GUI-explorer: Autonomous Exploration and Mining of Transition-aware Knowledge for GUI Agent

GUI-explorer: 面向 GUI 代理的自主探索与转移感知知识挖掘

Bin Xie [1] , Rui Shao [1†] , Gongwei Chen [1†] , Kaiwen Zhou [2] , Yinchuan Li2, Jie Liu', Min Zhang [1] , Liqiang Nie [1]

谢斌 [1]，邵锐 [1†]，陈公伟 [1†]，周凯文 [2]，李银川 2，刘杰'，张敏 [1]，聂立强 [1]

[1] Harbin Institute of Technology, Shenzhen, [2] Huawei Noah' s Ark Lab

[1] 哈尔滨工业大学深圳校区，[2] 华为诺亚方舟实验室

Abstract

摘要

GUI automation faces critical challenges in dynamic environments. MLLMs suffer from two key issues: misinterpreting UI components and outdated knowledge. Traditional fine-tuning methods are costly for app-specific knowledge updates. We propose GUI-explorer, a training-free GUI agent that incorporates two fundamental mechanisms: (1) Autonomous Exploration of Function-aware Trajectory. To comprehensively cover all application functionalities, we design a Function-aware Task Goal Generator that automatically constructs exploration goals by analyzing GUI structural information (e.g., screenshots and activity hierarchies). This enables systematic exploration to collect diverse trajectories. (2) Unsupervised Mining of Transition-aware Knowledge. To establish precise screen-operation logic, we develop a Transition-aware Knowledge Extractor that extracts effective screen-operation logic through unsupervised analysis the state transition of structured interaction triples (observation, action, outcome). This eliminates the need for human involvement in knowledge extraction. With a task success rate of 53.7% on SPA-Bench and 47.4% on AndroidWorld, GUI-explorer shows significant improvements over SOTA agents. It requires no parameter updates for new apps. GUI-explorer is open-sourced and publicly available at https: //github.com/JiuTian-VL/GUI-explorer.

图形用户界面 (GUI) 自动化在动态环境中面临关键挑战。多模态大语言模型 (MLLMs) 存在两个主要问题: 误解 UI 组件和知识过时。传统的微调方法在应用特定知识更新上成本高昂。我们提出了 GUI-explorer，一种无需训练的 GUI 代理，包含两个核心机制:(1) 功能感知轨迹的自主探索。为全面覆盖所有应用功能，我们设计了功能感知任务目标生成器，通过分析 GUI 结构信息 (如截图和活动层级) 自动构建探索目标，实现系统化探索以收集多样轨迹。(2) 转移感知知识的无监督挖掘。为建立精准的屏幕-操作逻辑，我们开发了转移感知知识提取器，通过无监督分析结构化交互三元组 (观察、动作、结果) 的状态转移，提取有效的屏幕-操作逻辑，免除人工参与知识提取。在 SPA-Bench 任务成功率达 53.7%，AndroidWorld 达 47.4%，GUI-explorer 较最先进代理表现显著提升，且无需针对新应用更新参数。GUI-explorer 已开源，公开地址:https://github.com/JiuTian-VL/GUI-explorer。

1 Introduction

1 引言

Automation in graphical user interfaces (GUIs) has rapidly advanced (Su et al., 2024). This progress is driven by foundational models like large language models (LLMs) (Touvron et al., 2023; Achiam et al., 2023; Yang et al., 2024a) and multimodal large language models (MLLMs) (Hurst et al., 2024; Chen et al., 2024; Shao et al., 2024; Google,

图形用户界面 (GUI) 自动化快速发展 (Su 等，2024)。这一进展得益于基础模型如大型语言模型 (LLMs)(Touvron 等, 2023；Achiam 等, 2023；Yang 等, 2024a) 和多模态大型语言模型 (MLLMs)(Hurst 等，2024；Chen 等，2024；Shao 等，2024；Google，

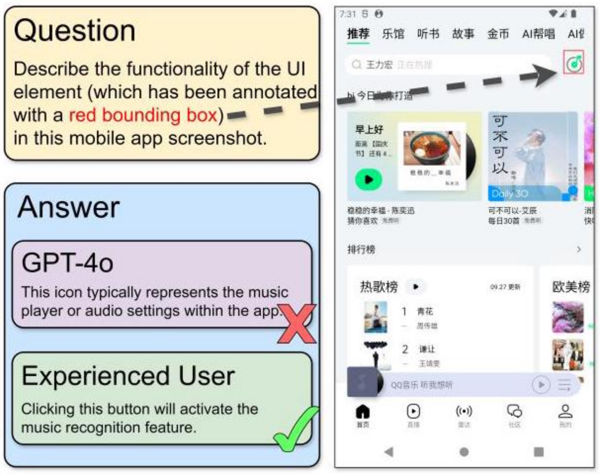

Question
Describe the functionality of the UI element (which has been annotated with a red bounding box) in this mobile app screenshot.
Answer
GPT-4o
This icon typically represents the music player or audio settings within the app.
Experienced User
Clicking this button will activate the music recognition feature.
推荐 乐馆 听书 故事 金币 AI帮唱
热歌榜
欧美榜

Figure 1: Comparison of GPT-40 and an user's interpretation of a UI element in QQ Music [1]. The red-bounded icon in the screenshot represents the music recognition feature, but GPT-4o misidentified it. This highlights the challenge of accurately interpreting UI elements in an ecosystem of diverse apps with distinct designs.

图 1:GPT-40 与用户对 QQ 音乐中 UI 元素的解读对比 [1]。截图中红框标注的图标代表音乐识别功能，但 GPT-40 误判了该图标。这凸显了在设计各异的多样化应用生态中准确解读 UI 元素的挑战。

2025; Shao et al., 2023; Li et al., 2025a; Shen et al., 2024). These innovations enable agents (Zheng et al., 2024; Zhang et al., 2025; Wang et al., 2024a; Li et al., 2025b; Ye et al., 2024; Li et al., 2025c) to handle tasks. They require no extensive fine-tuning or pretraining. This demonstrates their potential for diverse applications.

2025；Shao 等，2023；Li 等，2025a；Shen 等，2024)。这些创新使代理 (Zheng 等，2024；Zhang 等，2025；Wang 等，2024a；Li 等，2025b；Ye 等，2024；Li 等，2025c) 能够处理任务，无需大量微调或预训练，展现出广泛应用潜力。

However, the practical deployment of these models faces significant challenges. These challenges stem from the long-tail distribution of app/website variants and their rapid iteration cycles. While core functionalities might appear similar across platforms, critical design divergences exist. For example: shopping cart features in Amazon.com and Temu [2] share similarities. In contrast, Pin-duoduo [3] (China' s dominant e-commerce platform) eliminates cart functionality entirely. This requires

然而，这些模型的实际部署面临重大挑战，源于应用/网站变体的长尾分布及其快速迭代周期。尽管核心功能在各平台间似乎相似，但关键设计计存在差异。例如: 亚马逊 (Amazon.com) 和 Temu [2] 的购物车功能相似，而中国主导电商平台拼多多 [3] 完全取消了购物车功能，要求

---

[1] https://play.google.com/store/apps/details? id=com.tencent.qqmusic

[1] https://play.google.com/store/apps/details?id=com.tencent.qqmusic

[2] https://www.temu.com

[2] https://www.temu.com

[3] https://mobile.pinduoduo.com

[3] https://mobile.pinduoduo.com

† Corresponding authors. shaorui@hit.edu.cn, chengong-wei@hit.edu.cn.

† 通讯作者。shaorui@hit.edu.cn， chengong-wei@hit.edu.cn。

---

single-item purchases rather than batch checkout. Such inconsistencies extend beyond functionality to interface semantics. As shown in Figure 1, even advanced MLLMs such as GPT-40 (Hurst et al., 2024) can misinterpret the button's actual functionality. Human users familiar with the app, however, correctly interpret it through learned interaction patterns. Compounding this challenge, apps/websites undergo frequent updates. Amazon Shopping alone released 30 version iterations in 2024[4] . This renders static model knowledge obsolete. Retraining or fine-tuning (M)LLMs for every change proves prohibitively expensive and latency-prone.

单件购买而非批量结算。这类不一致不仅体现在功能上，也体现在界面语义上。如图 1 所示，即使是先进的多模态大语言模型 GPT-40(Hurst 等，2024) 也可能误解按钮的实际功能，而熟悉该应用的人类用户则通过学习的交互模式正确理解。更复杂的是，应用/网站频繁更新，仅亚马逊购物一年内就发布了 30 个版本迭代 2024[4] ，这使得静态模型知识迅速过时。对每次变更进行 (M)LLMs 的再训练或微调成本高昂且延迟明显。

In this paper, we propose Autonomous Exploration and Mining of Transition-aware Knowledge for GUI Agent (GUI-explorer). It syn-ergizes two key components: (1) Autonomous Exploration of Function-aware Trajectory. To cover all potential functions of target applications, we design a Function-aware Task Goal Generator. This module automatically constructs function-aware exploration goals by analyzing structural information of the environment, including screen-shots and activity lists from APK files. Through systematic exploration, we obtain diverse function-aware trajectories. (2) Unsupervised Mining of Transition-aware Knowledge. To establish precise operation logic, we develop a Transition-aware Knowledge Extractor. This component extracts effective operation logic through unsupervised analysis of state transitions from structured interaction triples (observation, action, outcome). This eliminates human involvement. Through multimodal state modeling incorporating visual patterns and semantic patterns, the extractor captures operation constraints and outcome dependencies, generating transition-aware knowledge with explicit action-effect correlations. Finally, by performing visual-semantic retrieval between current screen visuals and the knowledge vector store to construct Dynamic Guidance, it achieves two goals: suppressing the misinterpretation of UI components, and ensuring action proposals align with actual UI states. This approach facilitates precise, goal-oriented prompt generation. These prompts guide the agent in effectively understanding and interacting with GUI elements.

本文提出了面向 GUI 代理的自主探索与转移感知知识挖掘方法 (GUI-explorer)。该方法融合了两个关键组件:(1) 功能感知轨迹的自主探索。为了覆盖目标应用的所有潜在功能，我们设计了功能感知任务目标生成器。该模块通过分析环境的结构信息，包括 APK 文件中的屏幕截图和活动列表，自动构建功能感知的探索目标。通过系统化探索，获得多样的功能感知轨迹。(2) 转移感知知识的无监督挖掘。为建立精确的操作逻辑，我们开发了转移感知知识提取器。该组件通过对结构化交互三元组 (观察、动作、结果) 中的状态转移进行无监督分析，提取有效的操作逻辑，消除了人工干预。通过融合视觉模式和语义模式的多模态状态建模，提取器捕捉操作约束和结果依赖，生成具有明确动作-效果关联的转移感知知识。最后，通过在当前屏幕视觉与知识向量库之间执行视觉-语义检索构建动态引导，实现了两个目标: 抑制对 UI 组件的误解，确保动作建议与实际 UI 状态一致。该方法促进了精确、目标导向的提示生成，指导代理有效理解和交互 GUI 元素。

Our main contributions are listed below:

我们的主要贡献如下:

- We propose GUI-explorer, a novel training-

- 我们提出了 GUI-explorer，一种新颖的训练方法-

free agent that integrates two mechanisms: (1) Autonomous exploration of function-aware trajectory through environment-specific structural priors. (2) Unsupervised mining of transition-aware knowledge that extracts atomic screen-operation logic from raw interaction traces.

集成两种机制的自由代理:(1) 通过环境特定的结构先验自主探索具备功能感知的轨迹。(2) 无监督挖掘具备转移感知的知识，从原始交互轨迹中提取原子级屏幕操作逻辑。

- We conducted comprehensive evaluations of GUI-explorer across AndroidWorld and SPA-Bench benchmarks, our agent achieves 47.4% and 53.7% task success rates respectively, outperforming SOTA methods by 2.6%Ĩ1.7% improvement. Through ablation studies, we verified that our framework's transition-aware knowledge integration approach reduces prior knowledge errors by 16.0%.

- 我们在 AndroidWorld 和 SPA-Bench 基准测试中对 GUI-explorer 进行了全面评估，代理分别实现了 47.4% 和 53.7% 的任务成功率，较最先进方法提升了 2.6% 至 11.7%。通过消融研究，我们验证了框架中的转移感知知识整合方法将先验知识错误降低了 16.0%。

- We introduce a benchmark evaluating MLLMs' GUI understanding through 500 curated samples across 43 applications. Results reveal critical limitations in current models (15.2%2̃2.8% prior knowledge inaccuracies).

