a profile icon in the navigation bar switches between multiple accounts. Without clear visual affordances indicating these possibilities, VLM-based agents cannot reliably infer the correct interaction.

Existing mobile UI datasets [3, 11, 15, 36, 53] for training VLMs primarily focus on visually apparent interactions, such as button presses, swiping carousel, and form inputs. These datasets are typically constructed from UI screenshots and accompanying metadata, which predominantly represent elements with explicit visual cues. Although recent advances in UI understanding have made progress—with OmniParser [43] improving semantic interpretation of visible components (*e.g.*, icons, text) and Ferret-UI [73] supporting any-resolution image analysis—they still assume that actionable elements exhibit visual affordances. As a result, both existing datasets and models for UI understanding largely ignore *hidden interactions*—despite their critical role in modern mobile experiences. Consequently, VLM-driven agents struggle to infer and execute these gestures, significantly limiting their effectiveness in automating real-world mobile tasks that depend on these non-obvious interactions.

To address these limitations, we present GhostUI, the first comprehensive dataset specifically designed to train and evaluate VLMs on *hidden interactions* across diverse mobile applications. In developing this dataset, we aim to answer the following research questions:

- **RQ1** What methodological framework can systematically uncover *hidden interactions* across diverse mobile applications?
- **RQ2** How can we identify, categorize, and document *hidden interactions* based on their gesture types and interaction patterns?
- **RQ3** What types of visual and structural information are critical for VLMs to accurately predict these interactions?

To answer these questions, we first conducted a comprehensive analysis of gesture design guidelines from major mobile platforms, specifically Android and iOS. This analysis allowed us to identify a set of gesture types that are frequently used as *hidden interactions* in a variety of applications. Based on these findings, we developed a robust data collection pipeline tailored to systematically gather diverse *hidden interactions* from real-world mobile applications. Our pipeline executes the identified gesture types in a controlled, repeatable manner across various mobile applications, capturing comprehensive interaction data, filtering and validating data instances to retain only those that represent *hidden interactions*, and generating detailed contextual task descriptions to reflect realistic user goals.

We organized the collected data by gesture types to enable systematic analysis of interaction patterns and gesture-element relationships across diverse contexts. This structured approach revealed common gesture patterns and characterized the nature of *hidden interactions* in mobile interfaces, yielding critical insights about their prevalence and visual characteristics. Through comprehensive VLM experiments, we identified which interaction data types are most critical for model performance and measured improvements in the models' ability to predict and understand *hidden interactions*. This evaluation provides quantitative evidence that GhostUI improves VLM performance on tasks involving gesture prediction and contextual UI understanding. Lastly, we discuss potential applications enabled by GhostUI.

Our paper makes the following key contributions:

(1) We introduce GhostUI, a dataset of 1,970 *hidden interaction* instances collected from 81 popular mobile applications. Each entry includes paired before-and-after screenshots, simplified view hierarchies, detailed gesture metadata, and a

contextual task description—structured for effective model training.
(2) We establish a formal taxonomy for *hidden interactions* in mobile applications, categorizing six distinct gesture types and their contextual usage patterns.
(3) We demonstrate that VLMs fine-tuned with GhostUI significantly outperform baseline models in both gesture classification and spatial localization. Our ablation studies further reveal the relative importance of different input features for detecting *hidden interactions*.

## 2 Related Work

### 2.1 Hidden Interactions in Mobile UIs

Our work builds upon the concept of affordance, originally introduced in ecological psychology [18], which describes the actionable properties of objects perceived by users. Norman [47] adapted this concept for interface design, emphasizing the importance of perceived affordances in determining usability. Gaver [16] further refined the concept by distinguishing between perceptible, hidden, and false affordances—where hidden affordances represent action possibilities that exist but lack visible cues for discovery. Recent work [41, 54, 58, 68] has expanded these ideas, using data-driven and learning-based methods to detect and represent affordances in both physical and graphical user interfaces (GUIs).

Interactions with clear visual cues, such as tapping a button or an icon, typically require minimal cognitive effort due to their intuitive affordances. However, a growing number of mobile interactions rely on gestures that lack visual cues and are difficult to discover without prior experience or trial-and-error. We define these as *hidden interactions* —interactions characterized by three properties: the absence of visual cues indicating interaction possibilities; gesture-specific activation that deviates from expected patterns for given UI elements; and functionality that is not apparent from the initial interface state. This prevalence stems from design constraints in mobile environments, where limited screen space drives designers to map multiple gestures onto single UI elements while preserving rich functionality [46]. While space-efficient, this approach often leaves users struggling to discover features, particularly when gestures vary across applications [31, 45]. As mobile applications become increasingly feature-rich and gesture-oriented, understanding and modeling these interactions becomes crucial for enhancing user experience.

Discovering *hidden interactions* requires exhaustively testing all gestures across all interactive elements. Users cannot predict which elements respond to which gestures or what outcomes these interactions will produce without actually trying them. Accordingly, we systematically applied six gesture types to interactive elements across popular applications. We then conducted manual verification to filter out non-functional interactions and validate genuine *hidden interactions*, combining automated exploration with human judgment to ensure accuracy. Through this semi-automated approach, we identified gesture-specific patterns revealing how *hidden interactions* manifest in practice. Our findings provide empirical evidence for improving interaction discoverability and offer designers insights into prevalent gesture-element mappings in contemporary mobile interfaces.

### 2.2 Mobile UI Understanding

***Mobile UI Datasets.*** Mobile UI datasets have been instrumental in advancing UI understanding. They provide large-scale training data for tasks such as UI design retrieval [6, 11, 49, 50], UI element detection [20, 71], and mobile task automation [32, 56, 63, 76]. RICO [11] laid the foundation with over 72k screenshots and view hierarchies from Android applications, while Enrico [33] added semantic annotations for design-based retrieval. Screen2Words [61] introduced screen-description pairs for natural language grounding, ScreenQA [23] provided question-answer pairs for UI reasoning, and UIBert [3] offered pre-training data for multimodal UI understanding. Recent datasets have targeted more complex interaction scenarios: MoTIF [7] studied task feasibility with sequences of both feasible and infeasible commands, AITW [53] captured human demonstrations for natural language-driven device control, and MobileViews [15] introduced automated collection pipelines for screen assistant tasks. ActionBert [21] demonstrated that user action sequences can reveal functional UI semantics beyond visual appearance.

***Mobile Task Automation.*** Early research in mobile task automation focused on grounding natural language to UI actions [36, 51], with subsequent work employing multimodal transformers [38] and interactive frameworks [35] to advance UI reasoning. The recent emergence of powerful vision language models (VLMs) like GPT-4 [1] and Qwen2.5-VL [5] has significantly accelerated progress in this domain, providing robust screenshot understanding and GUI analysis capabilities. Building on these foundations, specialized models such as ScreenAI [2], Ferret-UI [40], and OmniParser [43] have further enhanced UI understanding from visual, structural, and interaction perspectives. These advances have led to the development of numerous mobile agents that automate complex UI tasks through natural language instructions. Systems like AppAgentX [28], MobileAgent-E [62] and CogAgent [22] demonstrate diverse approaches to mobile automation.

However, our analysis of mobile agents reveals severely constrained action spaces, as shown Table 1—none support double tap or pinch gestures, and only five systems implement long press, despite these being fundamental interactions in modern applications. This limitation stems from a critical gap in existing datasets. While datasets have evolved from basic screenshot collections to complex task demonstrations, they predominantly capture interactions with clear visual affordances and simple gestures (*e.g.*, tap, scroll). Complex gestures like long press, double tap, and pinch remain largely undocumented, and *hidden interactions* are entirely absent. This dataset bias directly constrains system capabilities: agents cannot learn to perform interactions they have never seen in training data. The disconnect between the rich interaction vocabulary of real-world applications and the limited action spaces in current datasets fundamentally restricts practical deployment of mobile automation systems.

To address this gap, we present GhostUI, the first dataset to systematically document *hidden interactions* across six gesture types in 81 popular mobile applications. By expanding the action space and capturing interactions without visual cues, GhostUI enables future research toward truly comprehensive mobile UI understanding and automation. Our dataset not only reveals the prevalence of *hidden*

**Table 1: Comparison of Action Spaces across Mobile Agents and Interactive Environments**

| System | Tap | Double Tap | Long Press | Swipe | Scroll | Pinch |
|---|---|---|---|---|---|---|
| **GhostUI** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ***Mobile Agents*** | | | | | | |
| ResponsibleTA [78] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| DroidGPT [64] | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| AppAgent X [28] | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ |
| MobileAgent E [62] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| AutoDroid [63] | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ |
| VLUI [30] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| MetaGUI [57] | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| CogAgent [22] | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ |
| AutoGUI [77] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| UI-VLM [13] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| Coco-Agent [44] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| DigiRL [4] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| SphAgent [8] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| MobileVLM [69] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| OdysseyAgent [42] | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| ***Interactive Environment*** | | | | | | |
| AndroidEnv [59] | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ |
| AppBuddy [55] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Mobile-Env [74] | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ |
| AndroidWorld [52] | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ |
| DroidTask [63] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| B-MoCA [30] | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |

**Table 2: Definitions of Gesture Types used in GhostUI.**

| Gesture | Description |
|---|---|
| Tap | Single touch-and-release gesture at specific coordinates. |
| Double tap | Two consecutive taps with a brief pause between actions. |
| Long press | Extended touch at a single point, lasting for a duration. |
| Swipe | Touch movement from one position to another in a horizontal direction (left or right). |
| Scroll | Touch movement from one position to another in a vertical direction (up or down). |
| Pinch | Two-finger movement relative to a center point, either from close to far positions (zoom in) or from far to close positions (zoom out). |

*interactions* in contemporary mobile interfaces but also provides the necessary training data for next-generation mobile agents to achieve human-like interaction capabilities.

## 3 GhostUI

### 3.1 Action Space

We first identify gesture types commonly used in mobile user interfaces by inspecting platform design guidelines from Apple Human Interface Guidelines [26] and Google Material Design [12]. We focused on six fundamental gestures that are prevalent in mobile applications and frequently trigger functionalities with no visual affordances, which highlight critical gaps in current mobile automation systems (Table 1).

While these gestures are typically applied to elements with visual cues — tap for buttons and links, swipe on carousel indicators, scroll for scrollbars and overflowing content — we discovered they often trigger hidden functionality in contexts without explicit affordance. For instance, as shown in Figure 1, swiping horizontally on email items in Gmail archives messages without any visual indicators suggesting this action. Similarly, long pressing the same items enables multi-selection mode despite the absence of selection checkboxes. We systematically explore these six gesture types across mobile applications to uncover such *hidden interactions*.

### 3.2 Dataset Collection Method

To construct GhostUI, we implemented a three-phase data collection pipeline consisting of: *i)* systematic interaction discovery using

automated gesture testing across six gesture types, *ii)* validating and filtering *hidden interactions* through manual annotation, and *iii)* task contextualization with natural language descriptions. This methodology addresses **RQ1** by establishing a scalable pipeline that effectively captures *hidden interactions* across diverse mobile applications, yielding a task-oriented, multimodal dataset to advance intelligent mobile UI understanding.

*App Selection Criteria.* To ensure that our dataset reflects applications users commonly encounter, we focused on popular mobile apps and selected them from the Google Play Store[1] rankings as of March 5, 2025, starting from the top-ranked applications and proceeding in order. During this process, we excluded applications requiring subscriptions or external hardware (*e.g.*, Netflix, Bose), streaming services whose screen content could not be reliably captured due to DRM (Digital Rights Management) protection (*e.g.*, Pick Drama, ReelShort), and applications handling sensitive personal information, such as banking and financial services (*e.g.*, PayPal, Cash App). Applying these criteria yielded a final set of 81 applications. Despite the exclusions, the resulting set spans diverse categories (*e.g.*, *social*, *productivity*, *shopping*, *lifestyle*) and provides broad coverage of interaction designs commonly encountered in contemporary mobile applications.