- 我们引入了一个基准测试，通过 43 个应用中的 500 个精选样本评估多模态大语言模型 (MLLMs) 对图形用户界面 (GUI) 理解的能力。结果显示当前模型存在关键限制 (先验知识错误率为 15.2%2̃2.8%)。

2 Related Work

2 相关工作

GUI Agents Modern GUI agents leverage MLLMs to interpret interface states and execute actions. SeeAct (Zheng et al., 2024) pioneers GPT-4V (OpenAI, 2023) for web task automation through visual understanding and HTML-guided action grounding. MobileAgentV2 (Wang et al., 2024a) implements multi-agent collaboration with memory units to track task progress and interface focus. M3A (Rawles et al., 2024) integrates ReAct-style (Yao et al., 2023) reasoning with Set-of-Mark (SoM) (Yang et al., 2023) visual annotations for Android device control, demonstrating zero-shot generalization across applications.

GUI 代理现代 GUI 代理利用多模态大型语言模型 (MLLMs) 来解释界面状态并执行操作。SeeAct(Zheng 等，2024) 开创性地使用 GPT-4V(OpenAI，2023) 通过视觉理解和基于 HTML 的动作定位实现网页任务自动化。MobileAgentV2(Wang 等，2024a) 实现了带有记忆单元的多代理协作，以跟踪任务进展和界面焦点。M3A(Rawles 等，2024) 结合了 ReAct 风格 (Yao 等，2023) 的推理与 Set-of-Mark(SoM)(Yang 等，2023) 的视觉注释，用于安卓设备控制，展示了跨应用的零样本泛化能力。

Exploration & Knowledge-aware Agents Autonomous exploration mechanisms vary in supervision requirements. AppAgent (Zhang et al., 2025) requires manually designed exploration tasks for knowledge acquisition, while AutoDroid (Wen et al., 2024) and MobileGPT (Lee et al., 2024) generates random action sequences for environment interaction. DigiRL (Zhou et al., 2024) employs reinforcement learning with Gemini-based (Google, 2025) trajectory filtering to collect successful demonstrations as training data.

探索与知识感知代理自主探索机制在监督需求上存在差异。AppAgent(张等，2025) 需要手动设计探索任务以获取知识，而 AutoDroid(温等，2024) 和 MobileGPT(李等，2024) 则通过生成随机动作序列与环境交互。DigiRL(周等，2024) 采用基于 Gemini(谷歌，2025) 的轨迹过滤的强化学习，收集成功示范作为训练数据。

Knowledge utilization strategies focus on experience retention and retrieval. CAT (Feng

知识利用策略侧重于经验的保留与检索。CAT(Feng

---

[4] https://www.apkmirror.com/uploads/ ?appcategory=amazon-shopping

[4] https://www.apkmirror.com/uploads/ ?appcategory=amazon-shopping

---

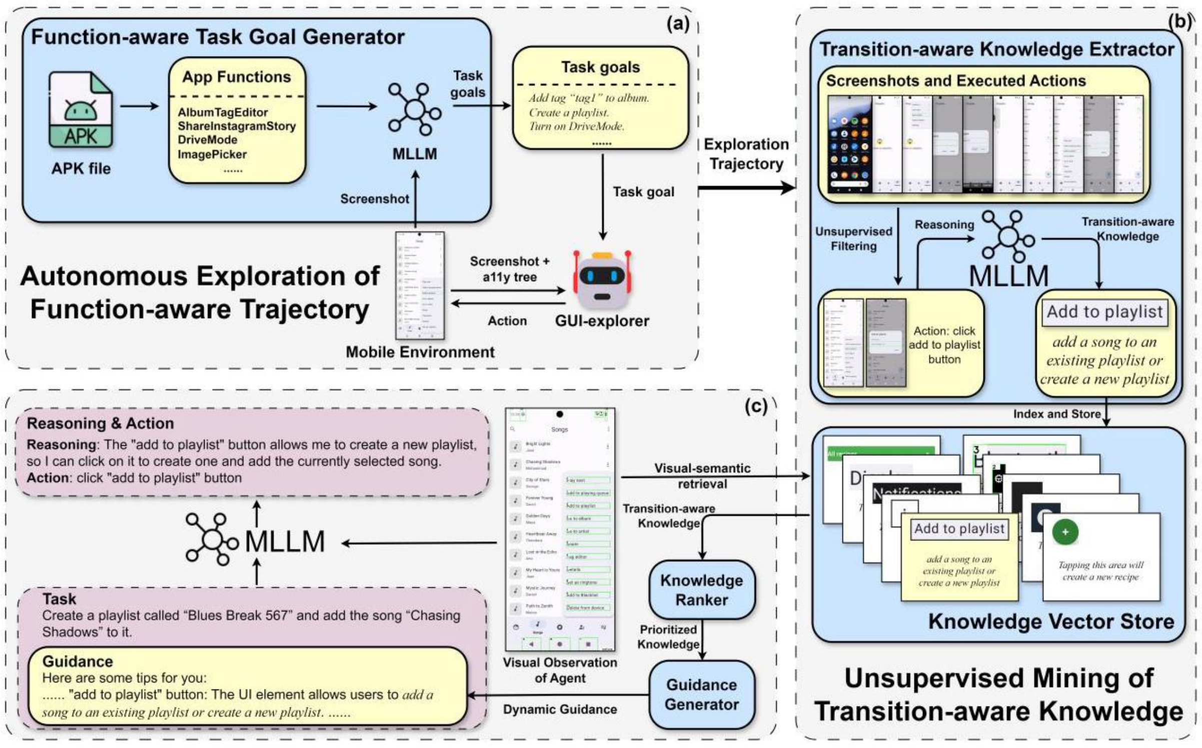

(a)
Function-aware Task Goal Generator
APK
APK file
App Functions
AlbumTagEditor
ShareInstagramStory
DriveMode
ImagePicker
......
MLLM
Task goals
Task goals
Add tag "tag1" to album.
Create a playlist.
Turn on DriveMode.
......
Screenshot
Task goal
Autonomous Exploration of Function-aware Trajectory
Screenshot + a11y tree
Action
GUI-explorer
Mobile Environment
Exploration Trajectory
(b)
Transition-aware Knowledge Extractor
Screenshots and Executed Actions
Unsupervised Filtering
Reasoning
Transition-aware Knowledge
MLLM
Action: click add to playlist button
Add to playlist
add a song to an existing playlist or create a new playlist
Index and Store
Knowledge Vector Store
Tapping this area will create a new recipe
Unsupervised Mining of Transition-aware Knowledge
(c)
Reasoning & Action
Reasoning: The "add to playlist" button allows me to create a new playlist, so I can click on it to create one and add the currently selected song.
Action: click "add to playlist" button
MLLM
Task
Create a playlist called "Blues Break 567" and add the song "Chasing Shadows" to it.
Guidance
Here are some tips for you:
...... "add to playlist" button: The UI element allows users to add a song to an existing playlist or create a new playlist. ......
Visual Observation of Agent
Visual-semantic retrieval
Transition-aware Knowledge
Knowledge Ranker
Prioritized Knowledge
Guidance Generator
Dynamic Guidance

Figure 2: Overview of GUI-explorer. (a) Automatically constructing function-aware exploration goals by analyzing structural information from the GUI environment, followed by systematic exploration to collect diverse function-aware trajectories. (b) Extracting effective screen-operation logic through unsupervised analysis of structured interaction triples (observation, action, outcome), enabling unsupervised knowledge extraction. (c) Performing visual-semantic retrieval between screen visuals and the knowledge vector store to construct Dynamic Guidance achieves dual objectives: preventing UI misinterpretation and ensuring action proposals align with actual UI states.

图 2:GUI-explorer 概述。(a) 通过分析 GUI 环境中的结构信息，自动构建功能感知的探索目标，随后进行系统性探索以收集多样的功能感知轨迹。(b) 通过对结构化交互三元组 (观察、操作、结果) 进行无监督分析，提取有效的屏幕操作逻辑，实现无监督知识提取。(c) 在屏幕视觉与知识向量库之间执行视觉语义检索，构建动态引导，实现双重目标: 防止界面误解并确保操作建议与实际界面状态一致。

et al., 2024) employs retrieval-augmented generation with task-specific successful trajectories, though limited to pre-collected demonstrations. Synapse (Zheng et al., 2023) introduces trajectory-as-exemplar prompting with state abstraction to improve cross-task generalization. ICAL (Sarch et al., 2024) abstracts interaction traces into transferable knowledge through visual-language model summarization and human feedback.

et al., 2024) 采用了基于检索增强生成的任务特定成功轨迹，尽管仅限于预先收集的示范。Synapse(Zheng et al., 2023) 引入了以轨迹为示例的提示方法，结合状态抽象以提升跨任务泛化能力。ICAL(Sarch et al., 2024) 通过视觉-语言模型总结和人类反馈，将交互轨迹抽象为可迁移的知识。

While existing methods demonstrate progress, four critical limitations persist: (1) Exploration efficiency suffers from random action generation or manual task design; (2) Knowledge extraction relies on successful trajectories or human curation, limiting scalability; (3) Static knowledge bases struggle with rapidly evolving interfaces; (4) Binding knowledge to specific element IDs restricts reuse to identical UIs.

尽管现有方法取得了一定进展，但仍存在四个关键限制:(1) 探索效率因随机动作生成或手动任务设计而受限；(2) 知识提取依赖于成功轨迹或人工整理，限制了可扩展性；(3) 静态知识库难以应对快速变化的界面；(4) 将知识绑定到特定元素 ID 限制了其在相同用户界面中的复用。

3 Autonomous Exploration and Mining of Transition-aware Knowledge for GUI Agent

3 面向过渡感知知识的 GUI 代理自主探索与挖掘

As illustrated in Figure 2, GUI-explorer consists of two main components: autonomous exploration

如图 2 所示，GUI-explorer 由两个主要组件组成: 功能感知轨迹的自主探索

of function-aware trajectory and unsupervised mining of transition-aware knowledge. Building upon the dual components mentioned, we employ visual-semantic retrieval during the agent's task execution to extract relevant knowledge based on the current observation. This retrieval mechanism enables a dynamic knowledge integration process that enhances the agent's decision-making capabilities. Specifically, we construct task-specific guidance by synthesizing the retrieved knowledge with both the current task goal and observational data. This guidance framework facilitates sophisticated reasoning processes, allowing the agent to make more informed decisions while navigating complex task environments.

和转移感知知识的无监督挖掘。基于上述双重组件，我们在代理执行任务时采用视觉-语义检索，根据当前观察提取相关知识。该检索机制实现了动态的知识整合过程，提升了代理的决策能力。具体而言，我们通过将检索到的知识与当前任务目标及观察数据合成，构建任务特定的指导框架。该指导框架促进了复杂的推理过程，使代理在复杂任务环境中做出更明智的决策。

3.1 Autonomous Exploration of Function-aware Trajectory

3.1 功能感知轨迹的自主探索

The core of our method lies in autonomously generating diverse interaction trajectories without human supervision. This exploration is grounded in environment-specific structural priors. These priors suppress misinterpretations derived from MLLMs' obsolete domain priors. Algorithm 1 formalizes

我们方法的核心在于无需人工监督，自主生成多样的交互轨迹。此探索基于环境特定的结构先验。这些先验抑制了多模态大语言模型 (MLLMs) 因过时领域先验导致的误判。算法 1 通过两个关键组件形式化了该过程。首先，锚点引导的任务生成利用界面语义。其次，深度优先探索结合状态恢复机制。

this process through two key components. First, anchor-guided task generation leverages interface semantics. Second, depth-first exploration incorporates state restoration mechanisms.

该过程通过两个关键组件形式化。首先，锚点引导的任务生成利用界面语义。其次，深度优先探索结合状态恢复机制。

Given a target environment E , we first extract Exploration Anchors. These are structural primitives derived from E ’ s ground-truth architecture, as detailed further in Appendix D. For mobile apps, functional modules declared in manifest files (e.g., ”PaymentActivity”). These anchors serve as verifiable constraints during task generation, preventing MLLMs from proposing actions targeting nonexistent components. The Task_Generator function constructs prompts (see Appendix I.1) containing current observation o_t and valid anchors, then samples up to k candidate tasks from MLLM outputs.

给定目标环境 E ，我们首先提取探索锚点。这些锚点是从 E 的真实架构中获得的结构原语，详见附录 D。对于移动应用，功能模块声明于清单文件中 (例如“PaymentActivity”)。这些锚点作为任务生成中的可验证约束，防止 MLLM 提出针对不存在组件的操作。Task_Generator 函数构建包含当前观察 o_t 和有效锚点的提示 (见附录 I.1)，然后从 MLLM 输出中采样最多 k 个候选任务。

The exploration follows depth-first search (DFS) with configurable branching factor b and depth d . This strategy eliminates the first state restoration overhead when expanding child tasks. The elimination occurs because each branch naturally inherits the terminal state of its parent task. This differs from breadth-first search (BFS), which requires resetting to the parent state for each sibling task expansion. Starting from initial state state e_0 , each generated task initiates an exploration branch. After executing a task for up to s steps via Task_Executor, the environment rolls back to previous state state $_i$. This mechanism enables exhaustive traversal of interface pathways without manual reset. The executor terminates exploration branches under two conditions: when receiving an ”END” action, or when reaching maximum steps. This balances thoroughness with computational efficiency.

探索遵循深度优先搜索 (DFS)，具有可配置的分支因子 b 和深度 d 。该策略消除了扩展子任务时首次状态恢复的开销。原因在于每个分支自然继承其父任务的终止状态。这与广度优先搜索 (BFS) 不同，后者在扩展每个兄弟任务时需重置到父状态。以初始状态 e_0 为起点，每个生成的任务启动一个探索分支。通过 Task_Executor 执行任务最多 s 步后，环境回滚至先前状态 state $_i$ 。该机制实现了界面路径的穷尽遍历，无需手动重置。执行器在两种情况下终止探索分支: 收到“END”动作或达到最大步数。这在全面性与计算效率间取得平衡。

This design achieves two critical properties: (1) Semantic Grounding: Anchors tether generated tasks to actual interface functions. (2) Quadratic Coverage: Each d -depth exploration with branching factor b yields $O\left(b^d\right)$ distinct trajectories, systematically capturing combinatorial interaction patterns.

该设计实现了两个关键特性:(1) 语义锚定: 锚点将生成任务绑定到实际界面功能。(2) 二次覆盖: 每个深度为 d 、分支因子为 b 的探索产生 $O\left(b^d\right)$ 条不同轨迹，系统性捕获组合交互模式。

3.2 Unsupervised Mining of Transition-aware Knowledge

3.2 转移感知知识的无监督挖掘

The knowledge construction process focuses on mining atomic screen-operation logic. These logic are derived from exploration trajectories. Let $\xi = \langle o_1, a_1, \ldots, o_n, a_n \rangle$ denote an interaction trajectory. This trajectory is collected during autonomous exploration. We extract transition-aware GUI knowledge through a Transition-aware Knowledge Ex-

知识构建过程聚焦于挖掘原子级屏幕操作逻辑。这些逻辑源自探索轨迹。设 $\xi = \langle o_1, a_1, \ldots, o_n, a_n \rangle$ 为一条交互轨迹，该轨迹在自主探索过程中收集。我们通过转移感知知识提取算法抽取转移感知的 GUI 知识——

Algorithm 1: Autonomous Exploration of Function-aware Trajectory

算法 1: 功能感知轨迹的自主探索

```
Input: Environment $E$ , max_branching_factor $b$ ,
输入: 环境 $E$， 最大分支因子 $b$ ,
       max_depth $d$ , max_steps $s$
       最大深度 $d$， 最大步数 $s$
Function Explore_DFS $(E, b, d,$ depth , task , $s)$
函数 Explore_DFS $(E, b, d,$ depth , task , $s)$
    Task_Executor $(E,$ task , $s)$ ;
    Task_Executor $(E,$ task , $s)$ ;
    if current_depth $> d$ then
    如果当前深度 $> d$ 则
        return;
        返回;
    current_state ← E.get_current_state();
    current_state ← E.get_current_state();
    child_tasks ← Task_Generator $(E, b)$ ;
    child_tasks ← Task_Generator $(E, b)$ ;
    for $i = 0$ to length(child_tasks) -1 do
    对于 $i = 0$ 到 length(child_tasks) -1 执行
```

```
            if $i > 0$ then

          如果 $i > 0$ 则

                E.restore_to(current_state);

              E.restore_to(current_state);

            Explore_DFS $(E, b, d,$ depth + )

          Explore_DFS $(E, b, d,$ depth + )

              1, child_tasks $[i]$ , $s)$ ;

            1, child_tasks $[i]$ , $s)$ ;

    Function Task_Generator $(E, k)$

  函数 Task_Generator $(E, k)$

        anchors ← E.app_functions;

      anchors ← E.app_functions;

        p ← ConstructPrompt(E.observation, anchors);

      p ← ConstructPrompt(E.observation, anchors);

        return MLLM(p).sample_top_k(k);

      返回 MLLM(p).sample_top_k(k);

    Function Task_Executor $(E,$ task , $s)$

  函数 Task_Executor $(E,$ task , $s)$

        for round $= l$ to $s$ do

      对于 round $= l$ 到 $s$ 执行

            action ← MLLM(task, E.observation);

          action ← MLLM(task, E.observation);

            /* We store the observation and

          /* 我们存储观察结果和
```

action for knowledge vector

知识向量的动作

store construction */

存储构建 */

if action == ”END” then

如果动作 == 是”END” 则

return;

返回;

E.step(action);

E.step(action);

initial_state ← E.get_current_state();

initial_state ← E.get_current_state();

tasks ← Task_Generator (E, b) ;

tasks ← Task_Generator (E, b) ;

foreach task in tasks do

遍历 tasks 中的每个任务

Explore_DFS $(E, b, d, 0,$ task $, s)$;

Explore_DFS $(E, b, d, 0,$ task $, s)$;

E.restore_to(initial_state);

E.restore_to(initial_state);

---

tractor function F_{extract} . This function operates on state-action transitions:

tractor 函数 F_{extract} 。该函数作用于状态-动作转移:

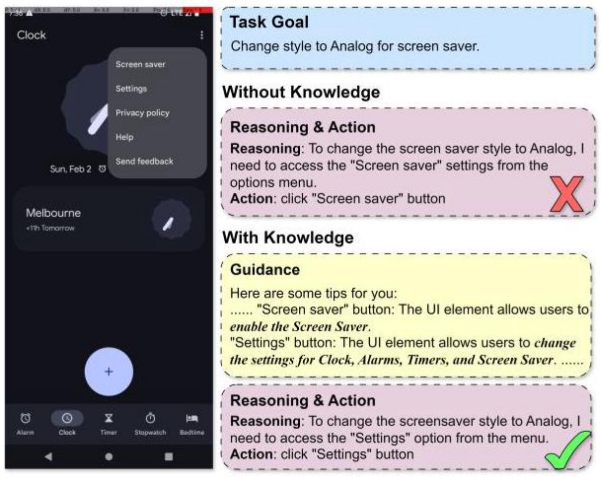
$$F_{\text{extract}} : (o_i, a_i, o_{i+1}) \rightarrow \{k_i : v_i\} \tag{1}$$

where o_i and o_{i+1} represent consecutive observations, a_i denotes the action executed, and $\{k_i : v_i\}$ outputs a set of visual-semantic knowledge entries. Each entry consists of: (1) k_i : visual patch of the interacted UI element,(2) v_i : operational knowledge (e.g., "Clicking this button opens search history").

其中 o_i 和 o_{i+1} 表示连续的观察，a_i 表示执行的动作，$\{k_i : v_i\}$ 输出一组视觉-语义知识条目。每个条目包括:(1) k_i : 交互的 UI 元素的视觉补丁，(2) v_i : 操作知识 (例如，“点击此按钮打开搜索历史”)。

Unlike previous work (Zheng et al., 2023; Feng et al., 2024; Sarch et al., 2024; Qin et al., 2025), which requires successful trajectories for in-context learning or fine-tuning, our approach has different requirements. Specifically, we only need valid state transitions. Therefore, we implement a filtering mechanism termed Transition Filtering to filter out invalid state transitions: Discard transitions where $o_i \approx o_{i+1}$. This similarity is measured via perceptual hashing (Marr and Hildreth, 1980). Such transitions indicate ineffective actions. These occur in two scenarios: when a_i fails to alter the environment (invalid action) or when the environment fails to respond (execution error).

与之前的工作 (Zheng 等，2023；Feng 等，2024；Sarch 等，2024；Qin 等，2025) 需要成功轨迹用于上下文学习或微调不同，我们的方法有不同的要求。具体来说，我们只需要有效的状态转移。因此，我们实现了一种称为转移过滤 (Transition Filtering) 的过滤机制，用于剔除无效的状态转移: 丢弃 $o_i \approx o_{i+1}$ 的转移。该相似度通过感知哈希 (Marr 和 Hildreth，1980) 测量。这类转移表示无效动作。出现于两种情况: 当 a_i 未能改变环境 (无效动作) 或环境未响应 (执行错误)。



Task Goal
Change style to Analog for screen saver.

Without Knowledge

Reasoning & Action
Reasoning: To change the screen saver style to Analog, I need to access the "Screen saver" settings from the options menu.
Action: click "Screen saver" button

With Knowledge

Guidance
Here are some tips for you:
...... "Screen saver" button: The UI element allows users to ***enable the Screen Saver***.
"Settings" button: The UI element allows users to ***change the settings for Clock, Alarms, Timers, and Screen Saver***. ......

Reasoning & Action
Reasoning: To change the screensaver style to Analog, I need to access the "Settings" option from the menu.
Action: click "Settings" button

Figure 3: Without transition-aware knowledge as reliable prior information, MLLMs may fail to reason correctly due to outdated prior knowledge or diverse GUI designs.

图 3: 缺乏转移感知知识作为可靠的先验信息，MLLMs 可能因先验知识过时或多样的 GUI 设计而无法正确推理。

The knowledge vector store $\mathcal{K}$ is structured as a multi-modal index:

知识向量存储 $\mathcal{K}$ 结构为多模态索引:

$$\mathcal{K} = \bigcup_{\xi \in \Xi} \bigcup_{t=1}^{|\xi|-1} F_{\text{extract}} \left(o_t, a_t, o_{t+1} \right) \tag{2}$$

where Ξ denotes all exploration trajectories and $|\xi|$ denotes the total steps of the trajectory ξ .

其中 Ξ 表示所有探索轨迹，$|\xi|$ 表示轨迹 ξ 的总步数。

This knowledge construction process enables Continuous Knowledge Refinement. New explorations iteratively update $\mathcal{K}$ through:

该知识构建过程实现了持续的知识精炼。新的探索通过以下方式迭代更新 $\mathcal{K}$:

$$\mathcal{K}=\begin{cases}\mathcal{K}\setminus\{(k_{\text{old}},v_{\text{old}})\}\cup\{(k_{\text{old}},v_{\text{old}}\oplus v_{\text{new}})\} & \text{if }\Phi\\ \mathcal{K}\cup\{(k_{\text{new}},v_{\text{new}})\} & \text{otherwise}\end{cases}\tag{3}$$

where $\mathcal{K}\setminus\{(k_{old},v_{old})\}$ denotes the removal of the original key-value pair from the knowledge vector store, $\oplus$ represents the concatenation of knowledge, condition Φ is formally defined as:

其中 $\mathcal{K}\setminus\{(k_{old},v_{old})\}$ 表示从知识向量存储中移除原始键值对，$\oplus$ 代表知识的拼接，条件 Φ 形式定义为:

$$\begin{aligned}&\exists(k_{\text{old}},v_{\text{old}})\in\mathcal{K}\\ &\text{s.t.}\begin{cases}\cos(Emb(k_{\text{new}}),Emb(k_{\text{old}}))\geq\delta_k\\ \cos(Emb(v_{\text{new}}),Emb(v_{\text{old}}))\leq\delta_v\end{cases}\end{aligned}\tag{4}$$

where δ_k and δ_v are similarity thresholds for key matching (≥ 0.99) and value merging (≤ 0.1) respectively, $\cos(\cdot)$ is cosine similarity, and $\text{Emb}(\cdot)$ is the embedding function. This prevents redundant entries while capturing novel interface behaviors.