*3.2.1* ***UI Probing Tool.*** We developed an automated UI probing tool to systematically discover *hidden interactions* across mobile applications. To facilitate reproducibility and enable crowdsourced data collection, we provide this tool as open-source on GitHub[2]. The tool uses Appium[3] to control both an Android Emulator (Google Pixel 7 Pro) and a physical device (Samsung Galaxy A16) to capture interactions across different device environments. As shown in Figure 2, this tool navigates through mobile application screens and applies every gesture to all elements of UI. To capture comprehensive UI state transitions, it records the complete before-and-after view hierarchies along with their simplified HTML-like representations, the element path, detailed gesture metadata, and corresponding screenshots for each interaction.

*Key Screen Selection Criteria.* Our approach prioritized screens central to the app's core functionalities, rather than exhaustively covering all screens. This strategy enhanced the efficiency of our data collection process, enabling us to gather interaction data from a wider variety of applications. To implement this approach, we define key screens as primary destinations within a mobile application that are directly accessible through top-level navigation (*e.g.*, navigation bar, tab bar). We adopted this definition from design guidelines, including Apple Human Interface Guidelines [26] and Google Material Design [12], which recommend placing important app destinations in persistent navigation components for optimal user experience.

*Element Detection and Tracking.* Our tool parses Android XML view hierarchies to identify interactive elements across screens. It employs a bottom-up traversal strategy, starting from leaf nodes and working upward to systematically test all elements while preventing touch point overlaps. It calculates precise bounding boxes for each element and, when determining touch points for parent nodes, carefully considers the spatial positioning of already-tested child elements to ensure non-overlapping interactions. For each element, it extracts coordinates, bounding boxes, and interactivity attributes, enabling targeted testing of elements that might support *hidden interactions*. To handle dynamic content where UI elements change between test sessions, we implemented a path-based element tracking mechanism. This approach maps each UI element to a unique hierarchical path using class names and indices from XML view hierarchies (*e.g.*, `FrameLayout[0]/View[0]/../Text[2]`), where each element receives a consistent index based on its position among siblings of the same type. Rather than relying on volatile properties like element size or position, this method ensures consistent identification across testing runs. By storing visited element paths in a persistent storage system classified by screen, our tool maintains consistent test coverage even when the app is restarted between testing runs or when the app's content changes. The system saves and restores test progress, allowing it to skip previously tested interactions and focus on unexplored UI elements, particularly valuable for testing dynamic applications like social media feeds where both content and spatial arrangement are frequently updated.

*Gesture Testing and State Monitoring.* The system executes six gesture types on identified elements: tap, double tap, long press, horizontal swipes (left, right), vertical scrolls (up, down), and pinch gestures (zoom in, zoom out). For directional gestures, we capture the specific direction of each interaction—distinguishing left from right swipes, up from down scrolls, and zoom in from zoom out pinches. To ensure reproducibility, we maintain consistent interaction parameters (*e.g.*, duration, distance) across different screens and element sizes. To detect *hidden interactions*, we capture and compare UI states at multiple points during gesture execution. For most gestures, we record before and after states to identify UI transitions. However, for long press and pinch gestures, we additionally capture a "during" state to detect transient effects—such as tooltips that appear only while pressing a UI element or temporary zoom indicators during pinching—that would otherwise be missed.

State detection employs two complementary approaches. We primarily compare view hierarchy trees to identify structural changes in the interface, such as new elements appearing or existing ones being modified. For pinch gestures, where visual changes often occur without hierarchy modifications (*e.g.*, image zooming), we supplement this with pixel-level screenshot comparison using a 5% average RGB difference threshold—empirically calibrated to balance robustness against noise while maintaining sensitivity to visual changes. This dual approach ensures we capture both structural and visual *hidden interactions*. When changes are detected, the system logs comprehensive interaction data including paired screenshots, view hierarchies, element paths, and gesture metadata with precise coordinate information—single points for taps, start-end trajectories for swipes, and multi-touch coordinates for pinches.

*Simplified View Hierarchy Conversion.* View hierarchies provide essential structural context for UI understanding, as demonstrated by prior work showing that models leveraging hierarchical information outperform purely visual approaches [3, 34, 37]. However, raw Android XML hierarchies contain excessive metadata that can

[1] https://play.google.com/
[2] `https://github.com/gh0stui/ghostui`
[3] https://appium.io/

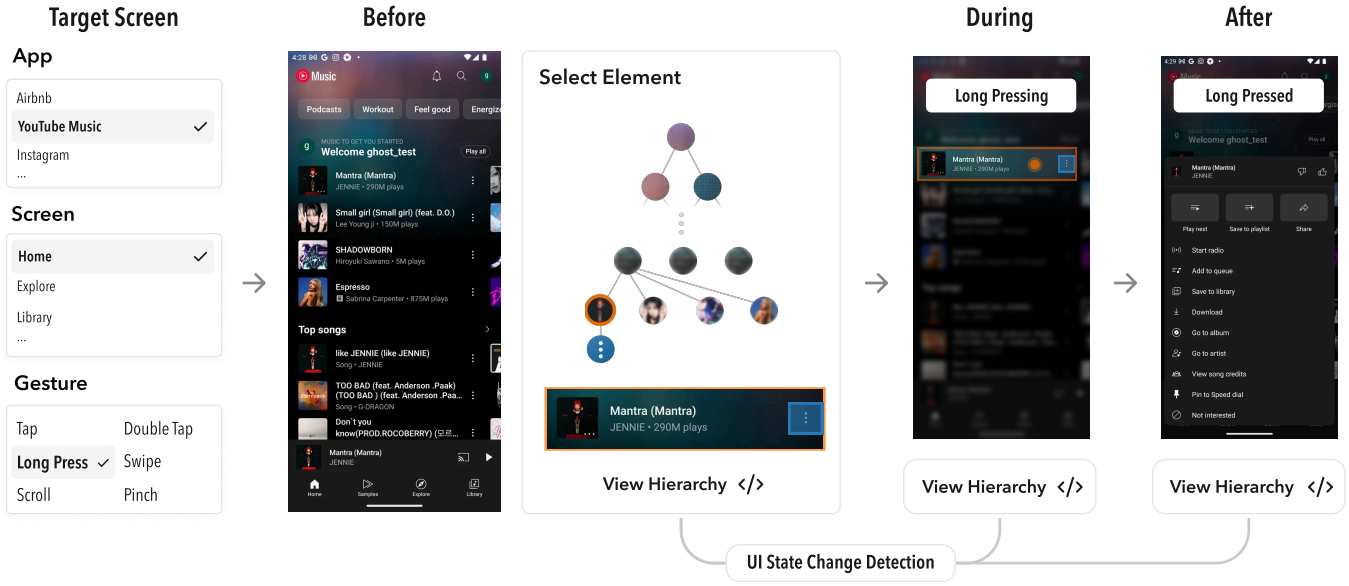


**Figure 2: Overview of the UI Probing Tool Operation.** The system identifies interactive UI elements through view hierarchy parsing, tracks elements across dynamic screens using path-based identification, executes diverse touch gestures, and monitors UI states to capture ephemeral changes.

overwhelm language models, these structures often exceed model context limits without improving performance. Recent work has shown that converting verbose hierarchies into simplified HTML-like representations improves model efficiency while maintaining semantic richness [60]. Following this approach, we developed a conversion routine that automatically transforms the captured XML view hierarchies into minimal, HTML-like structures. For each node in the XML tree, we map Android widget types (*e.g.*, `TextView`, `RecyclerView`) to semantically simpler HTML tags (*e.g.*, <p>, ) while preserving critical interaction properties (*e.g.*, `clickable`, bounds, `content-desc`) as HTML attributes. This conversion process reduces noise from Android-specific attributes and creates more accessible representations for downstream analysis. As demonstrated in Section 4.2, this simplified view hierarchy significantly improves VLMs' spatial localization performance in *hidden interaction* prediction tasks.

*3.2.2* ***Dataset Annotation and Validation.*** To ensure dataset quality and accurately identify true *hidden interactions*, we developed a web-based validation tool (shown in Figure 3) that enabled the authors to carefully review and filter all collected interaction data. We first established initial validation guidelines through consensus among five authors, then began distributed annotation. Cases that were ambiguous or difficult to classify (*e.g.*, discussions about what constitutes a visual cue for tap-based interactions) were flagged and brought to group meetings for resolution. Through this iterative process, we refined and finalized detailed validation guidelines, which were then applied to equally divided portions of the dataset. This manual validation process followed a structured assessment framework, ensuring consistent application across the entire dataset. Along with the UI probing tool described earlier, both the validation tool and detailed annotation guidelines are made publicly available to support reproducibility.

(1) **Interaction Validity**: We verified each interaction was error-free, with no error messages or unexpected behaviors. Unexpected behaviors include unintended gesture effects, such as a double tap triggering two separate taps or a long press acting like a standard tap when applied to elements that do not support these gesture types, or swipe gestures extending beyond the target UI area. We also confirmed that the interaction produced a meaningful UI state change, where *meaningful* denotes a visible change aligned with the application's intended functionality.
(2) **Visual Element Labeling**: To enable systematic assessment of visual affordances, we manually labeled UI elements using five visual categories: *border*, *text*, *icon*, *media* (*e.g.*, image, video, map), and *whitespace*. This minimal set, grounded in Atomic Design [14] and Gestalt theories [65], balances simplicity and expressiveness while capturing both content and layout-level visual cues. All elements within each gesture's bounding box were labeled using one or more of these categories, providing a structured basis for determining the visual affordance context of each interaction.
(3) **Hidden Nature**: We evaluated each interaction to determine whether it lacked obvious visual cues that would typically signal interactivity. Hidden nature was assessed not only from the element itself but also from its surrounding UI context. For tap gestures, we required that the tapped element contained only *whitespace* or *media* without conventional affordances (*e.g.*, borders, icons, stylized text) and that no contextual cues in the surrounding layout suggested tappability. For double tap and long press, we included cases where the gesture was applied

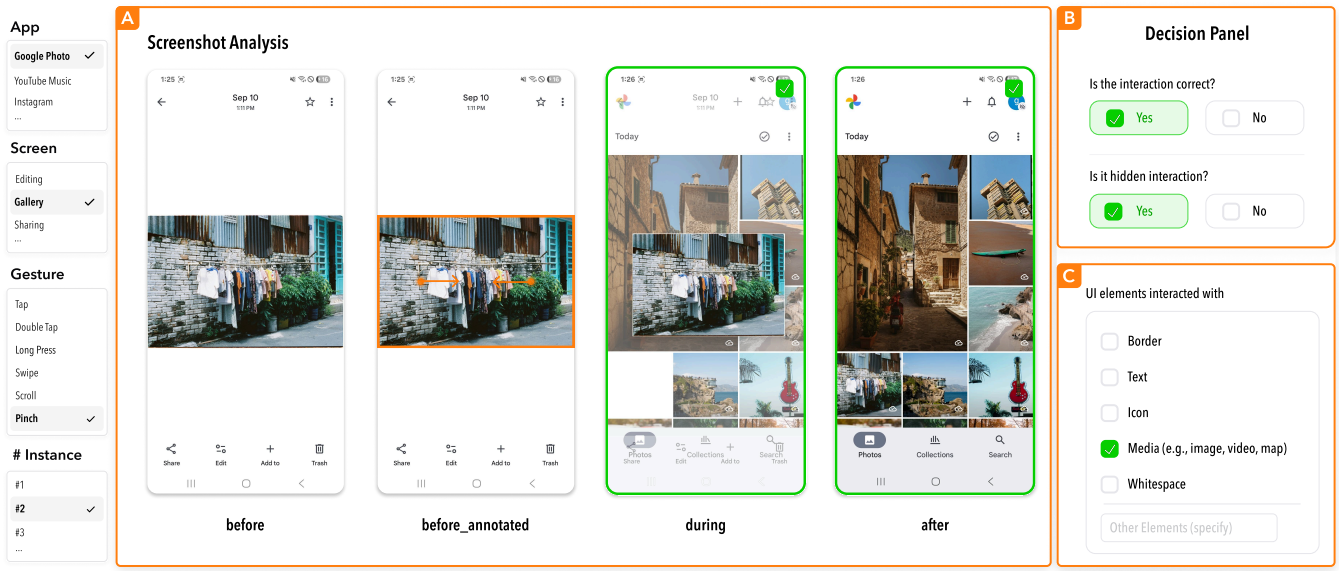


**Figure 3: Overview of Validation Tool Interface for manual annotation and filtering of collected interaction data. (A)** Screenshot comparison panel displaying temporal UI states (before, before_annotated with red outline indicating the target element) to facilitate visual change detection and outcome verification. **(B)** Decision panel for assessing interaction validity and determining hidden nature. **(C)** Visual element labeling panel for categorization of UI components within the target element's bounding box. In this example, only *media* is selected as it represents the visual content within the target area.

to the same element as a standard tap but produced distinct behaviors; in such cases, the interaction was considered hidden even if visual cues were present, since the alternate behaviors could not be inferred from design alone. For swipe and scroll gestures, we confirmed the absence of contextual indicators such as cropped content, pagination dots, or scroll bars. For pinch gestures, we verified that no visual cues suggested zoom functionality, such as zoom control icons or ratio displays.

To assess the reliability of this annotation framework, we conducted an inter-annotator agreement study. We sampled 242 interactions from the full dataset using stratified sampling to ensure proportional representation across all gesture types. All five annotators independently evaluated these samples using our established guidelines. We calculated *Fleiss' Kappa* [29] to measure agreement across two annotation dimensions: for interaction validity and hidden nature assessment, we achieved $\kappa$=0.89 (almost perfect agreement); for visual element labeling, we achieved a mean $\kappa$=0.76 across the five element categories (substantial agreement).

Representative examples of validated *hidden interactions* across all gesture types are provided in Figure 4, illustrating the diverse visual contexts and interaction patterns captured in our dataset.

*3.2.3* ***UI Task Generation.*** To contextualize *hidden interactions* within realistic usage scenarios, we generated natural language task descriptions for each interaction in the dataset. These descriptions express the user intent and expected outcome, such as *"Send a voice message in the chat"* for a long press on the voice icon in Messenger. We leveraged GPT-4o [25] to analyze the before and after states of each interaction and infer the corresponding user intent. VLMs have proven effective at understanding UI semantics and generating action descriptions in recent benchmarks [2, 52]. The model generated descriptions based on the UI context, interaction type, and resulting state changes. We then manually reviewed these descriptions to ensure accuracy and naturalness, with the detailed prompt template provided in Appendix A.

These task descriptions serve as multimodal supervision signals for training VLMs in affordance reasoning—learning to map high-level user intentions to concrete interaction actions. Given a visual UI state and a natural language goal, models must learn to predict both the appropriate gesture type and interaction location. This formulation enables end-to-end training for intent-to-action prediction, where VLMs learn the complex mapping between what users want to accomplish and the *hidden interactions* required to achieve those goals. By providing this rich link between visual context, user goals, and interaction outcomes, GhostUI offers a foundation for training models to understand and execute realistic mobile interactions.

## 3.3 Dataset Fields

As illustrated in Figure 5, each data sample in GhostUI contains the following core information. The complete dataset is publicly available on HuggingFace[4].

- **Screenshots:** For every interaction, we provide before and after UI screenshots, and an additional "during" screenshot for long press and pinch gestures.

[4] https://huggingface.co/datasets/ghostui/ghostui

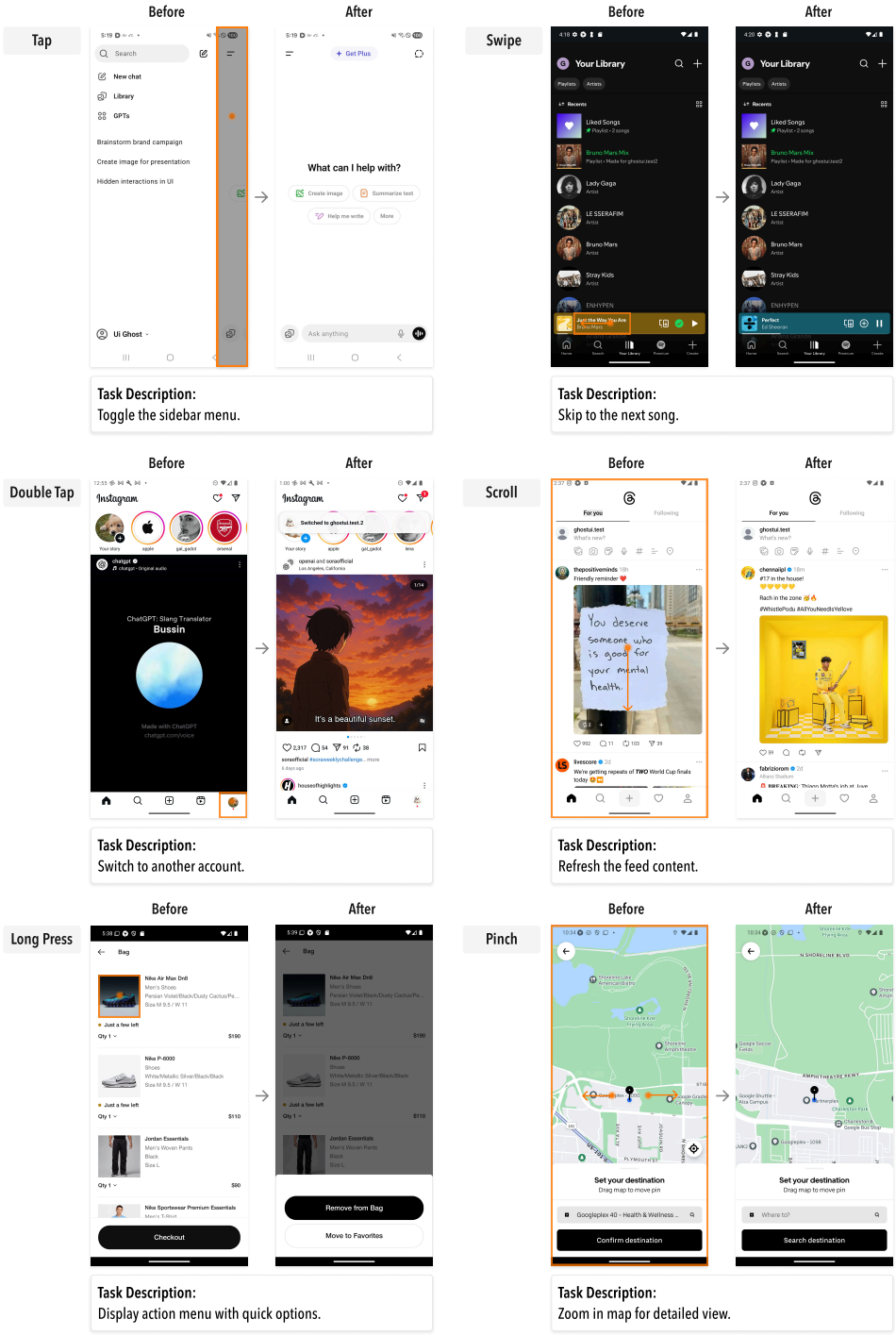


**Figure 4: Examples of *Hidden Interaction* across six gesture types: tap, double tap, long press, swipe, scroll, and pinch.** For each gesture type, paired before and after screenshots from real mobile apps illustrate the visual effect of interacting with a specific UI component (highlighted in orange in the before state). These examples illustrate the visual transitions that occur as a result of specific gestures applied to targeted elements.

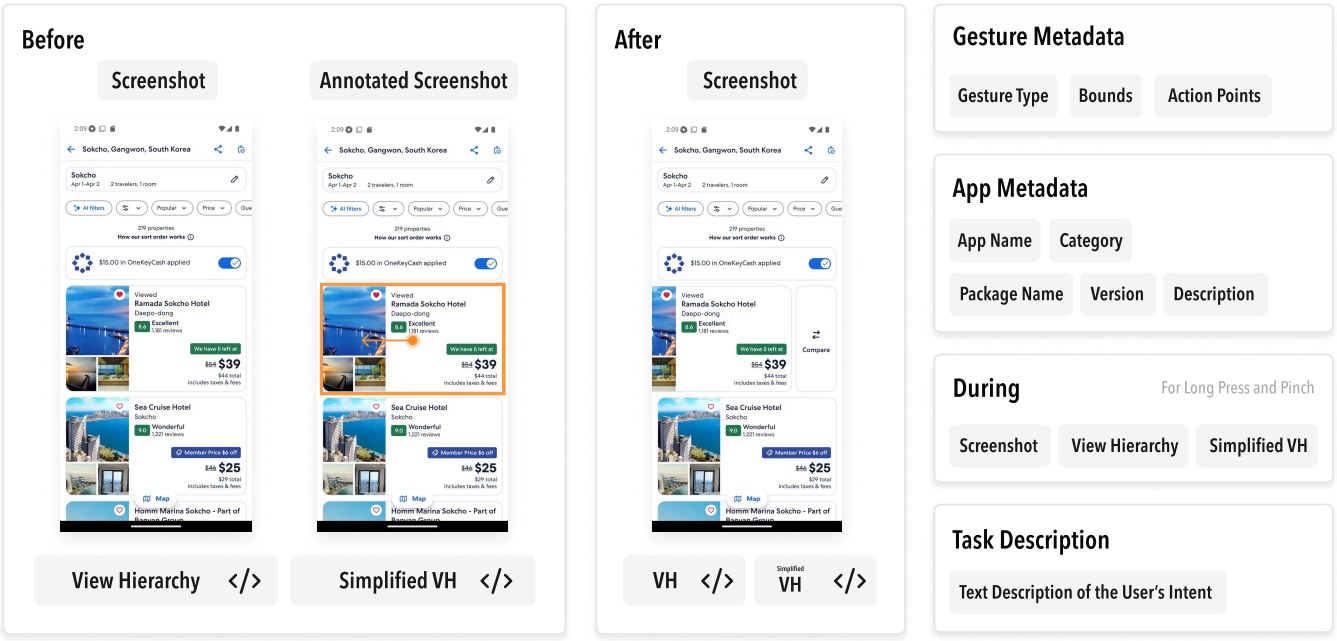


**Figure 5: Structure of the GhostUI, showing the comprehensive documentation of *hidden interaction*.** Before-and-After screenshots with view hierarchies (during state only included for long press and pinch gestures), action, app metadata, and a task description.

- **View Hierarchies:** Both the complete Android XML view hierarchy and a simplified HTML-like representation, preserving key attributes such as `clickable` or bounds.
- **Gesture Metadata:** The gesture type, the corresponding bounding box, and action points.
- **Task Description:** A natural language statement expressing user intent, which serves as natural language supervision for training and evaluating VLMs on intent-to-interaction understanding.
- **App Metadata:** High-level details about the application (*e.g.*, category, description) from Google Play Store.

By consolidating this diverse information into each instance, GhostUI provides a robust multimodal foundation for understanding how *hidden interactions* appear, how they alter the interface, and under what context they occur.

## 3.4 Dataset Statistics

GhostUI contains 1,970 validated *hidden interaction* instances collected from 81 popular mobile applications. Our automated UI probing tool initially gathered 8,312 interaction instances, which were then filtered through our thorough validation process to identify *hidden interactions*. This filtering process shows that *hidden interactions* require specific interaction patterns to discover, making them challenging for users and AI systems to identify without systematic exploration. To address **RQ2**, we systematically categorized and analyzed *hidden interactions* based on their gesture types and interaction patterns. Our analysis examined four key dimensions: gesture type distribution across *hidden interactions*, functional patterns and context-dependent usage across gesture types, visual element composition and co-occurrence within interactive regions, and the relationship between gesture-element combinations and interaction visibility.