其中 δ_k 和 δ_v 分别是键匹配 (≥ 0.99) 和值合并 (≤ 0.1) 的相似度阈值，$\cos(\cdot)$ 为余弦相似度，$\text{Emb}(\cdot)$ 为嵌入函数。这防止了冗余条目，同时捕捉了新颖的界面行为。

Figure 3 demonstrates the importance of transition-aware knowledge.

图 3 展示了对转移感知知识的重要性。

3.3 Dynamic Guidance for GUI Agent

3.3 GUI 代理的动态指导

The dynamic guidance mechanism connects acquired Transition-aware Knowledge to real-time

动态指导机制将获取的转移感知知识与实时环境连接起来

Algorithm 2: Dynamic Guidance for GUI Agent

算法 2:GUI 代理的动态指导

```
Input: Environment $E$ , Instruction $I$ ,
输入: 环境 $E$， 指令 $I$ ,
            Knowledge_Vector_Store $\mathcal{K}$ ,
            知识向量存储 $\mathcal{K}$ ,
            Knowledge_Ranker Ranker, max_steps $s$
            知识排序器 Ranker， 最大步骤数 $s$
Function Get_Guidance(obs, I, K)
函数 Get_Guidance(obs, I, K)
        annot_scr ← Get_Annotated_Screenshot(obs);
        annot_scr ← 获取带注释的截图 (obs);
        ui_elements ← Extract_UI_Elements(obs);
        ui_elements ← 提取 UI 元素 (obs);
        all_knol ← $\varnothing$ ; // all_knowledge
        all_knol ← $\varnothing$ ; // 所有知识
        foreach ui_element in ui_elements do
        遍历每个 ui_element 于 ui_elements 中执行
            all_knol.append(Retrieve_Knowledge( $\mathcal{K}$ ,
            all_knol.append(检索知识 ( $\mathcal{K}$ ,
                ui_element));
                ui_element));
        prioritized_knol ← Ranker(I, all_knol);
        prioritized_knol ← 排序器 (I, all_knol);
        guidance ← Create_Guidance_Prompt(I,
        指导 ← Create_Guidance_Prompt(I,
        rioritized_knol, annot_scr);
```

```
        rioritized_knol, annot_scr);
```

```
        return guidance;
```

```
        返回指导;
```

```
    for $idx = 1$ to $s$ do
```

```
    for $idx = 1$ 到 $s$ 执行
```

```
        obs ← E.observation;
```

```
        obs ← E.observation;
```

```
        operational_guid ← Get_Guidance(obs, $I, \mathcal{K}$ );
```

```
        operational_guid ← Get_Guidance(obs, $I, \mathcal{K}$ );
```

```
        action ← MLLM(I, operational_guid, obs);
```

```
        action ← MLLM(I, operational_guid, obs);
```

```
        if action == ”END” then
```

```
        if action == ”END” 则
```

```
            break;
```

```
            break;
```

```
        E.step(action);
```

```
        E.step(action);
```

---

task execution. This connection is achieved through a ranking architecture. As detailed in Algorithm 2, our approach uses a two-phase process. The first phase involves visual-semantic knowledge retrieval. The second phase performs instruction-aware prioritization.

任务执行。该连接通过排序架构实现。如算法 2 所述，我们的方法采用两阶段流程。第一阶段涉及视觉-语义知识检索。第二阶段执行指令感知的优先级排序。

Knowledge Ranking Formulation Given an instruction I and candidate knowledge entries $\mathcal{C} = \{k_1, \ldots, k_n\}$, we define the optimal knowledge ordering $\mathcal{C}^*$ through pairwise utility comparison:

知识排序公式: 给定指令 I 和候选知识条目 $\mathcal{C} = \{k_1, \ldots, k_n\}$ ，我们通过成对效用比较定义最优知识排序 $\mathcal{C}^*$:

$$\mathcal{C}^* = \underset{\pi \in \Pi(\mathcal{C})}{\arg\max} \sum_{i=1}^{|\mathcal{C}|-1} \operatorname{int}\left(u\left(k_{\pi(i)}, I\right) \geq u\left(k_{\pi(i+1)}, I\right)\right) \tag{5}$$

where $\Pi(\mathcal{C})$ denotes all permutations of $\mathcal{C}$, $\operatorname{int}(\cdot)$ converts bool to integer (false as 0, true as 1), and utility function $u(k, I)$ measures the relevance between knowledge entry k and instruction I . We implement $u(\cdot)$ through an MLLM-based pairwise comparator:

其中 $\Pi(\mathcal{C})$ 表示 $\mathcal{C}$ 的所有排列，int $(\cdot)$ 将布尔值转换为整数 (false 为 0，true 为 1)，效用函数 $u(k, I)$ 衡量知识条目 k 与指令 I 之间的相关性。我们通过基于多模态大语言模型 (MLLM) 的成对比较器实现 $u(\cdot)$:

$$u(k_a, I) > u(k_b, I) \Leftrightarrow f_{\text{rank}}\left(g(I, k_a, k_b)\right) = 1 \tag{6}$$

where $g(\cdot)$ constructs the ranking prompt (see Appendix I.3), and f_{rank} represents the MLLM’s binary classification. When the classification result is 1, it indicates k_a is more helpful than k_b for this instruction. When the result is 2, it means k_b is more helpful than k_a . This formulation enables efficient sorting through a modified merge sort algorithm:

其中 $g(\cdot)$ 构建排序提示 (见附录 I.3)，f_{rank} 表示 MLLM 的二分类。当分类结果为 1 时，表示 k_a 对该指令比 k_b 更有帮助；当结果为 2 时，表示 k_b 比 k_a 更有帮助。该公式通过改进的归并排序算法实现高效排序:

$$\operatorname{Sort}(\mathcal{C}, I) = \begin{cases} \mathcal{C} & |\mathcal{C}| \leq 1 \\ \operatorname{Merge}\left(\operatorname{Sort}(\mathcal{C}_L, I), \operatorname{Sort}(\mathcal{C}_R, I), I\right) & \text{otherwise} \end{cases} \tag{7}$$

The merge operation recursively compares head elements from sorted sublists using f_{rank} :

归并操作递归比较已排序子列表的首元素，使用 f_{rank} :

$$\operatorname{Merge}(A, B, I) = \begin{cases} [a_0] \oplus \operatorname{Merge}(A_{1:}, B, I) & f_{\text{rank}}\left(g(I, a_0, b_0)\right) = 1 \\ A \oplus B & A = \varnothing \vee B = \varnothing \\ [b_0] \oplus \operatorname{Merge}(A, B_{1:}, I) & \text{otherwise} \end{cases}$$

(8)

where a_0 and b_0 denote the first elements of lists A and B respectively.

其中 a_0 和 b_0 分别表示列表 A 和 B 的第一个元素。

Operational Guidance Generation At each execution step t , the system: (1) Extracts UI elements $\mathcal{U}_t$ from current observation o_t ; (2) Retrieves associated knowledge entries $\mathcal{K}_t \subseteq \mathcal{K}$; (3) Sorts entries via $\mathcal{K}_t^* = \operatorname{Sort}(\mathcal{K}_t, I)$; (4) Constructs guidance prompt p_t with relevant knowledge.

操作指导生成在每个执行步骤 t ，系统:(1) 从当前观察中提取 UI 元素 $\mathcal{U}_t$ ；(2) 检索相关知识条目 $\mathcal{K}_t \subseteq \mathcal{K}$ ；(3) 通过 $\mathcal{K}_t^* = \text{Sort}(\mathcal{K}_t, I)$ 对条目进行排序；(4) 利用相关知识构建指导提示 p_t 。

As shown in Figure 2 (c), the dynamic guidance mechanism enables precise alignment between operational knowledge and real-time interface states.

如图 2(c) 所示，动态指导机制实现了操作知识与实时界面状态的精确对齐。

4 GUI-Knowledge Reasoning Benchmark

4 GUI-知识推理基准

We introduce the GUI-Knowledge Reasoning Benchmark (GUI-KRB). This benchmark evaluates MLLMs' accuracy in two areas: prior knowledge accuracy and dynamic UI comprehension for mobile environments. Existing benchmarks primarily focus on task completion. In contrast, GUI-KRB assesses models' fundamental understanding of UI elements and their behaviors. It contains 500 carefully curated samples spanning 43 applications across 8 categories. Appendix B shows the proportion of apps in each category.

我们引入了 GUI-知识推理基准 (GUI-KRB)。该基准评估多模态大语言模型 (MLLMs) 在两个方面的准确性: 先验知识准确性和移动环境下的动态 UI 理解。现有基准主要关注任务完成情况，而 GUI-KRB 评估模型对 UI 元素及其行为的基础理解。它包含 500 个精心挑选的样本，涵盖 8 个类别中的 43 个应用。附录 B 展示了各类别应用的比例。

4.1 Tasks and Metrics

4.1 任务与指标

GUI-KRB includes two evaluation tasks: (1) Prior Knowledge Assessment: Models must identify the functionality of specified UI elements. They are given a single screenshot, its accessibility tree, and a task context about this element. This task simulates the planning phase in GUI automation. During planning, agents must understand element functionality before acting. Success here indicates effective use of prior training knowledge. (2) Dynamic Comprehension Assessment: Models analyze UI element functionality by comparing pre-interaction and post-interaction states within the task context of this transition. These states include screenshots and accessibility trees. This task evaluates reasoning about cause-effect logic in GUI interactions. It simulates the knowledge extraction method we use in this paper.

GUI-KRB 包含两个评估任务:(1) 先验知识评估: 模型需识别指定 UI 元素的功能。提供单张截图、其无障碍树及该元素的任务上下文。此任务模拟 GUI 自动化中的规划阶段，代理需在行动前理解元素功能。成功表明有效利用了先前训练知识。(2) 动态理解评估: 模型通过比较交互前后状态 (包括截图和无障碍树) 分析 UI 元素功能，结合该转换的任务上下文。此任务评估对 GUI 交互中因果逻辑的推理，模拟本文所用的知识提取方法。

For both tasks, responses are evaluated against human-annotated keywords. A response is consid-

两个任务中，回答均与人工标注关键词进行比对。回答被视为

ered correct if it contains at least 50% of expert-identified keywords. This metric balances precision with flexibility for valid phrasings. (During keyword labeling, we include up to 50% synonyms to accommodate diverse responses.)

正确，若包含至少 50% 的专家识别关键词。该指标在精确性与有效表达的灵活性间取得平衡。(关键词标注时包含最多 50% 的同义词以适应多样回答。)

4.2 Annotation Process

4.2 标注流程

GUI-KRB was built through a rigorous multi-stage process: (1) Trajectory Collection: We collected over 300 task execution trajectories in a mobile environment. These trajectories contain more than 7,000 interaction steps across diverse mobile applications. They capture authentic user interactions in real-world scenarios. (2) Element Extraction: From these trajectories, we extracted individual UI elements using bounding box information from accessibility trees. To ensure diversity and remove redundancy, we eliminated duplicate elements using perceptual hashing techniques (Marr and Hildreth, 1980). (3) Keyword Annotation: Human experts identified essential keywords uniquely associated with each UI element's functionality. These keywords capture both the element's immediate purpose and its broader role in the interface. (4) Validation: The authors conducted a comprehensive review of all annotations, verifying keyword accuracy and ensuring consistent annotation quality across the dataset.

GUI-KRB 通过严格的多阶段流程构建:(1) 轨迹收集: 收集了 300 多条移动环境下的任务执行轨迹，涵盖 7000 多个交互步骤，涉及多样移动应用，反映真实用户交互场景。(2) 元素提取: 从轨迹中利用无障碍树的边界框信息提取单个 UI 元素。为保证多样性并去除冗余，采用感知哈希技术 (Marr 和 Hildreth，1980) 剔除重复元素。(3) 关键词标注: 人工专家识别与每个 UI 元素功能唯一相关的关键字，涵盖元素的直接用途及其在界面中的更广泛角色。(4) 验证: 作者对所有标注进行了全面审查，核实关键词准确性，确保数据集标注质量一致。

The final dataset provides triplets of target UI elements, their corresponding screen states (before and after interaction), and expert-validated keyword sets. Example annotations are provided in Appendix C.

最终数据集提供目标 UI 元素、其对应的屏幕状态 (交互前后) 及专家验证的关键词集三元组。示例标注见附录 C。

5 Experiments

5 实验

5.1 Experimental Setup

5.1 实验设置

5.1.1 Datasets

5.1.1 数据集

We evaluate GUI-explorer on two comprehensive, open-source benchmarks: MIT-licensed SPA-Bench (Chen et al., 2025) and Apache-2.0-licensed AndroidWorld (Rawles et al., 2024). Both benchmarks emphasize real-world GUI and provide automated evaluation pipelines for rigorous agent assessment.

我们在两个全面的开源基准上评估 GUI-explorer:MIT 许可的 SPA-Bench(Chen 等，2025) 和 Apache-2.0 许可的 AndroidWorld(Rawles 等，2024)。两者均强调真实 GUI 场景，提供自动化评估流程以严谨评测代理。

SPA-Bench SPA-Bench is a benchmark simulating daily smartphone usage scenarios with 58 mainstream apps (e.g., Facebook and Gmail). It contains three progressively challenging task levels (Level 1-3), where Level 3 represents the most complex real-world workflows.

SPA-Bench 是一个模拟日常智能手机使用场景的基准，包含 58 个主流应用 (如 Facebook 和 Gmail)。它设有三个逐级递增难度的任务等级 (1-3 级)，其中 3 级代表最复杂的真实工作流。

AndroidWorld AndroidWorld is an Android environment featuring 116 tasks across 20 real-world apps. The benchmark dynamically generates task variants through randomized parameters (e.g., message content, contact names, and calendar dates), creating millions of unique task instantiations.

AndroidWorld 是一个包含 20 个真实应用中 116 个任务的安卓环境。该基准通过随机参数 (如消息内容、联系人姓名和日历日期) 动态生成任务变体，创造出数百万个独特的任务实例。

5.1.2 Implementation Details

5.1.2 实现细节

To ensure fair evaluation across benchmarks, we carefully selected base models according to their characteristics. For SPA-Bench and AndroidWorld, we adopted GPT-4o (Hurst et al., 2024) as the unified base model, which has been the de facto standard model in prior works including, but not limited to, SPA-Bench (Chen et al., 2025) and Aria-UI (Yang et al., 2024b), eliminating performance variance caused by heterogeneous model capabilities. In contrast, for GUI-KRB, we intentionally utilized the weakest-performing Qwen2-VL-72B-Instruct-GPTQ-Int4 (Wang et al., 2024b) as our base model, to rigorously validate the robustness of our method.

为了确保跨基准的公平评估，我们根据模型特性精心选择了基础模型。对于 SPA-Bench 和 AndroidWorld，我们采用了 GPT-40(Hurst 等，2024) 作为统一基础模型，该模型已成为包括但不限于 SPA-Bench(Chen 等，2025) 和 Aria-UI(Yang 等，2024b) 等先前工作的事实标准模型，消除了异构模型能力带来的性能差异。相比之下，对于 GUI-KRB，我们有意使用表现最弱的 Qwen2-VL-72B-Instruct-GPTQ-Int4(Wang 等，2024b) 作为基础模型，以严格验证我们方法的鲁棒性。

We configured the exploration process with a branching factor of 10, a maximum depth of 5, and a step limit of 30 for AndroidWorld and SPA-Bench. This setup facilitated the automated discovery of over 1,300 knowledge items (detailed distribution in Appendix H) across 46 applications. For visual-semantic retrieval, we utilized google/siglip-so400m-patch14-384[5] as the embedding model. Hardware configurations are provided in Appendix A.

我们为 AndroidWorld 和 SPA-Bench 配置了分支因子为 10、最大深度为 5、步数限制为 30 的探索过程。该设置促成了在 46 个应用中自动发现超过 1300 个知识项 (详细分布见附录 H)。在视觉语义检索中，我们使用了 google/siglip-so400m-patch14-384[5] 作为嵌入模型。硬件配置详见附录 A。

5.1.3 Comparative Baselines

5.1.3 对比基线

We select three baselines with exploration and knowledge extraction capabilities for comprehensive comparison. AppAgent (Zhang et al., 2025) requires manually designed exploration tasks to guide its interaction with GUI environments for knowledge acquisition, whereas AutoDroid (Wen et al., 2024) eliminates task-specific human effort by autonomously generating random action sequences to collect exploration trajectories. Both methods extract structured text-based knowledge from raw textual observations during exploration. DigiRL (Zhou et al., 2024) adopts a distinct reinforcement learning framework to iteratively explore environments while utilizing the Gemini (Google, 2025) model to filter successful trajectories as training data, enabling adaptive explo-

我们选择了三种具备探索和知识提取能力的基线方法进行全面比较。AppAgent(Zhang 等，2025) 需要手动设计探索任务以指导其与 GUI 环境的交互以获取知识，而 AutoDroid(Wen 等，2024) 通过自主生成随机动作序列收集探索轨迹，消除了任务特定的人力投入。两种方法均从探索过程中的原始文本观察中提取结构化文本知识。DigiRL(Zhou 等，2024) 采用不同的强化学习框架迭代探索环境，同时利用 Gemini(Google，2025) 模型筛选成功轨迹作为训练数据，实现自适应利用-

Agent	Input	Base Model	Task Success Rate (%)
AppAgent (Zhang et al., 2025)	SoM	GPT-40	14.0
AutoDroid (Wen et al., 2024)	a11y tree	GPT-40	12.0
CogAgent (Hong et al., 2024)	screen	CogAgent	0
DigiRL (Zhou et al., 2024)	screen	DigiRL	0
M3A (Rawles et al., 2024)	SoM	GPT-40	42.0
MobileAgentV2 (Wang et al., 2024a)	SoM	GPT-40	20.0
SeeAct (Zheng et al., 2024)	SoM	GPT-40	12.0
GUI-explorer (Ours)	SoM	GPT-40	53.7

代理	输入	基础模型	任务成功率 (%)
AppAgent(Zhang 等，2025)	SoM	GPT-4o	14.0
AutoDroid(Wen 等，2024)	a11y 树	GPT-4o	12.0
CogAgent(Hong 等，2024)	屏幕	CogAgent	0
DigiRL(Zhou 等，2024)	屏幕	DigiRL	0
M3A(Rawles 等，2024)	SoM	GPT-4o	42.0
MobileAgentV2(Wang 等，2024a)	SoM	GPT-4o	20.0
SeeAct(Zheng 等，2024)	SoM	GPT-4o	12.0
GUI-explorer(本研究)	SoM	GPT-4o	53.7

Table 1: Performance comparison on SPA-Bench single-app English Level 3 tasks. Results for the first 7 agents are from the SPA-Bench (Chen et al., 2025). SoM (Yang et al., 2023) utilizes the bounding boxes (bbox) recorded in the a11y tree to annotate UI elements with numerical labels in screenshots.

表 1:SPA-Bench 单应用英文三级任务的性能比较。前 7 个代理的结果来自 SPA-Bench(Chen 等, 2025)。SoM(Yang 等，2023) 利用辅助功能树 (a11y tree) 中记录的边界框 (bbox) 在截图中用数字标签标注 UI 元素。

ration with minimal human intervention. For completeness, we also report results from additional baselines in their respective benchmark papers as performance references.

在最小人工干预下进行协作。为完整起见，我们还报告了各自基准论文中的额外基线结果，作为性能参考。

5.2 Experimental Results

5.2 实验结果

Our comprehensive evaluation demonstrates GUI-explorer's superior performance across multiple dimensions. As shown in Table 1, GUI-explorer achieves 53.7% task success rate on SPA-Bench single-app English Level 3 tasks. This represents a 28.1% absolute improvement over M3A, the previous state-of-the-art. Our transition-aware knowledge mining approach proves highly effective in complex, real-world scenarios.

我们的综合评估展示了 GUI-explorer 在多个维度上的卓越表现。如表 1 所示，GUI-explorer 在 SPA-Bench 单应用英文三级任务中实现了 53.7% 的任务成功率。这比之前的最先进方法 M3A 提升了 28.1 个百分点。我们的基于转移感知的知识挖掘方法在复杂的真实场景中表现出高度有效性。

The AndroidWorld results in Table 3 further validate GUI-explorer's generalizability. Our agent achieves 47.4% success rate. This surpasses vision-centric Aria-UI at 44.8%. It also outperforms multimodal M3A at 40.5%.

表 3 中的 AndroidWorld 结果进一步验证了 GUI-explorer 的泛化能力。我们的代理实现了 47.4% 的成功率，超过了以视觉为中心的 Aria-UI 的 44.8%，也优于多模态的 M3A 的 40.5%。

The GUI-KRB evaluation reveals critical insights about MLLMs' GUI reasoning limitations. GPT-4o shows an 18.2% prior knowledge error rate. These errors mainly stem from the misinterpreting of UI components and outdated interface understanding. Our method reduces these errors by 16.0% when applied to Qwen2-VL-72B-Instruct-GPTQ-Int4. This demonstrates the effectiveness of transition-aware knowledge. The dynamic comprehension assessment shows similar improvements. GUI-explorer-enabled models achieve 13.4% lower error rates than base models.

GUI-KRB 评估揭示了多模态大语言模型 (MLLMs) 在 GUI 推理方面的关键局限。GPT-4o 表现出 18.2% 的先验知识错误率。这些错误主要源于对 UI 组件的误解和界面理解的过时。我们的方法在应用于 Qwen2-VL-72B-Instruct-GPTQ-Int4 时将这些错误减少了 16.0%，证明了转移感知知识的有效性。动态理解评估显示了类似的改进。启用 GUI-explorer 的模型比基础模型的错误率低 13.4%。

5.3 Analysis and Discussion

5.3 分析与讨论

The ablation study in Figure 4 quantifies the impact of our key components. Removing dynamic guidance construct by transition-aware knowledge causes a 12.2% performance drop. This empha-

图 4 中的消融研究量化了我们关键组件的影响。移除由转移感知知识构建的动态指导导致性能下降 12.2%。这强调了

[5] https://huggingface.co/google/

[5] https://huggingface.co/google/

siglip-so400m-patch14-384

siglip-so400m-patch14-384

App Category	Retrieval Time per Step (sec)	Ranking Time per Step (sec)	Reasoning Time per Step (sec)	Total Time per Step (sec)	Ranking Cost per Step (0.1USD)	Reasoning Cost per Step (USD)	Total Cost per Step (USD)
Travel & Navigation	7.663	33.084	31.400	72.147	0.017	0.066	0.068
Shopping & Finance	8.613	24.922	36.622	70.157	0.013	0.063	0.065
News & Reading	8.123	17.317	29.272	54.712	0.008	0.053	0.053
System Applications	6.955	31.083	34.513	72.552	0.016	0.065	0.067
Productivity & Tools	7.136	28.091	28.382	63.609	0.016	0.064	0.066
Media & Entertainment	7.549	32.481	30.586	70.615	0.017	0.066	0.068
Communication & Social	6.176	25.662	27.293	59.130	0.013	0.057	0.058
Food & Lifestyle	6.304	9.511	30.481	46.296	0.004	0.041	0.042
Overall	7.120	28.462	30.796	66.378	0.015	0.062	0.064

应用类别	每步检索时间 (秒)	每步排序时间 (秒)	每步推理时间 (秒)	每步总时间 (秒)	每步排序成本 (0.1 美元)	每步推理成本 (美元)	每步总成本 (美元)
旅行与导航	7.663	33.084	31.400	72.147	0.017	0.066	0.068
购物与金融	8.613	24.922	36.622	70.157	0.013	0.063	0.065
新闻与阅读	8.123	17.317	29.272	54.712	0.008	0.053	0.053
系统应用	6.955	31.083	34.513	72.552	0.016	0.065	0.067
生产力与工具	7.136	28.091	28.382	63.609	0.016	0.064	0.066
媒体与娱乐	7.549	32.481	30.586	70.615	0.017	0.066	0.068
通讯与社交	6.176	25.662	27.293	59.130	0.013	0.057	0.058
美食与生活方式	6.304	9.511	30.481	46.296	0.004	0.041	0.042
总体	7.120	28.462	30.796	66.378	0.015	0.062	0.064

Table 2: Per-Step Computational Overhead Analysis: Breakdown of time consumption (seconds) and API costs (USD) across application categories. Note that Ranking Cost per Step is presented in Dimes (0.1 USD) for better readability due to its small magnitude.