*3.4.1* ***Distribution of Hidden Interactions Across Gesture Types.*** As shown in Table 3, *hidden interactions* in our dataset are distributed unevenly across gesture types. Tap gestures are the most prevalent (30.3%), representing the most fundamental input in mobile interfaces while also accounting for a substantial portion of *hidden interactions*. Swipes (26.0%) and long presses (19.3%) also account for large proportions, reflecting their frequent use for navigating content and for accessing additional options such as contextual menus. Double taps (9.5%) and pinches (8.9%) appear less frequently overall but remain essential in specialized contexts. Scroll gestures represent the smallest portion (6.0%), likely because scrolling is conventionally tied to continuous navigation rather than triggering hidden functionality. This distribution highlights how both fundamental and specialized gestures contribute to *hidden interactions* across diverse mobile applications.

*3.4.2* ***Gesture Usage Patterns.*** The representative usage patterns in Table 3 highlight the functional diversity of *hidden interactions* across applications. To derive these patterns, we applied topic modeling using *Latent Dirichlet Allocation* (*LDA*) [27], setting $k$=3 topics per gesture. This value was selected based on preliminary coherence tests to balance interpretability and coverage. For each task description, we extracted verb and noun phrases that capture the action–object structure of the interaction, and used them as inputs for clustering. This process yielded semantically coherent groups of interaction intents (*e.g.*, *inspect photos*, *zoom maps*, *manage conversations*), which we then distilled into representative usage categories for each gesture type. Our findings show both diversity

**Table 3: Distribution and Usage Patterns of *Hidden Interactions* across gesture types.** The table shows frequency counts, percentages, and representative usage patterns in GhostUI.

| Gesture | Direction | Count (%) | Representative Usage Patterns |
|---|---|---|---|
| Tap | - | 596 (30.3%) | Viewing multimedia content<br>Exploring detailed information<br>Toggling navigation elements |
| Double Tap | - | 188 (9.5%) | Engaging with social media content and profiles<br>Interacting with maps and location-based services<br>Searching products and services |
| Long Press | - | 379 (19.3%) | Opening contextual menus or hidden options<br>Selecting and managing items<br>Triggering secondary actions |
| Swipe | Left / Right | 513 (26.0%) | Navigating across content types or screens<br>Managing conversations and notifications<br>Exploring product categories or item lists |
| Scroll | Up / Down | 118 (6.0%) | Discovering new recommendations<br>Revealing extended menus or UI states<br>Browsing continuous feeds |
| Pinch | Zoom In / Out | 176 (8.9%) | Inspecting photos and other visual content<br>Zooming in/out on maps for spatial details<br>Adjusting perspective or layout views |

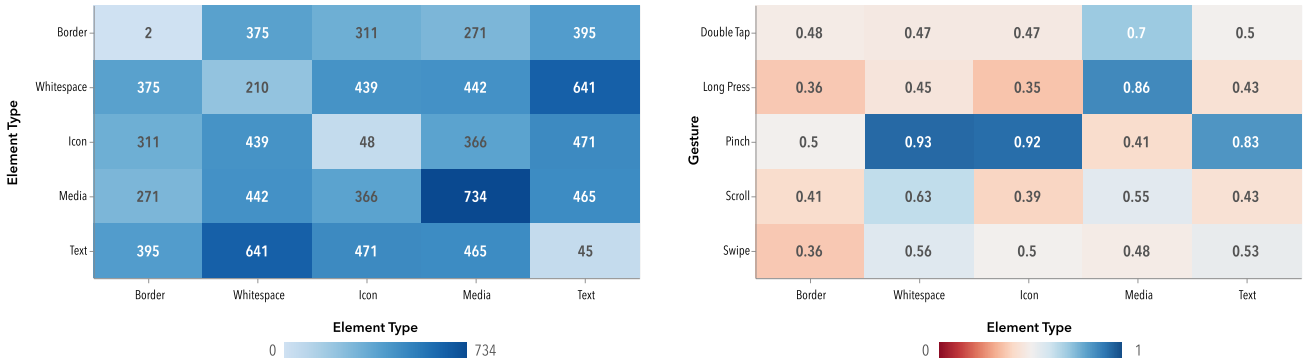


**Figure 6: Element Label Co-occurrences (left) and Gesture-Element Distributions by element type (right).** The heatmap on the left shows how often different visual labels (*e.g.*, *border*, *whitespace*, *icon*, *media*, *text*) overlap within the same bounding box, revealing frequent single-label regions (diagonal entries) and multi-label patterns (off-diagonal cells). The heatmap on the right displays each gesture type's relative association with hidden vs. open interactions across element labels, highlighting specific label–gesture pairings more likely to indicate *hidden interactions*.

and redundancy: distinct gestures often lead to similar outcomes across contexts. For example, double tap is widely used for content appreciation in social media apps, while both double tap and pinch gestures provide zooming functionality in maps and image viewers. Similarly, long press and swipe can both reveal hidden options, though their primary roles differ—long press for item selection and contextual menus, and swipe for navigation across content streams.

*3.4.3* ***Element Label Composition and Co-occurrence.*** To better understand the visual context associated with *hidden interactions*, we analyzed the semantic composition of UI elements involved in these gestures. During annotation, each target element's bounding box was labeled using five visual categories: *border*, *icon*, *text*, *media*, and *whitespace*. Our probing tool applies gestures to potentially interactive elements identified from XML view hierarchy (*e.g.*, `clickable` attributes for tap, double tap, long press). However, interactive regions often encompass multiple visual components within a single container. For instance, when only parent containers are marked as `clickable` while child elements are not, tapping different visual regions within the container may produce the same interaction outcome. The orange-highlighted component in Figure 2 exemplifies this behavior, where tapping different areas,

such as album art, title text, artist name, or surrounding whitespace, all yield the same result (excluding the blue-boxed "more" icon, which triggers a distinct action). In such cases, we assign multiple labels to regions within the bounding box that yield this shared behavior, explaining why container-based interactions frequently involve multiple visual element types.

As these labels are not mutually exclusive, we first examined how frequently different label combinations occurred together. Figure 6 (left) shows a heatmap representing the co-occurrence frequencies of label pairs. Diagonal cells correspond to instances where only a single label was present. Notably, *media* frequently appears as the sole label, suggesting a tendency for this type to occupy isolated visual regions. In contrast, labels such as *text*, *icon*, and *border* commonly co-occur with other types, reflecting their integration into composite UI components. This behavior is particularly common with text elements, where containers wrap both textual content and surrounding whitespace, explaining the high *text-whitespace* co-occurrence observed in our data. Surprisingly, we observed 210 instances of *whitespace* being labeled alone, highlighting the prevalence of hidden gestures in visually unmarked areas.

*3.4.4* ***Gesture-Element Distribution by Visibility Context.*** To further assess the affordance context of *hidden* versus *non-hidden* interactions, we compared the normalized gesture-element distributions across both conditions. For each gesture and element type pair, we computed the relative proportion of occurrences in hidden versus open settings. The resulting heatmap in Figure 6 (right) uses a diverging color scale (blue for hidden, red for open), where values closer to 1 indicate strong association with *hidden interactions*. The visualization reveals distinct patterns in how *hidden interactions* manifest across different visual contexts. *Border* elements show the weakest association with *hidden interactions* across all gestures, indicating that bordered elements typically provide clearer visual affordances. In contrast, elements without explicit boundaries exhibit strong *hidden interaction* patterns: *whitespace* shows high associations with hidden pinch and scroll gestures, while *media* elements consistently support *hidden interactions* for long press and double tap. Similarly, areas containing *icon* and *text* elements frequently support hidden pinch gestures. These findings suggest that the absence of visual boundaries correlates strongly with hidden functionality, requiring users to rely on exploratory behavior or prior knowledge to discover interactions.

*3.4.5* ***Summary and Implications.*** Our analysis of GhostUI reveals that *hidden interactions* are not edge cases but systematic patterns in mobile design. Specialized gestures (long press, double tap, pinch) constitute 37.7% of hidden functionality yet lack both visual indicators and automation support. The container-based architecture creates ambiguous interaction targets where visually distinct regions trigger identical responses, while 210 instances of isolated *whitespace* interactions demonstrate that completely unmarked areas have become legitimate targets. These empirical patterns provide a foundation for understanding how *hidden interactions* manifest in real-world mobile interfaces and quantify the discoverability challenges that both users and automated systems face when interacting with modern mobile applications.

# 4 Experiment

In this section, we present experiments designed to evaluate the effectiveness of GhostUI in enhancing *hidden interaction* understanding in VLMs. Our experiments address **RQ3** by *i)* identifying critical information for VLMs in understanding *hidden interactions* and *ii)* validating the efficacy of our dataset through quantifiable improvements. We evaluate performance on two key tasks: *Hidden Interaction Prediction*, which evaluates whether a model can identify the correct gesture type and interaction location given a UI screenshot, and *UI Transition Prediction*, which measures how accurately the model can anticipate the interface changes when given a specific *hidden interaction*. Complete prompt templates used in our experiments are provided in Appendix B.

## 4.1 Experimental Setup

*Models.* We selected two state-of-the-art VLMs representing different model scales: Qwen2.5-VL [5], a smaller open-source model with 7B parameters, and GPT-4o [25], a larger closed-source model with significantly more parameters. For each model, we established zero-shot performance as our baseline and compared it against models fine-tuned on GhostUI. We employed Low-Rank Adaptation (LoRA) [24] for Qwen2.5-VL and the Vision Fine-tuning API [48] for GPT-4o.

*Dataset Split.* To ensure robust evaluation and prevent information leakage from app-specific interaction patterns, we split the dataset such that training and test sets contain entirely different applications. Specifically, we allocated 56 apps for training and 25 apps for testing, maintaining an approximately 70:30 sample ratio while carefully balancing gesture type distributions across both splits. This app-level split ensures that models must learn generalizable patterns rather than memorizing app-specific behaviors.

*Task Definition.* We consider two complementary tasks to measure how well VLMs handle *hidden interactions*. First, *Hidden Interaction Prediction* (Section 4.2) requires predicting which gesture (*e.g.*, tap, long press) is needed and localizing it on the screen, given a before-interaction screenshot and a task description. Second, *UI Transition Prediction* (Section 4.3) tests whether the model can anticipate changes caused by a given *hidden interaction* by requiring it to generate a text description of the post-gesture UI, which is then compared to a ground truth narrative.

*Evaluation Metrics.* In Section 4.2, we evaluated model performance using two complementary metrics. First, classification accuracy measures whether the model correctly predicts the gesture type, including directional specifications for swipes (left, right), scrolls (up, down), and pinch gestures (zoom in, zoom out). Second, Intersection over Union (IoU) quantifies the overlap between predicted and ground truth bounding boxes to assess spatial localization precision. For Section 4.3, we measured the model's understanding of interaction outcomes by calculating cosine similarity between predicted UI descriptions and ground truth descriptions generated from after-interaction screenshots.

**Table 4: *Hidden Interaction Prediction* Performance (%) across input configurations.** All-inclusive denotes the default setting with all input modalities: screenshot, view hierarchy, gesture usage pattern, and app metadata. Accuracy indicates correct gesture type classification. IoU measures spatial precision based on the overlap between predicted and ground truth bounding boxes. Relative changes from the All-inclusive setting are shown in parentheses.

| Model | Setting | All-inclusive | | w/o View Hierarchy | | w/o Gesture Pattern | | w/o App Metadata | | Image only | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | IoU | Acc. | IoU | Acc. | IoU | Acc. | IoU | Acc. | IoU |
| GPT-4o | Zero-shot | 51.1 | 36.0 | 49.6 (↓1.5) | 3.4 (↓32.6) | 48.2 (↓2.9) | 35.0 (↓1.0) | 50.9 (↓0.2) | 35.4 (↑0.6) | 46.3 (↓4.8) | 3.5 (↓32.5) |
| | Fine-tuned | 65.6 | 42.5 | 62.2 (↓3.4) | 4.9 (↓37.6) | 57.3 (↓8.3) | 44.0 (↑1.5) | 64.4 (↓1.2) | 41.8 (↓0.7) | 52.8 (↓12.8) | 5.7 (↓36.8) |
| Qwen2.5-VL | Zero-shot | 33.3 | 19.5 | 36.8 (↑3.5) | 1.7 (↓17.8) | 12.0 (↓21.3) | 23.0 (↑3.5) | 34.4 (↑1.1) | 19.4 (↓0.1) | 19.5 (↓13.8) | 1.9 (↓17.6) |
| | Fine-tuned | 40.5 | 22.8 | 42.2 (↑1.7) | 6.2 (↓16.6) | 36.9 (↓3.6) | 28.8 (↑6.0) | 43.8 (↑3.3) | 21.2 (↓1.6) | 41.7 (↑1.2) | 5.0 (↓17.8) |

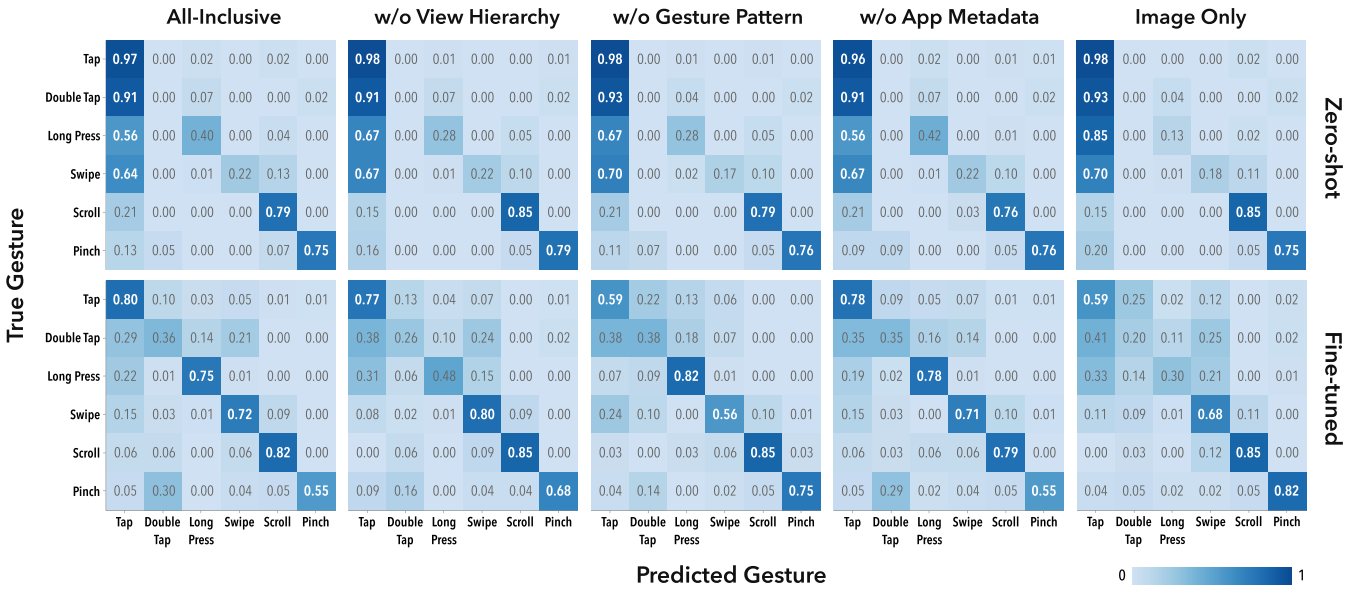


**Figure 7: Confusion Matrices showing gesture type classification performance.** The matrices compare GPT-4o's performance across five input configurations in both zero-shot and fine-tuned settings. The diagonal values represent correct gesture classification probabilities, while off-diagonal values indicate confusion between gesture types. Higher values along the diagonal indicate better performance, with the fine-tuned models demonstrating significant improvements in classification accuracy compared to their zero-shot counterparts, particularly in the All-inclusive configuration (bottom-left).

## 4.2 Hidden Interactions Prediction

To identify which input components contribute most significantly to VLM performance for *hidden interaction* prediction, we conducted a comprehensive ablation study followed by fine-tuning experiments.

*4.2.1* ***Experimental Design.*** Models were provided with a task description and before-interaction screenshot, then asked to predict the appropriate gesture type (*e.g.*, scroll down, pinch zoom in) and the precise location where the gesture should be applied to complete the task successfully. We designed our experiments around 5 distinct input configurations to isolate the contribution of each information type. Our complete configuration (*All-inclusive*) included before screenshot, simplified view hierarchy, gesture usage patterns described in Section 3.4.2, and app metadata. We then removed individual resources to create four additional configurations: one without the simplified view hierarchy, another without gesture usage patterns, a third without app metadata, and finally a minimal configuration with only before screenshot (*Image only*).

*4.2.2* ***Results.*** As shown in Table 4, our experiments revealed several key insights into *hidden interaction* prediction. In zero-shot settings, GPT-4o demonstrated superior performance compared to Qwen2.5-VL across all input configurations, achieving 51.1% accuracy and 36.0% IoU with the complete information sources (*All-inclusive*), while Qwen2.5-VL reached 33.3% accuracy and 19.5% IoU. The removal of simplified view hierarchy information resulted in a substantial decrease in spatial prediction accuracy, with IoU dropping by 32.6% for GPT-4o and 17.8% for Qwen2.5-VL, while gesture classification accuracy remained relatively stable. This indicates

the critical importance of structural UI information for precisely localizing interaction targets. Notably, the absence of gesture pattern information had particularly severe consequences for Qwen2.5-VL's performance, causing accuracy to plummet from 33.3% to 12.0%. This suggests that smaller models rely more heavily on explicit gesture contextual information to make accurate predictions.

After fine-tuning on GhostUI, both models exhibited notable performance improvements. In the *All-inclusive* configuration, GPT-4o achieved 65.6% gesture classification accuracy and 42.5% IoU, while Qwen2.5-VL reached 40.5% and 22.8% respectively. The performance gains over zero-shot baselines (14.5% accuracy for GPT-4o, 7.2% for Qwen2.5-VL) demonstrate that GhostUI provides transferable knowledge for predicting *hidden interactions* in unseen applications. Removing gesture usage patterns caused an 8.3% accuracy drop for fine-tuned GPT-4o (65.6% to 57.3%), indicating that gesture conventions serve as valuable prior knowledge. Removing the simplified view hierarchy continued to severely impact spatial localization across both models, while removing app metadata had minimal effect, suggesting that high-level application descriptions provide little actionable information for *hidden interaction* prediction.

To better understand how well the model distinguishes between individual gesture types, we analyzed gesture-wise confusion matrices for GPT-4o across five input configurations in Figure 7. We focused on GPT-4o for this in-depth analysis because it demonstrated consistently superior performance compared to Qwen2.5-VL on *hidden interaction* prediction, allowing us to examine subtle patterns in gesture type classification without being confounded by overall model performance issues.

In zero-shot settings, the model exhibited a severe bias toward predicting tap gestures. Across all configurations, double tap instances were misclassified as tap over 91% of the time, and long press instances showed 56–85% tap misclassification rates depending on input configuration. Scroll and pinch gestures were relatively better preserved even in zero-shot settings, achieving 76–85% and 75–79% accuracy respectively, likely because their distinct interaction patterns provide clearer differentiation signals.

Fine-tuning alleviated this tendency, leading to more balanced and accurate predictions across gesture types. In the *All-inclusive* configuration, double tap recognition improved from 0% to 36%, long press accuracy increased from 40% to 75%, and swipe recognition improved from 22% to 72%. The visualization in Figure 7 reveals a progressive improvement in diagonal pattern clarity as we move from zero-shot to fine-tuned models and from limited to more comprehensive input configurations. This strengthening of the diagonal elements visually confirms the model's increasing ability to correctly distinguish between different gesture types. The fine-tuned models show markedly clearer diagonal patterns compared to their zero-shot counterparts, demonstrating that task-specific training on GhostUI significantly improves the model's ability to differentiate between gesture types regardless of the input configuration used.

However, certain challenges persisted even after fine-tuning. Double tap remained the most difficult gesture to classify (36% accuracy), with substantial confusion toward tap (29%) and swipe (21%). Pinch gestures showed notable confusion with double tap (30% misclassification), likely because both gestures often target similar visual contexts such as images and maps. The confusion matrices also reveal configuration-specific patterns: without view hierarchy, long press accuracy dropped from 75% to 48% with increased tap confusion; without gesture patterns, swipe recognition degraded from 72% to 56%. The image-only configuration showed the most severe degradation for long press (30% accuracy), confirming that this gesture type is particularly dependent on structural information.

## 4.3 UI Transition Prediction

While Section 4.2 focuses on identifying appropriate gestures and their locations, *UI Transition Prediction* evaluates whether models can accurately anticipate the interface changes that result from performing these interactions. This task provides a complementary measure of *hidden interaction* understanding, as it requires models to comprehend not only where and how to interact, but also the consequences of those interactions. Understanding gesture consequences is a key component of comprehensive interaction reasoning, as effective planning depends on anticipating post-gesture outcomes—a critical capability for real-world mobile agents.

*4.3.1* ***Experimental Design.*** In this experiment, models were provided with a before-interaction screenshot and action details (gesture type and bounding box location), then tasked with predicting the visual changes that would result from performing the specified gesture. We evaluated predictions by comparing them against descriptions of the actual after-interaction screenshots. Following prior work that demonstrates GPT-4's ability to generate accurate and fine-grained UI descriptions comparable to human annotations [2, 50, 60, 70], we used GPT-4o to generate ground truth descriptions from the after-interaction screenshots. To reduce sensitivity to prompt variations and description detail differences, we measured performance by computing the cosine similarity between text embeddings of the predicted and ground truth descriptions using Google's `gemini-embedding-001` model with *semantic similarity* mode. This metric focuses on semantic alignment rather than exact lexical matching, where higher values indicate better understanding of gesture effects.

*4.3.2* ***Results.*** As shown in Figure 8, fine-tuning on GhostUI improved both models' ability to predict interface changes resulting from *hidden interactions* across most gesture types. We conducted a Wilcoxon Signed-rank test [66] to assess statistical significance ($^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$). Both models showed significant improvements for five of six gesture types: swipe and scroll ($p < 0.001$), tap ($p < 0.001$ for GPT-4o, $p < 0.01$ for Qwen2.5-VL), pinch ($p < 0.001$ for GPT-4o, $p < 0.05$ for Qwen2.5-VL), and long press ($p < 0.05$).