表 2: 每步计算开销分析: 各应用类别的时间消耗 (秒) 和 API 费用 (美元) 细分。由于排名每步成本较小，为便于阅读，排名成本以角分 (0.1 美元) 表示。

Agent	Input	Base Model	Task Success Rate (%)
Human (Rawles et al., 2024)	screen	-	80.0
Aguvis (Xu et al., 2024)	screen	GPT-4o	37.1
AppAgent (Zhang et al., 2025)	SoM	GPT-4o	14.9
Aria-UI (Yang et al., 2024b)	screen	GPT-4o	44.8
AutoDroid (Wen et al., 2024)	a11y tree	GPT-4o	15.7
DigiRL (Zhou et al., 2024)	screen	DigiRL	0.9
M3A (Rawles et al., 2024)	SoM	GPT-4o	40.5
Ponder&Press (Wang et al., 2024c)	screen	GPT-4o	34.5
SeeAct (Rawles et al., 2024)	SoM	GPT-4-turbo	15.5
UGround (Gou et al., 2025)	screen	GPT-4o	32.8
GUI-explorer (Ours)	SoM	GPT-4o	47.4

代理	输入	基础模型	任务成功率 (%)
人类 (Rawles 等，2024)	屏幕	-	80.0
Aguvis(Xu 等，2024)	屏幕	GPT-4o	37.1
AppAgent(Zhang 等，2025)	SoM	GPT-4o	14.9
Aria-UI(Yang 等，2024b)	屏幕	GPT-4o	44.8
AutoDroid(Wen 等，2024)	a11y 树	GPT-4o	15.7
DigiRL(Zhou 等，2024)	屏幕	DigiRL	0.9
M3A(Rawles 等，2024)	SoM	GPT-4o	40.5
Ponder&Press(Wang 等，2024c)	屏幕	GPT-4o	34.5
SeeAct(Rawles 等，2024)	SoM	GPT-4-turbo	15.5
UGround(Gou 等，2025)	屏幕	GPT-4o	32.8
GUI-explorer(本研究)	SoM	GPT-4o	47.4

Table 3: Performance comparison on AndroidWorld.

表 3:AndroidWorld 上的性能比较。

Model	Prior Knowledge Error Rate (%)	Dynamic Comprehension Rrror Rate (%)
Qwen2-VL (Wang et al., 2024b)	22.8	19.8
Qwen2.5-VL (Bai et al., 2025)	16.6	14.0
Gemini 2.0 Flash (Google, 2025)	15.2	11.2
GPT-4o (Hurst et al., 2024)	18.2	13.4
GUI-explorer (w/o Ranker)	9.8	6.8
GUI-explorer	6.8	6.4

模型	先验知识错误率 (%)	动态理解错误率 (%)
Qwen2-VL(Wang 等，2024b)	22.8	19.8
Qwen2.5-VL(Bai 等，2025)	16.6	14.0
Gemini 2.0 Flash(Google，2025)	15.2	11.2
GPT-40(Hurst 等，2024)	18.2	13.4
GUI-explorer(无排序器)	9.8	6.8
GUI-explorer	6.8	6.4

Table 4: Performance comparison on GUI-KRB. For all methods, we selected the highest-performing models within device VRAM constraints: Qwen2-VL-72B-Instruct-GPTQ-Int4 for Qwen2-VL, and Qwen2.5-VL- 7B-Instruct for Qwen2.5-VL.

表 4:GUI-KRB 上的性能比较。对于所有方法,我们在设备显存限制内选择了表现最好的模型:Qwen2-VL 选用 Qwen2-VL-72B-Instruct-GPTQ-Int4，Qwen2.5-VL 选用 Qwen2.5-VL-7B-Instruct。

sizes the critical role of transition-aware knowledge. Cross-Environment Guidance improves performance by 4.3% compared to No Guidance. This demonstrates that our transition-aware knowledge exhibits promising generalization capabilities. It effectively guides agent reasoning even in previously unseen scenarios. The knowledge learned can transfer across different UI environments.

强调了转移感知知识的关键作用。跨环境引导 (Cross-Environment Guidance) 相比无引导提升了 4.3% 的性能，表明我们的转移感知知识具有良好的泛化能力，能够有效指导智能体在此前未见过的场景中推理。所学知识可跨不同 UI 环境迁移。

Our computational overhead analysis appears in Table 2. It reveals practical tradeoffs. The ranking component contributes 42.9% of time. This comes primarily from MLLM-based pairwise comparisons. However, we use a merge sort implementation. This ensures $O(n \log n)$ complexity. This keeps practical costs acceptable (0.0015 USD/step average). Additionally, Table 4 shows another benefit. The ranking component reduced the error rate by 3% by prioritizing more relevant knowledge.

我们的计算开销分析见表 2，揭示了实际权衡。排序组件占用 42.9% 的时间，主要来自基于多模态大模型 (MLLM) 的成对比较。但我们采用了归并排序实现，确保了 $O(n \log n)$ 复杂度，使实际成本保持在可接受范围 (平均每步 0.0015 美元)。此外，表 4 显示排序组件通过优先考虑更相关的知识，将错误率降低了 3% 。

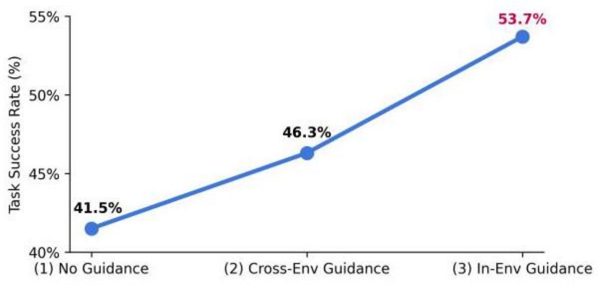


Figure 4: Ablation study of operational guidance configurations on SPA-Bench: (1) Baseline without dynamic guidance, (2) Guidance derived from cross-environment exploration (AndroidWorld), (3) Guidance generated through in-environment exploration (SPA-Bench).

图 4:SPA-Bench 上操作引导配置的消融研究:(1) 无动态引导的基线，(2) 基于跨环境探索 (AndroidWorld) 生成的引导，(3) 基于环境内探索 (SPA-Bench) 生成的引导。

The GUI-KRB results expose two fundamental limitations in current MLLMs. First, there are persistent prior knowledge gaps. Even Gemini 2.0 Flash (Google, 2025) has a 15.2% error rate. Second, there is limited dynamic reasoning capability.

GUI-KRB 结果揭示了当前多模态大模型 (MLLM) 存在的两大根本性限制。首先，持续存在先验知识缺口。即使是 Gemini 2.0 Flash(谷歌，2025) 错误率仍达 15.2%。其次，动态推理能力有限。

The GUI-KRB Dynamic Comprehension task, equivalent to transition-aware knowledge mining, achieved 86.6% accuracy with GPT-4o, indicating comparable reliability in our GPT-4o-built Knowledge Vector Store.

GUI-KRB 动态理解任务，相当于转移感知知识挖掘，使用 GPT-4o 达到了 86.6% 的准确率，表明我们基于 GPT-4o 构建的知识向量库具有相当的可靠性。

6 Conclusion

6 结论

We present GUI-explorer, a GUI agent designed to address two key challenges: misinterpretation of UI components and knowledge obsolescence. Our approach achieves this through autonomous exploration and transition-aware knowledge mining. Experimental results demonstrate our SOTA performance across major benchmarks.We introduce the GUI-KRB benchmark, which reveals fundamental limitations in current MLLMs' interface understanding capabilities. Our dynamic guidance mechanism effectively mitigates these limitations.

我们提出了 GUI-explorer，一种旨在解决 UI 组件误解和知识过时两大关键挑战的 GUI 智能体。该方法通过自主探索和转移感知知识挖掘实现。实验结果表明我们在主要基准测试中达到了最先进水平。我们引入了 GUI-KRB 基准，揭示了当前多模态大模型在界面理解能力上的根本限制。我们的动态引导机制有效缓解了这些限制。

Limitations

限制

While GUI-explorer demonstrates significant advancements in GUI automation, several limitations warrant discussion. First, our current implementation of exploration anchors relies on mobile app manifest declarations (e.g., Android Activity components), which limits direct applicability to web and desktop environments. Second, although the current Knowledge Ranker takes only 28.5 seconds per step, it's still a bit slow. Future work will focus on extending this approach to web and desktop and speeding up Knowledge Ranker.

尽管 GUI-explorer 在 GUI 自动化方面取得了显著进展，但仍存在若干限制。首先，我们当前的探索锚点实现依赖于移动应用清单声明 (如 Android Activity 组件)，限制了其在网页和桌面环境中的直接应用。其次，尽管当前知识排序器每步仅需 28.5 秒，仍显得稍慢。未来工作将致力于扩展至网页和桌面环境，并加速知识排序器。

Ethics Statement

伦理声明

Our work introduces GUI-explorer, an autonomous agent for graphical user interface automation, and raises several ethical considerations inherent to AI-driven interaction systems. First, while our exploration process utilizes application screenshots and accessibility metadata, we strictly employ open-source or publicly available applications, ensuring no collection of private user data or infringement of intellectual property rights.

我们的工作介绍了 GUI-explorer，一种用于图形用户界面自动化的自主智能体，并提出了 AI 驱动交互系统固有的若干伦理考量。首先，虽然我们的探索过程利用了应用截图和辅助功能元数据，但严格使用开源或公开应用，确保不收集私人用户数据或侵犯知识产权。

Second, our reliance on large multimodal models introduces potential risks of perpetuating societal biases embedded in their training data. Though our transition-aware knowledge mechanism mitigates the misinterpretation of UI components, we acknowledge that residual biases in element interpretation could lead to unintended operational consequences. We strongly advocate for human oversight in real-world deployments, particularly for sensitive applications in healthcare or finance domains.

其次，我们依赖的大型多模态模型可能带来训练数据中社会偏见的延续风险。尽管我们的转移感知知识机制减轻了 UI 组件误解，但仍承认元素解释中的残余偏见可能导致意外操作后果。我们强烈建议在实际部署中进行人工监督，尤其是在医疗或金融等敏感领域。

The computational costs associated with our approach (average 66 seconds per interaction step) raise environmental concerns regarding energy consumption. While our method eliminates the need for model retraining—a significant carbon footprint contributor—future work must prioritize efficiency optimizations to enable sustainable scaling.

我们方法的计算成本 (平均每交互步骤 66 秒) 引发了关于能源消耗的环境关注。虽然该方法避免了模型重训练——一大碳足迹来源，但未来工作必须优先考虑效率优化以实现可持续扩展。

We recognize potential dual-use risks where autonomous GUI agents could be misused for malicious automation (e.g., credential stuffing or click fraud), much like other AI technologies can be used for creating deceptive presentations or face presentation attacks (Shao et al., 2019, 2025).

我们认识到潜在的双重用途风险，自主 GUI 智能体可能被滥用于恶意自动化 (如凭证填充或点击欺诈)，类似于其他 AI 技术被用于制造欺骗性展示或面部展示攻击 (Shao 等，2019，2025)。

Finally, our benchmark construction followed ethical annotation practices, with contributors compensated at fair market rates and granted full rights to withdraw their participation.

最后，我们的基准构建遵循了伦理标注规范，贡献者获得了公平市场报酬，并享有完全退出参与的权利。

References

参考文献

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.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat 等. 2023. GPT-4 技术报告. arXiv 预印本 arXiv:2303.08774.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wen-bin Ge, Sibo Song, Kai Dang, Peng Wang, Shi-jie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-vl technical report. arXiv preprint arXiv:2502.13923.

白帅, 陈克勤, 刘雪晶, 王佳林, 葛文斌, 宋思博, 党凯, 王鹏, 王世杰, 唐军, 钟虎门, 朱元志, 杨明坤, 李兆海, 万建强, 王鹏飞, 丁伟, 傅哲仁, 徐一恒, 叶家博, 张曦, 谢天宝, 程泽森, 张航, 杨志博, 徐海洋, 林俊阳. 2025. Qwen2.5-vl 技术报告. arXiv 预印本 arXiv:2502.13923.