Double tap was the only gesture type showing no significant improvement ($p > 0.05$). Analysis of model outputs revealed that zero-shot models exhibited a strong bias toward predicting zoom-related outcomes, despite zoom being rare in our test set's ground truth. Additionally, double tap showed high outcome diversity, with context-dependent results varying from content engagement to text selection. These factors made it difficult for fine-tuning to overcome pre-existing biases. Overall, these findings indicate that fine-tuning on GhostUI improves gesture-induced UI transition

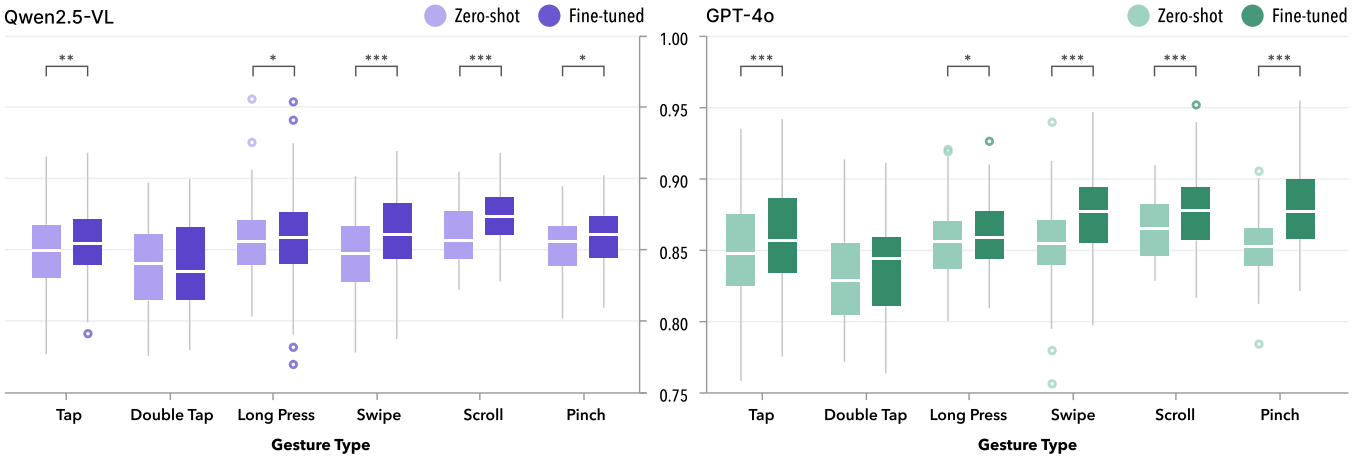


**Figure 8: Cosine Similarity between predicted and ground truth after-interaction screenshot descriptions for each gesture type, comparing zero-shot and fine-tuned variants of Qwen2.5-VL (left) and GPT-4o (right).** Fine-tuning improves performance across most gesture types, with double tap being the exception. Boxes represent interquartile ranges, with medians shown as horizontal bars. Statistical significance between zero-shot and fine-tuned variants is indicated by asterisks ($^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$).

prediction, though gestures with high outcome diversity and strong pre-training biases remain challenging.

## 5 Potential Applications

Based on our demonstration that models trained on GhostUI can effectively predict gesture type, target region, and ensuing UI state changes, this section describes mock-up applications enabled by GhostUI, highlighting its potential across user guidance and interaction design.

### 5.1 Interactive Guidance System for Hidden Interactions

Users often struggle to discover hidden features in mobile apps, missing functionality that could enhance their experience. An interactive guidance system powered by GhostUI could reveal these interactions through contextual assistance. As shown in Figure 9, when a user asks *"How can I play the reels at 2x speed?"*, the system identifies that long pressing on the left or right edges of the video immediately plays it at 2x speed while held, displaying translucent overlays to guide the interaction. Similarly, for camera mode switching (right), the system reveals that double tapping anywhere on the camera screen switches to selfie mode—a hidden alternative to the visible flip icon that users might not be aware of. This approach preserves clean interfaces while ensuring users can discover hidden functionality precisely when they need it, following progressive disclosure principles where hints appear contextually and fade after successful execution.

### 5.2 Interaction Design Recommendations

Designers currently lack data-driven methods to evaluate whether their *hidden interaction* implementations align with user expectations and platform conventions. GhostUI enables automated design conformance analysis by comparing implementations against established patterns from popular apps. As illustrated in Figure 10, when a designer wants to implement disappearing messages (left), the system analyzes the target element (*whitespace*) and recommends a scroll-up gesture based on empirical patterns where upward scrolls commonly reveal ephemeral controls. It warns against using tap on *whitespace*, as this gesture typically dismisses keyboards rather than triggering hidden features, potentially confusing users. For adding features to chat bubbles (right), the system provides a comprehensive mapping of gestures to their appropriate functionalities: tap for viewing message details, double tap for reactions, long press for contextual menus, and swipe for quick reply or archive actions. This empirical feedback helps designers make informed decisions that balance innovation with familiarity, ensuring *hidden interactions* remain discoverable through consistency with established patterns.

## 6 Discussions

GhostUI provides a foundation for understanding *hidden interactions* in mobile UIs, while also opening numerous avenues for future research. In this section, we discuss the implications of our findings and outline promising directions to extend our work.

### 6.1 Advanced Mobile Agents

In GhostUI, 23.7% of all validated interactions are classified as *hidden interactions*, underscoring the substantial portion of mobile functionality that current agents struggle to address. Among these *hidden interactions*, 37.7% rely on gesture types rarely or never supported by existing mobile agent frameworks, such as double tap, long press, and pinch. The limitations are twofold: current agents face difficulty detecting *hidden interactions* in general, and a considerable fraction remain entirely inaccessible due to unsupported gesture types.

Our experiments also highlight the importance of multimodal input. While recent GUI agents have increasingly adopted vision-centric approaches that rely solely on screenshots [19, 72], our

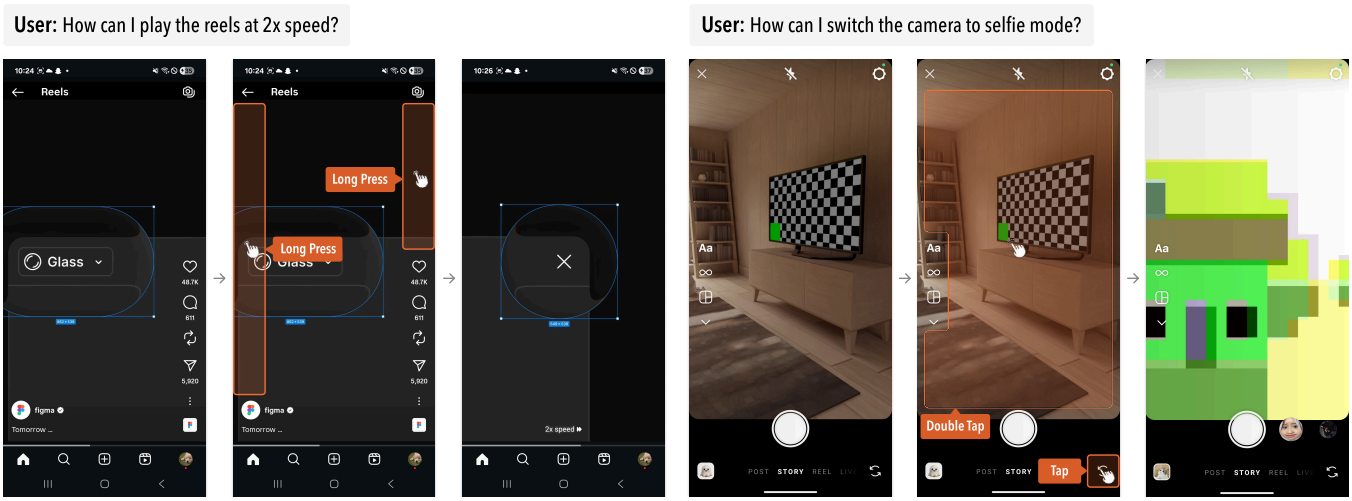


**Figure 9: Illustrations of Interactive Guidance for discovering *hidden interactions* based on user queries. (Left)** Long press gesture on specific regions of the video area to activate 2x playback speed. **(Right)** Double tap gesture on camera screen to switch to selfie mode, with alternative tap gesture shown for comparison.

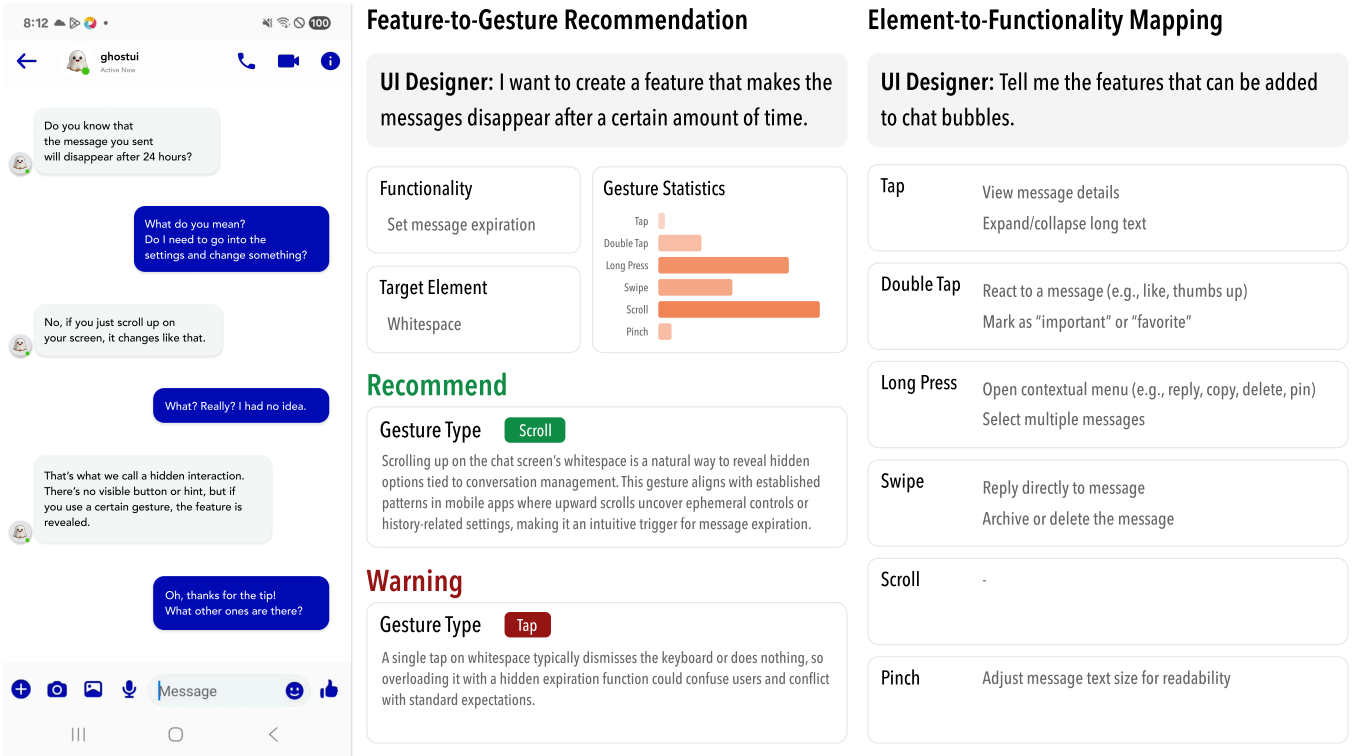


**Figure 10: Illustrations of Design Recommendations for *Hidden Interactions* in messaging apps. (Left: Feature-to-Gesture Recommendation)** The system analyzes a disappearing message feature and provides gesture suggestions based on empirical patterns, highlighting recommended options as well as potential pitfalls. **(Right: Element-to-Functionality Mapping)** The system presents mappings between gestures and their empirically observed functionalities in chat bubble interactions, such as viewing details, reactions, contextual menus, and quick replies or archiving.

findings reveal a fundamental limitation of such methods for *hidden interactions*. Removing simplified view hierarchy information caused IoU to drop dramatically, while classification accuracy remained relatively stable. This disparity stems from a mismatch between visual and interactive boundaries: VLMs tend to localize visually salient objects such as text or icons, whereas actual touch targets in mobile interfaces often encompass entire containers including surrounding whitespace and padding. For conventional UI elements like buttons, visual boundaries approximate interactive boundaries, enabling visual grounding approaches such as SeeClick [9] and OmniParser [43] to succeed. However, these methods are trained and evaluated exclusively on visible elements. *Hidden interactions* lack visual affordances by definition, making their interactive boundaries invisible to vision-only methods. Accurate localization of such interactions requires structural information that captures interactive boundaries independent of visual appearance, whether obtained directly from system-level view hierarchies or learned from specialized datasets like GhostUI that explicitly document these invisible targets.

Taken together, these results point to direct design implications for next-generation mobile agents. Expanding the action space ensures that agents can encompass the full range of gestures observed in practice, while multimodal input equips them to resolve ambiguities and localize targets more effectively. For practical deployment, GhostUI-trained models can augment existing agent architectures as specialized perception modules. The model should be invoked at three key decision points during agent execution: after initial screen analysis when no visible elements match the task intent, when action confidence scores fall below predefined thresholds, or when task descriptions explicitly reference gesture-based interactions (*e.g., "long press to select"*). Integration naturally occurs within the perception layer of current frameworks—positioned between screen parsing and action planning stages—where the model can be accessed as a tool through standardized protocols such as Model Context Protocol (MCP). These advances enable agents to cover workflows that are currently inaccessible, reduce multi-step reasoning chains into single-step predictions, and improve overall task success rates. GhostUI thus provides both empirical evidence of current limitations and a foundation for systematically developing and benchmarking more capable mobile agents.

## 6.2 Usability and Interaction Design Implications

Our analysis of gesture usage patterns reveals both diversity and redundancy in how gestures contribute to application functionality. We identified representative patterns across apps, yet many gestures also enable app-specific functions that are not fully captured by broader cross-app clusters. This highlights the dual character of gesture usage: while certain patterns recur across applications, others serve highly specialized roles, reflecting the variability of design practices in modern mobile ecosystems. The element-level labeling analysis further clarifies how different visual contexts relate to gesture activation. By examining label composition and co-occurrence, as well as gesture–element distributions by visibility context, we identified consistent associations between gestures and their target elements. For example, double taps are frequently tied to image or video regions and borders to visible and thus non-hidden interactions. These associations provide a systematic view of the affordance structures underlying mobile interfaces, offering empirical evidence of where discoverability breaks down.

These patterns raise an important question for end users: do *hidden interactions* serve as convenient shortcuts, or do they represent the sole pathway to functionality? To explore this, we extracted recurring intents from task descriptions, selected 200 interactions through stratified sampling across intent categories, and manually examined alternative pathways in a sample of interactions. Our analysis revealed significant variation across applications. In many cases, *hidden interactions* provided efficient shortcuts to functionality also accessible through visible UI elements—for instance, in Etsy, long pressing an item reveals options for reporting, adding to collections, and sharing, all of which are also available through buttons on the item's detail page. However, certain features remained exclusively hidden; emoji reactions in chat messages consistently lacked visible triggers across applications, unlike reactions in feeds which typically display like buttons. Most concerning were cases where alternatives themselves were *hidden interactions*—Band allows message replies through either long press or swipe, both lacking visual cues. Such redundant hidden pathways provide no benefit for discoverability. When *hidden interactions* lack visible alternatives, users must rely on prior experience, external documentation, or trial-and-error exploration to access features. This challenge is amplified for users with motor impairments or those relying on assistive technologies that do not readily support complex touch gestures. The prevalence of exclusively hidden functionality—and especially cases where even the alternatives remain hidden—highlights a systematic discoverability gap in contemporary mobile applications.

For designers and developers, GhostUI offers a data-driven foundation for determining when *hidden interactions* align with conventional usage patterns and when they diverge. By grounding design decisions in empirical patterns from widely-used applications, they can audit their applications for discoverability gaps, anticipate user expectations, and identify cases where core functionality should be made available through more accessible alternatives. In this way, GhostUI supports both improved user experience and principled design innovation, ensuring that the trade-off between interface minimalism and feature accessibility is approached in an evidence-based manner.

## 6.3 Improving Dataset Coverage and Scalability

Our collection strategy focused on key screens—primary destinations accessible through top-level navigation—to efficiently cover many applications. However, this approach inherently limited exploration of nested interfaces where additional *hidden interactions* may exist. The cold-start problem further constrained our coverage: in freshly installed apps, generating sufficient usage history required considerable time, delaying access to features that only emerge after regular use, such as chat history interactions or personalized recommendations triggered after viewing several videos. Moreover, manual annotation of *hidden interactions* remains time-consuming, as each collected instance requires careful human verification to ensure quality.

Looking forward, autonomous agents could address the cold-start problem by navigating through multiple screens and building up usage history. If agents could also handle tedious tasks such as login and sign-up, we could focus our time on covering more screens and discovering *hidden interactions* in deeper parts of applications. We experimented with mobile automation frameworks that enable LLMs to interact with mobile applications, though current performance remains insufficient for reliable autonomous exploration. For the annotation bottleneck, semi-automatic visual element labeling (*e.g.*, VLM-assisted pre-labeling with human confirmation) and lightweight crowdsourced validation could reduce verification overhead while maintaining reliability. As these technologies mature, combining agent-driven exploration with VLM-assisted annotation could transform GhostUI's collection from a semi-automated process with limited scope to an automated, comprehensive system that continuously discovers *hidden interactions* across the full depth of mobile applications.

## 6.4 Limitations and future work

*Platform and Device Coverage.* The current version of GhostUI focuses exclusively on Android mobile applications. While this scope enabled systematic data collection, it does not capture the broader diversity of platforms and interaction paradigms in contemporary computing. Future extensions should consider other operating systems, particularly iOS with its distinct gesture conventions, and additional device form factors. Wearable devices with constrained displays present particularly compelling targets for investigation, as their limited screen real estate likely necessitates even greater reliance on *hidden interactions* for accessing functionality.

*Interaction Type Coverage.* Beyond the six gestures currently documented, numerous complex interaction types remain unexplored. Drag-and-drop operations and multi-finger gestures represent additional interaction patterns that our dataset does not address. Furthermore, sensor-based interactions—such as shaking, tilting, or rotating the device—constitute another category of hidden functionality that merits future investigation. Expanding the action space to encompass these richer interaction types would enable more comprehensive automation capabilities and better reflect the full spectrum of user behaviors in real-world applications.

*Human Factors Research.* While our dataset quantifies the prevalence and distribution of *hidden interactions*, it does not address fundamental questions about human cognition and behavior. Understanding the discoverability and learnability of *hidden interactions* requires dedicated user studies that examine how people encounter, internalize, and recall these gestures over time. Human-centered research investigating these aspects could inform design guidelines that balance the space-efficiency benefits of *hidden interactions* with their inherent accessibility challenges, complementing our technical contributions with insights into user experience dimensions.

*Toward Task-Level Evaluation.* Our experiments primarily assess gesture-level prediction accuracy, which provides a meaningful foundation for understanding model capabilities on *hidden interactions*. However, two aspects of our evaluation warrant extension. First, our test set exclusively contains screens where *hidden interactions* exist, whereas realistic deployment requires agents to handle mixed conditions—including screens with visible affordances for the same tasks and screens where tasks are infeasible. Second, our accuracy metric evaluates exact gesture matching, yet different gestures can sometimes achieve equivalent functionality; for instance, both double tap and pinch can accomplish zooming, but predicting one when the other is the ground truth is currently marked as incorrect. Future work should address these limitations by developing mixed test sets and evaluation frameworks that account for functional equivalence, enabling more comprehensive assessment of how GhostUI-trained models perform in real-world agent pipelines.

## 7 Conclusion

In this paper, we presented GhostUI, the first dataset explicitly designed to document *hidden interactions* in mobile user interfaces. Through automated probing of 81 Android applications, we identified and validated 1,970 instances where essential functionality exists without visual affordances—revealing a critical blind spot in current UI understanding approaches. Our experiments demonstrate that training on GhostUI enables vision language models to predict both interaction gestures and resulting UI transitions. These findings highlight the importance of expanding both the action spaces and contextual understanding capabilities of mobile automation systems. By open-sourcing our collection framework alongside the dataset, we provide both the empirical foundation and methodological tools for the research community to address this overlooked dimension of mobile interaction. We hope GhostUI catalyzes further research at the intersection of UI understanding, mobile task automation, and human-computer interaction, ultimately paving the way for more intuitive and capable mobile experiences for all users.

## Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2023R1A2C200520911), the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) [NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], and by the SNU-Global Excellence Research Center establishment project. The ICT at Seoul National University provided research facilities for this study.

## References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615* (2024).

[3] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. 2021. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731* (2021).

[4] Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous

reinforcement learning. *Advances in Neural Information Processing Systems* 37 (2024), 12461–12495.

[5] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923* (2025).

[6] Sara Bunian, Kai Li, Chaima Jemmali, Casper Harteveld, Yun Fu, and Magy Seif Seif El-Nasr. 2021. Vins: Visual search for mobile user interface design. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[7] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. 2021. Mobile app tasks with iterative feedback (motif): Addressing task feasibility in interactive visual environments. *arXiv preprint arXiv:2104.08560* (2021).

[8] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, and Hongsheng Li. 2024. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490* (2024).

[9] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 9313–9332.

[10] Javier Cuello and José Vittone. 2013. *Designing mobile apps*. José Vittone.

[11] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*. 845–854.

[12] Google Material Design. 2024. Material Design 3. https://m3.material.io Accessed: 2025-04-09.

[13] Nicolai Dorka, Janusz Marecki, and Ammar Anwar. 2024. Training a vision language model as smartphone assistant. *arXiv preprint arXiv:2404.08755* (2024).

[14] Brad Frost. 2016. *Atomic design*. Brad Frost Pittsburgh.

[15] Longxi Gao, Li Zhang, Shihe Wang, Shangguang Wang, Yuanchun Li, and Mengwei Xu. 2024. Mobileviews: A large-scale mobile gui dataset. *arXiv preprint arXiv:2409.14337* (2024).

[16] William W Gaver. 1991. Technology affordances. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 79–84.

[17] Akash Ghosh, Arkadeep Acharya, Sriparna Saha, Vinija Jain, and Aman Chadha. 2024. Exploring the frontier of vision-language models: A survey of current methodologies and future directions. *arXiv preprint arXiv:2404.07214* (2024).

[18] James J Gibson. 2014. *The ecological approach to visual perception: classic edition*. Psychology press.

[19] 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).

[20] Zhangxuan Gu, Zhuoer Xu, Haoxing Chen, Jun Lan, Changhua Meng, and Weiqiang Wang. 2023. Mobile user interface element detection via adaptively prompt tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11155–11164.

[21] Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers, Gabriel Schubiner, Ruby Lee, and Jindong Chen. 2021. Actionbert: Leveraging user actions for semantic understanding of user interfaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 5931–5938.

[22] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14281–14290.

[23] Yu-Chung Hsiao, Fedir Zubach, Gilles Baechler, Victor Carbune, Jason Lin, Maria Wang, Srinivas Sunkara, Yun Zhu, and Jindong Chen. 2022. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199* (2022).

[24] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[25] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

[26] Apple Inc. 2024. Human Interface Guidelines. https://developer.apple.com/design/human-interface-guidelines Accessed: 2025-04-09.

[27] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. 2019. Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *Multimedia tools and applications* 78, 11 (2019), 15169–15211.

[28] Wenjia Jiang, Yangyang Zhuang, Chenxi Song, Xu Yang, and Chi Zhang. 2025. AppAgentX: Evolving GUI Agents as Proficient Smartphone Users. *arXiv preprint arXiv:2503.02268* (2025).

[29] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[30] Juyong Lee, Taywon Min, Minyong An, Dongyoon Hahm, Haeone Lee, Changyeon Kim, and Kimin Lee. 2024. Benchmarking Mobile Device Control Agents across Diverse Configurations. *arXiv preprint arXiv:2404.16660* (2024).

[31] Moon-Hwan Lee, Da-Hoon Kim, Hyun-Jeong Kim, and Tek-Jin Nam. 2012. Understanding impacts of hidden interfaces on mobile phone user experience. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*. 45–48.