Gongwei Chen, Leyang Shen, Rui Shao, Xiang Deng, and Liqiang Nie. 2024. Lion: Empowering multimodal large language model with dual-level visual knowledge. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 26540-26550.

陈功伟, 沈乐阳, 邵锐, 邓翔, 聂立强. 2024. Lion: 通过双层视觉知识赋能多模态大型语言模型. 载于 IEEE/CVF 计算机视觉与模式识别会议 (CVPR) 论文集，页码 26540-26550.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, Kaiwen Zhou, Rui Shao, Liqiang Nie, Yasheng Wang, Jianye HAO, Jun Wang, and Kun Shao. 2025. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In The Thirteenth International Conference on Learning Representations.

陈景轩, 袁德瑞, 谢斌, 杨宇昊, 陈功伟, 吴志豪, 李一行, 周旭睿, 刘伟文, 王帅, 周凯文, 邵锐, 聂立强, 王亚胜, 郝建业, 王军, 邵昆. 2025. Spa-bench: 智能手机代理评测的综合基准. 载于第十三届国际学习表征会议论文集.

Sidong Feng, Haochuan Lu, Jianqin Jiang, Ting Xiong, Likun Huang, Yinglin Liang, Xiaoqin Li, Yuetang Deng, and Aldeida Aleti. 2024. Enabling cost-effective ui automation testing with retrieval-based llms: A case study in wechat. In Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, pages 1973-1978.

冯思东, 卢浩川, 蒋建勤, 熊婷, 黄立坤, 梁英林, 李晓琴, 邓月堂, Aldeida Aleti. 2024. 利用基于检索的大型语言模型实现高性价比的 UI 自动化测试: 以微信为例. 载于第 39 届 IEEE/ACM 自动化软件工程国际会议论文集，页码 1973-1978.

Google. 2025. Gemini 2.0 is now available to everyone.

谷歌. 2025. Gemini 2.0 现已向所有人开放.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In The Thirteenth International Conference on Learning Representations.

苟博宇, 王若涵, 郑博远, 谢雅楠, 常成, 舒一恒, 孙欢, 苏宇. 2025. 像人类一样导航数字世界: 面向 GUI 代理的通用视觉定位. 载于第十三届国际学习表征会议论文集.

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, pages 14281-14290.

洪文怡, 王伟涵, 吕庆松, 徐家政, 余文萌, 纪俊辉, 王岩, 王子涵, 董宇霄, 丁明等. 2024. Cogagent: 面向 GUI 代理的视觉语言模型. 载于 IEEE/CVF 计算机视觉与模式识别会议论文集，页码 14281-14290.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Os-trow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-40 system card. arXiv preprint arXiv:2410.21276.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford 等. 2024. GPT-40 系统说明书. arXiv 预印本 arXiv:2410.21276.

Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steve Ko, Sangeun Oh, and Insik

Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steve Ko, Sangeun Oh, 和 Insik

Shin. 2024. Mobilegpt: Augmenting llm with humanlike app memory for mobile task automation. In Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, ACM Mo-biCom ’24, page 1119-1133, New York, NY, USA. Association for Computing Machinery.

Shin. 2024. MobileGPT: 通过类人应用记忆增强大型语言模型以实现移动任务自动化. 载于第 30 届年度国际移动计算与网络会议，ACM MobiCom ’24，页码 1119-1133，美国纽约，计算机协会出版.

Wei Li, Bing Hu, Rui Shao, Leyang Shen, and Liqiang Nie. 2025a. Lion-fs: Fast & slow video-language thinker as online video assistant. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

李伟, 胡兵, 邵锐, 沈乐阳, 聂立强. 2025a. Lion-fs: 快速与慢速视频语言思考者，作为在线视频助手. 载于 IEEE 计算机视觉与模式识别会议 (CVPR).

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. 2025b. Optimus-1:

Hybrid multimodal memory empowered agents excel in long-horizon tasks. Advances in neural information processing systems, 37:49881-49913.

李在京, 谢宇权, 邵锐, 陈功伟, 蒋东梅, 聂立强. 2025b. Optimus-1: 混合多模态记忆赋能代理在长时任务中的卓越表现. 神经信息处理系统进展，37:49881-49913.

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. 2025c. Optimus-2: Multimodal minecraft agent with goal-observation-action conditioned policy. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.

李在京, 谢宇泉, 邵锐, 陈功伟, 蒋冬梅, 聂立强. 2025c. Optimus-2: 具备目标-观察-动作条件策略的多模态 Minecraft 代理。在 2025 年 IEEE/CVF 计算机视觉与模式识别会议 (CVPR) 论文集。IEEE。

David Marr and Ellen Hildreth. 1980. Theory of edge detection. Proceedings of the Royal Society of London. Series B. Biological Sciences, 207(1167):187- 217.

David Marr 和 Ellen Hildreth. 1980. 边缘检测理论。伦敦皇家学会学报 B 辑: 生物科学, 207(1167):187-217。

OpenAI. 2023. Gpt-4v(ision) system card.

OpenAI. 2023. GPT-4V(视觉) 系统说明。

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. 2025. Ui-tars: Pioneering automated gui interaction with native agents. arXiv preprint arXiv:2501.12326.

秦宇佳, 叶一宁, 方俊杰, 王浩明, 梁世豪, 田志佐, 张俊达, 李嘉豪, 李云欣, 黄世觉等. 2025. UI-TARS: 开创性的本地代理自动化 GUI 交互。arXiv 预印本 arXiv:2501.12326。

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.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala 等. 2024. AndroidWorld: 面向自主代理的动态基准测试环境。arXiv 预印本 arXiv:2405.14573。

Gabriel Herbert Sarch, Lawrence Jang, Michael J Tarr, William W Cohen, Kenneth Marino, and Katerina Fragkiadaki. 2024. Vlm agents generate their own memories: Distilling experience into embodied programs of thought. In The Thirty-eighth Annual Conference on Neural Information Processing Systems.

Gabriel Herbert Sarch, Lawrence Jang, Michael J Tarr, William W Cohen, Kenneth Marino, Katerina Fragkiadaki. 2024. VLM 代理自生成记忆: 将经验提炼为具身思维程序。发表于第 38 届神经信息处理系统年会。

Rui Shao, Xiangyuan Lan, Jiawei Li, and Pong C Yuen. 2019. Multi-adversarial discriminative deep domain generalization for face presentation attack detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10023- 10031.

邵锐, 兰向远, 李嘉伟, 袁炳聪. 2019. 面部呈现攻击检测的多对抗判别深度领域泛化方法。发表于 IEEE/CVF 计算机视觉与模式识别会议论文集，页码 10023-10031。

Rui Shao, Tianxing Wu, and Ziwei Liu. 2023. Detecting and grounding multi-modal media manipulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6904- 6913.

邵锐, 吴天行, 刘子威. 2023. 多模态媒体篡改的检测与定位。发表于 IEEE/CVF 计算机视觉与模式识别会议论文集，页码 6904-6913。

Rui Shao, Tianxing Wu, Liqiang Nie, and Ziwei Liu. 2025. Deepfake-adapter: Dual-level adapter for deep-fake detection. International Journal of Computer Vision, pages 1-16.

邵锐, 吴天行, 聂立强, 刘子威. 2025. Deepfake-Adapter: 用于深度伪造检测的双层适配器。《国际计算机视觉杂志》，页码 1-16。

Rui Shao, Tianxing Wu, Jianlong Wu, Liqiang Nie, and Ziwei Liu. 2024. Detecting and grounding multi-modal media manipulation and beyond. IEEE Transactions on Pattern Analysis and Machine Intelligence.

邵锐, 吴天行, 吴建龙, 聂立强, 刘子威. 2024. 多模态媒体篡改的检测与定位及其扩展。IEEE 模式分析与机器智能汇刊。

Leyang Shen, Gongwei Chen, Rui Shao, Weili Guan, and Liqiang Nie. 2024. Mome: Mixture of multimodal experts for generalist multimodal large language models. In Advances in neural information processing systems.

沈乐阳, 陈功伟, 邵锐, 管伟立, 聂立强. 2024. MOME: 面向通用多模态大语言模型的多模态专家混合方法。发表于神经信息处理系统进展。

Yu Su, Diyi Yang, Shunyu Yao, and Tao Yu. 2024. Language agents: Foundations, prospects, and risks. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts, pages 17-24, Miami, Florida, USA. Association for Computational Linguistics.

苏宇, 杨迪一, 姚顺宇, 余涛. 2024. 语言代理: 基础、前景与风险。发表于 2024 年自然语言处理实证方法会议教程摘要，页码 17-24，美国佛罗里达迈阿密。计算语言学协会。

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 14918-14937, Singapore. Association for Computational Linguistics.

孙伟伟, 颜凌勇, 马新宇, 王帅强, 任鹏杰, 陈竹敏, 尹大伟, 任昭春. 2023. ChatGPT 擅长搜索吗?探究大型语言模型作为重排序代理的能力。发表于 2023 年自然语言处理实证方法会议，页码 14918-14937，新加坡。计算语言学协会。

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodríguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodríguez, Armand Joulin, Edouard Grave, Guillaume Lample. 2023. LLaMA: 开放且高效的基础语言模型。arXiv 预印本 arXiv:2302.13971。

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In Advances in Neural Information Processing Systems, volume 37, pages 2686- 2710. Curran Associates, Inc.

王俊阳, 徐海洋, 贾海涛, 张曦, 闫明, 沈伟舟, 张骥, 黄飞, 桑吉涛. 2024a. Mobile-Agent-V2: 通过多代理协作实现高效导航的移动设备操作助手。发表于神经信息处理系统进展, 第 37 卷, 页码 2686-2710。Curran Associates, Inc.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhi-hao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024b. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191.

王鹏，白帅，谭思楠，王世杰，范志豪，白金泽，陈克勤，刘雪晶，王佳林，葛文斌，范洋，党凯，杜梦飞，任轩成，门锐，刘大义恒，周畅，周景仁，林俊阳。2024b。Qwen2-vl: 提升视觉-语言模型对任意分辨率世界的感知。arXiv 预印本 arXiv:2409.12191。

Yiqin Wang, Haoji Zhang, Jingqi Tian, and Yansong Tang. 2024c. Ponder & press: Advancing visual gui agent towards general computer control. arXiv preprint arXiv:2412.01268.

王一勤，张浩骥，田晶琪，唐彦松。2024c。Ponder & press: 推动视觉 GUI 代理迈向通用计算机控制。arXiv 预印本 arXiv:2412.01268。

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu,

文浩，李元春，刘国宏，赵善辉，余涛，李家俊，蒋世奇，刘云浩，

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, pages 543-557.

张雅琴，刘云新。2024。Autodroid: 基于大型语言模型 (LLM) 的安卓任务自动化。载于第 30 届国际移动计算与网络会议论文集，页 543-557。

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tian-bao 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.

徐一恒，王泽坤，王俊立，卢敦杰，谢天宝，Amrita Saha，Doyen Sahoo，余涛，熊才明。2024。Aguvis: 统一的纯视觉代理实现自主 GUI 交互。arXiv 预印本 arXiv:2412.04454。

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayi-heng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024a. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115.

杨安，杨宝松，张北辰，惠斌元，郑博，余博文，李成元，刘大义恒，黄飞，魏浩然，林欢，杨健，涂建宏，张建伟，杨建新，杨佳熙，周景仁，林俊阳，党凯，卢克明，包克勤，杨可欣，余乐，李梅，薛明峰，张培，朱琴，门锐，林润基，李天昊，夏廷宇，任兴章，任轩成，范洋，苏阳，张一昌，万宇，刘玉琼，崔泽宇，张振儒，邱子涵。2024a。Qwen2.5 技术报告。arXiv 预印本 arXiv:2412.15115。

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chun-yuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. arXiv preprint arXiv:2310.11441.

杨建伟，张浩，李峰，邹雪燕，李春元，高建峰。2023。Set-of-mark 提示释放 GPT-4V 中卓越的视觉定位能力。arXiv 预印本 arXiv:2310.11441。

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2024b. Aria-ui: Visual grounding for gui instructions. arXiv preprint arXiv:2412.16256.

杨宇昊，王越，李东旭，罗子阳，陈蓓，黄超，李俊南。2024b。Aria-ui: 面向 GUI 指令的视觉定位。arXiv 预印本 arXiv:2412.16256。

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR).