[32] Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steven Y Ko, Sangeun Oh, and Insik Shin. 2023. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation. *arXiv preprint arXiv:2312.03003* (2023).

[33] Luis A Leiva, Asutosh Hota, and Antti Oulasvirta. 2020. Enrico: A dataset for topic modeling of mobile UI designs. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–4.

[34] Gang Li and Yang Li. 2022. Spotlight: Mobile ui understanding using vision-language models with a focus. *arXiv preprint arXiv:2209.14927* (2022).

[35] Tao Li, Gang Li, Jingjie Zheng, Purple Wang, and Yang Li. 2022. Mug: Interactive multimodal grounding on user interfaces. *arXiv preprint arXiv:2209.15099* (2022).

[36] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020. Mapping natural language instructions to mobile UI action sequences. *arXiv preprint arXiv:2005.03776* (2020).

[37] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. 2020. Widget captioning: Generating natural language description for mobile user interface elements. *arXiv preprint arXiv:2010.04295* (2020).

[38] Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey Gritsenko. 2021. Vut: Versatile ui transformer for multi-modal multi-task user interface modeling. *arXiv preprint arXiv:2112.05692* (2021).

[39] Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824* (2024).

[40] Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967* (2024).

[41] Yi-Chi Liao, Kashyap Todi, Aditya Acharya, Antti Keurulainen, Andrew Howes, and Antti Oulasvirta. 2022. Rediscovering affordance: A reinforcement learning perspective. In *Proceedings of the 2022 CHI conference on human factors in computing systems*. 1–15.

[42] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451* (2024).

[43] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203* (2024).

[44] Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. *arXiv preprint arXiv:2402.11941* (2024).

[45] Mubashar Munir and Pietro Murano. 2023. The Usability of Hidden Functional Elements in Mobile User Interfaces. (2023).

[46] Erik G Nilsson. 2009. Design patterns for user interface for mobile applications. *Advances in engineering software* 40, 12 (2009), 1318–1328.

[47] Donald A Norman. 1999. Affordance, conventions, and design. *interactions* 6, 3 (1999), 38–43.

[48] OpenAI. 2024. Fine-tuning Guide. https://platform.openai.com/docs/guides/fine-tuning Accessed: 2025-04-08.

[49] Seokhyeon Park, Wonjae Kim, Young-Ho Kim, and Jinwook Seo. 2023. Computational approaches for app-to-app retrieval and design consistency check. *arXiv preprint arXiv:2309.10328* (2023).

[50] Seokhyeon Park, Yumin Song, Soohyun Lee, Jaeyoung Kim, and Jinwook Seo. 2025. Leveraging Multimodal LLM for Inspirational User Interface Search. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 579, 22 pages. doi:10.1145/3706598.3714213

[51] Panupong Pasupat, Tian-Shun Jiang, Evan Zheran Liu, Kelvin Guu, and Percy Liang. 2018. Mapping natural language commands to web elements. *arXiv preprint arXiv:1808.09132* (2018).

[52] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573* (2024).

[53] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems* 36 (2023), 59708–59728.

[54] Eldon Schoop, Xin Zhou, Gang Li, Zhourong Chen, Bjoern Hartmann, and Yang Li. 2022. Predicting and explaining mobile ui tappability with vision modeling and saliency analysis. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–21.

[55] Maayan Shvo, Zhiming Hu, Rodrigo Toro Icarte, Iqbal Mohomed, Allan D Jepson, and Sheila A McIlraith. 2021. AppBuddy: Learning to Accomplish Tasks in Mobile Apps via Reinforcement Learning.. In *Canadian AI*.

[56] Yunpeng Song, Yiheng Bian, Yongtao Tang, Guiyu Ma, and Zhongmin Cai. 2024. Visiontasker: Mobile task automation using vision based ui understanding and llm task planning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–17.

[57] 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).

[58] Amanda Swearngin and Yang Li. 2019. Modeling mobile interface tappability using crowdsourcing and deep learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–11.

[59] Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. 2021. Androidenv: A reinforcement learning platform for android. *arXiv preprint arXiv:2105.13231* (2021).

[60] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.

[61] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile UI summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 498–510.

[62] Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. 2025. Mobile-Agent-E: Self-Evolving Mobile Assistant for Complex Tasks. *arXiv preprint arXiv:2501.11733* (2025).

[63] 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. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 543–557.

[64] Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. 2023. Droidbot-gpt: Gpt-powered ui automation for android. *arXiv preprint arXiv:2304.07061* (2023).

[65] Max Wertheimer. 1938. Gestalt theory. (1938).

[66] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* 1, 6 (1945), 80–83.

[67] Biao Wu, Yanda Li, Meng Fang, Zirui Song, Zhiwei Zhang, Yunchao Wei, and Ling Chen. 2024. Foundations and recent trends in multimodal mobile agents: A survey. *arXiv preprint arXiv:2411.02006* (2024).

[68] Jason Wu, Rebecca Krosnick, Eldon Schoop, Amanda Swearngin, Jeffrey P Bigham, and Jeffrey Nichols. 2023. Never-ending learning of user interfaces. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–13.

[69] Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024. Mobilevlm: A vision-language model for better intra-and inter-ui understanding. *arXiv preprint arXiv:2409.14818* (2024).

[70] Wenhao Wu, Huanjin Yao, Mengxi Zhang, Yuxin Song, Wanli Ouyang, and Jingdong Wang. 2023. GPT4Vis: what can GPT-4 do for zero-shot visual recognition? *arXiv preprint arXiv:2311.15732* (2023).

[71] Shuhong Xiao, Yunnong Chen, Yaxuan Song, Liuqing Chen, Lingyun Sun, Yankun Zhen, and Yanfang Chang. 2024. UI Semantic Group Detection: Grouping UI Elements with Similar Semantics in Mobile Graphical User Interface. *arXiv preprint arXiv:2403.04984* (2024).

[72] 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. *arXiv preprint arXiv:2412.04454* (2024).

[73] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2024. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *European Conference on Computer Vision*. Springer, 240–255.

[74] Danyang Zhang, Zhennan Shen, Rui Xie, Situo Zhang, Tianbao Xie, Zihan Zhao, Siyuan Chen, Lu Chen, Hongshen Xu, Ruisheng Cao, et al. 2023. Mobile-Env: Building Qualified Evaluation Benchmarks for LLM-GUI Interaction. *arXiv preprint arXiv:2305.08144* (2023).

[75] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. 2024. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[76] Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. 2024. Llamatouch: A faithful and scalable testbed for mobile ui task automation. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–13.

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

[78] Zhizheng Zhang, Xiaoyi Zhang, Wenxuan Xie, and Yan Lu. 2023. Responsible task automation: Empowering large language models as responsible task automators. *arXiv preprint arXiv:2306.01242* (2023).

# A UI Task Generation Prompt

You will receive:

1. A screenshot of a mobile app before the interaction, with the interacted UI component highlighted in orange (BEFORE)
2. A screenshot after the interaction (AFTER)
3. The type of gesture performed (e.g., tap, double_tap, long_press, swipe_left, swipe_right, scroll_up, scroll_down, pinch_zoom_in, pinch_zoom_out)

Your task is to:

- Carefully analyze the visual changes between BEFORE and AFTER screenshots.
- Use the orange-highlighted component in the BEFORE screenshot to identify what UI element the user interacted with.
- Based on the visual difference and the UI context, infer the user's intent and the effect of the interaction.

Important guidelines:

- Focus on WHY the user performed the action, not HOW (the gesture).
- Do not mention the gesture type (tap, double tap, long press, swipe left, swipe right, scroll up, scroll down, pinch zoom in, pinch zoom out) in your description.
- Describe tasks in terms of user goals (e.g., "Search for funny cat videos" instead of "Enter “funny cat” in search box").
- Write in imperative form as if it's a task instruction (e.g., "Watch a recommended video" not "User watched a video").
- Provide your response as a SINGLE SENTENCE.

Here is the BEFORE: ${before_screenshot}
Here is the AFTER: ${after_screenshot}
Here is the gesture type: ${gesture_type}

**Figure 11: Prompt Template for Task Generation.** GPT-4o is given before and after screenshots of a mobile app (with the interacted UI element highlighted in the before image) and the gesture type. The task is to generate a single sentence, imperative instruction that captures the user's intent and the effect of the interaction, without mentioning the gesture.

# B Experimental Prompts

## B.1 *Hidden Interaction* Prediction

```
You will receive:
    - A task description (what the user wants to accomplish)
    - A screenshot of a mobile app
    - A mobile application metadata (e.g., app name, category, description, etc.)
    - A simplified view hierarchy of the screen in HTML format
    - Gesture-specific usage patterns

Your task is to:
    - Carefully analyze the screenshot and the view hierarchy to identify all interactive UI elements.
    - Determine which specific UI element needs to be interacted with to complete the task.
    - Specify the precise gesture needed. Gesture must be one of:
      - tap, double_tap, long_press, swipe_left, swipe_right, scroll_up, scroll_down, pinch_zoom_in, pinch_zoom_out
    - Identify the bounding box of the UI element in the format [x1,y1][x2,y2] (use absolute pixel coordinates based on resolution).

Important Guidelines:
    - The gesture must be one of the following:
      - tap, double_tap, long_press, swipe_left, swipe_right, scroll_up, scroll_down, pinch_zoom_in, pinch_zoom_out
    - Respond strictly in the following JSON format:
      {
        "gesture": "<one of the gesture types above>",
        "bounds": "[x1,y1][x2,y2]"
      }

The task you need to help with is: ${task}
Here is the mobile screenshot: ${before_screenshot}
Here is the mobile app metadata: ${app_metadata}
Here is the simplified view hierarchy of the screen: ${simplified_view_hierarchy}
Here is the the gesture-specific usage patterns: ${gesture_usage_patterns}
```

**Figure 12: Prompt Template for Hidden Interaction Prediction.** Models are given a task description, a mobile app screenshot, app metadata, a simplified HTML view hierarchy, and gesture-specific usage patterns. The task is to identify the correct UI element, select an appropriate gesture from a predefined set, and return both the gesture and bounding box in a structured JSON format. An ablation study tests the effect of removing the view hierarchy, gesture patterns, or app metadata to measure each component's impact on performance.

## B.2 UI Transition Prediction

```
You will receive:
    - A screenshot of a mobile app's current state
    - A specific gesture that will be performed (tap, double_tap, long_press, swipe_left, swipe_right, scroll_up, scroll_down, pinch_zoom_in, pinch_zoom_out)
    - The bounding box coordinates of the UI element being interacted with, in format [x1,y1][x2,y2]

Your task is to:
    - Predict what happens after the specified gesture is performed on that element.
    - Use present tense for your entire description.

Important Guidelines:
    - Describe what is displayed on the new screen (2-3 sentences maximum).
    - Keep your response concise and focused.
    - Do not include any additional information or sections.

Here is the current state: ${before_screenshot}
Here is the gesture type: ${gesture_type}
Here is the bounding box: ${bounding_box}
```

**Figure 13: Prompt Template for UI Transition Prediction.** Models are given a screenshot of a mobile app's current state, a specific gesture to be performed, and the bounding box of the target UI element. The task is to predict what the screen will display after the gesture is executed on the specified element. The output is a concise 2–3 sentence description in present tense, focusing solely on the visual result.

```
You will receive:
    - A screenshot of a mobile app's current state.

Your task is to:
    - Describe the screen.
    - Use present tense for your entire description.

Important Guidelines:
    - Describe what is displayed on the screen (2-3 sentences maximum).
    - Keep your response concise and focused.
    - Do not include any additional information or sections.

Here is the mobile screenshot: ${after_screenshot}
```

**Figure 14: Prompt Template for AFTER Screenshot Description.** GPT-4o is given a screenshot of a mobile app's current state. The task is to concisely describe the visible content on the screen using 2–3 sentences in present tense, focusing only on what is shown without adding extra context.