姚顺宇，赵杰弗里，余点，杜楠，Izhak Shafran，Karthik Narasimhan，曹源。2023。ReAct: 在语言模型中协同推理与行动。载于国际学习表征会议 (ICLR)。

Qilang Ye, Zitong Yu, Rui Shao, Xinyu Xie, Philip Torr, and Xiaochun Cao. 2024. Cat: Enhancing multimodal large language model to answer questions in dynamic audio-visual scenarios. Preprint, arXiv:2403.04640.

叶启朗，余子彤，邵锐，谢新宇，Philip Torr，曹晓春。2024。CAT: 增强多模态大型语言模型以回答动态视听场景中的问题。预印本，arXiv:2403.04640。

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025. Appagent: Multimodal agents as smartphone users. In Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, CHI ’25, New York, NY, USA. Association for Computing Machinery.

张驰，杨钊，刘佳轩，李彦达，韩宇成，陈昕，黄泽彪，付斌，余刚。2025。AppAgent: 作为智能手机用户的多模态代理。载于 2025 年 CHI 人机交互大会论文集，CHI ’25，美国纽约。计算机协会。

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded. In Forty-first International Conference on Machine Learning.

郑博远，苟博宇，Kil Jihyung，孙欢，苏宇。2024。GPT-4V(ision) 是通用网络代理，前提是有落地支持。载于第 41 届国际机器学习大会。

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2023. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In The Twelfth International Conference on Learning Representations.

郑龙涛，王润东，王新润，安博。2023。Synapse: 基于记忆的轨迹示例提示用于计算机控制。载于第十二届国际学习表征会议。

Yifei Zhou, Hao Bai, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Di-girl: Training in-the-wild device-control agents with autonomous reinforcement learning. In Automated Reinforcement Learning: Exploring Meta-Learning, AutoML, and LLMs.

周一飞，白浩，Mert Cemri，潘佳怡，Alane Suhr，Sergey Levine，和 Aviral Kumar。2024。Di-girl: 通过自主强化学习训练野外设备控制代理。在《自动化强化学习: 探索元学习、自动机器学习 (AutoML) 和大型语言模型 (LLMs)》中。

A Hardware Configurations

A 硬件配置

Hardware configurations were optimized for cost-effectiveness: Most experiments ran on a single NVIDIA GeForce RTX 4070 Laptop GPU (8GB VRAM). For GUI-KRB evaluations involving open-source MLLMs, we scaled to two NVIDIA L40S GPUs (48GB VRAM) to accommodate larger VRAM requirements.

硬件配置经过成本效益优化: 大多数实验在单个 NVIDIA GeForce RTX 4070 笔记本 GPU(8GB 显存) 上运行。对于涉及开源多模态大型语言模型 (MLLMs) 的 GUI-KRB 评估，我们扩展到两块 NVIDIA L40S GPU(48GB 显存) 以满足更大的显存需求。

B GUI-KRB Benchmark Distributions

B GUI-KRB 基准分布

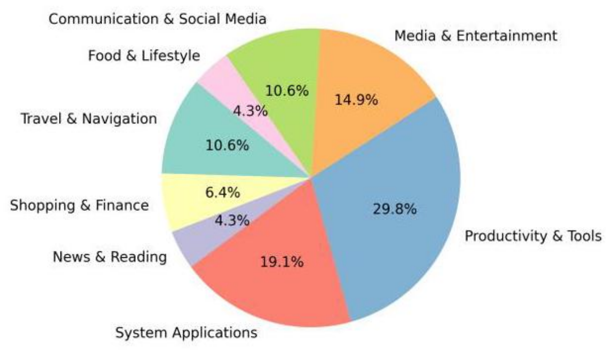


Figure 5: Distribution of apps in GUI-KRB.

图 5:GUI-KRB 中应用程序的分布。

Figure 5 shows the distribution of the number of apps in GUI-KRB.

图 5 展示了 GUI-KRB 中应用程序数量的分布情况。

C GUI-KRB Benchmark Sample Data

C GUI-KRB 基准样本数据

This section presents an example from the GUI-KRB benchmark to illustrate its data structure, as shown in Figure 6. Each sample consists of five main components. First, it contains screenshots captured before and after the interaction with the target element to demonstrate the visual state transition. Second, it includes the accessibility tree representation of the interface. Third, the broader task context describes the necessary interaction with the target element required to complete the task. Fourth, the transition-aware knowledge associated with the element is documented but excluded from the test input. Finally, for automated evaluation purposes, the sample includes evaluation keywords (also excluded from test input) that incorporate synonyms and related terms (such as ”modify” and ”Main”) to accommodate various valid responses and reduce false judgments during model assessment.

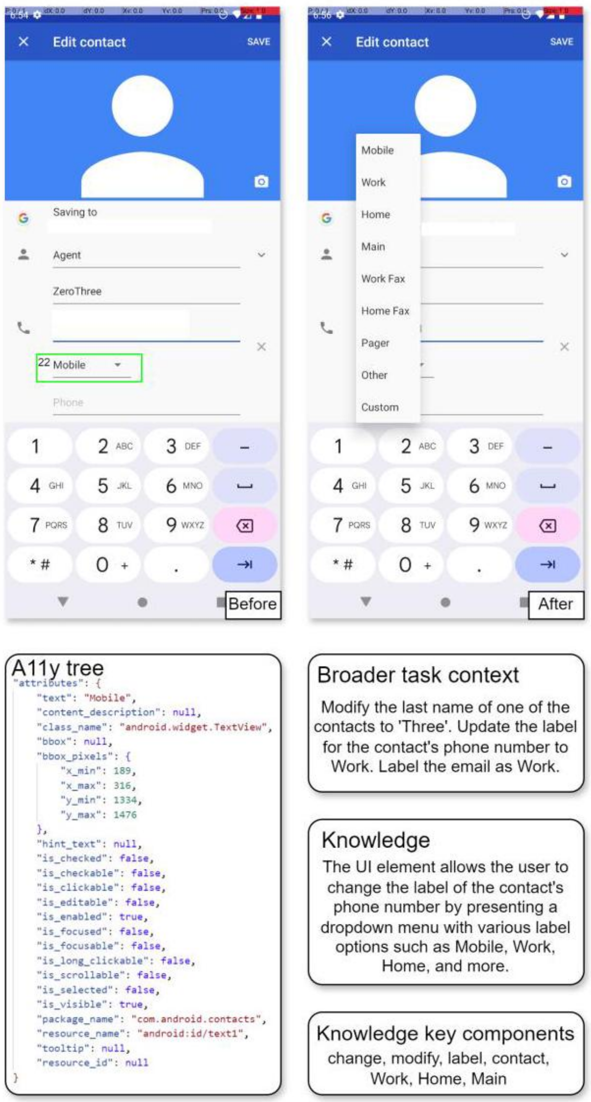
本节展示了 GUI-KRB 基准的一个示例以说明其数据结构，如图 6 所示。每个样本包含五个主要组成部分。首先，包含与目标元素交互前后的截图，以展示视觉状态的变化。其次，包含界面的无障碍树表示。第三，更广泛的任务上下文描述了完成任务所需与目标元素的交互。第四，记录了与元素相关的状态转换知识，但不包含在测试输入中。最后，为自动化评估目的，样本包含评估关键词 (同样不包含在测试输入中)，这些关键词涵盖同义词和相关术语 (如“修改”和“主界面”)，以适应多种有效回答并减少模型评估中的误判。

D Understanding Exploration Anchors

D 理解探索锚点

As introduced in Section 3.1, Exploration Anchors are key entry points to application functionalities. These are explicitly declared by develop-

如第 3.1 节所述，探索锚点是应用功能的关键入口点。这些由开发者在应用的清单文件中明确声明 (例如 Android 的 AndroidManifest.xml)，通常对应于活动 (activities)，代表应用中的单个屏幕或独立功能单元。



```
"attributes": {
    "text": "Mobile",
    "content_description": null,
    "class_name": "android.widget.TextView",
    "bbox": null,
    "bbox_pixels": {
        "x_min": 189,
        "x_max": 316,
        "y_min": 1334,
        "y_max": 1476
    },
    "hint_text": null,
    "is_checked": false,
    "is_checkable": false,
    "is_clickable": false,
    "is_editable": false,
    "is_enabled": true,
    "is_focused": false,
    "is_focusable": false,
    "is_long_clickable": false,
    "is_scrollable": false,
    "is_selected": false,
    "is_visible": true,
    "package_name": "com.android.contacts",
    "resource_name": "android:id/text1",
    "tooltip": null,
    "resource_id": null
}
```

Broader task context

Modify the last name of one of the contacts to 'Three'. Update the label for the contact's phone number to Work. Label the email as Work.

Knowledge

The UI element allows the user to change the label of the contact's phone number by presenting a dropdown menu with various label options such as Mobile, Work, Home, and more.

Knowledge key components

change, modify, label, contact, Work, Home, Main

Figure 6: A comprehensive sample from GUI-KRB benchmark illustrating: (1) before/after screenshots of target element interaction, (2) accessibility tree representation, (3) broader task context, (4) transition-aware knowledge (excluded from test input), and (5) evaluation keywords with synonyms for robust assessment.

图 6:GUI-KRB 基准的综合样本，展示:(1) 目标元素交互前后截图，(2) 无障碍树表示，(3) 更广泛的任务上下文，(4) 状态转换知识 (不包含在测试输入中)，以及 (5) 包含同义词的评估关键词以实现稳健评估。

ers within the app's manifest file (e.g., Android's AndroidManifest.xml) and typically correspond to activities, which represent individual screens or distinct functional units within the app.

开发者在应用的清单文件中声明，通常对应于活动，代表应用中的单个屏幕或独立功能单元。

D.1 Extraction and Utilization

D.1 提取与利用

Exploration Anchors are systematically extracted from the app’s manifest file, which enumerates all developer-declared activities. During the autonomous exploration process, the agent utilizes these anchors to formulate intermediate navigational goals. For example, if an anchor corresponding to a specific activity (e.g., an activity named ShareInstagramStory) is not directly accessible

探索锚点系统地从应用的清单文件中提取，该文件列举了所有开发者声明的活动。在自主探索过程中，代理利用这些锚点制定中间导航目标。例如，如果对应特定活动 (如名为 ShareInstagramStory 的活动) 的锚点当前屏幕无法直接访问，

from the agent’s current screen, the agent will iteratively generate sub-goals, such as navigating through menus or interacting with UI elements, to eventually reach the target anchor activity.

代理将迭代生成子目标，如通过菜单导航或与界面元素交互，最终到达目标锚点活动。

D.2 Illustrative Case Studies

D.2 说明性案例研究

To further clarify the concept and utility of Exploration Anchors, consider the following examples from the ”Retro Music” application. Its manifest file declares activities including ShareInstagramStory, DriveModeActivity, and RestoreActivity. If the agent begins its exploration from a screen displaying ”Most played” tracks, it might generate intermediate goals such as:

为了进一步阐明探索锚点 (Exploration Anchors) 的概念和作用，以下以“复古音乐”(Retro Music) 应用为例。其清单文件声明了包括 ShareInstagramStory、DriveModeActivity 和 RestoreActivity 等活动。如果代理从显示“播放次数最多”曲目的界面开始探索，可能会生成如下中间目标:

1. ”Tap ’Most played’, then share the top song to Instagram Story.”

1. “点击‘播放次数最多’，然后将最热门的歌曲分享到 Instagram 故事。”

(Targeting anchor: ShareInstagramStory)

(目标锚点:ShareInstagramStory)

2. ”Tap the settings icon, then navigate to Drive Mode and enable it.”

2. “点击设置图标，然后进入驾驶模式并启用它。”

(Targeting anchor: DriveModeActivity)

(目标锚点:DriveModeActivity)

E Error Analysis

E 错误分析

In this section, we categorize and discuss three primary error types observed in our evaluation trajectories, detailing their component-level manifestations and root causes.

本节中，我们对评估轨迹中观察到的三类主要错误进行分类和讨论，详细说明其组件级表现及根本原因。

E.1 Perceptual Errors

E.1 感知错误

Perceptual errors occur when agents misinterpret visual or contextual cues. For instance, in the SPA-Bench dictionary_merriam_webster_2 task, the agent failed to recognize that a solid icon indicated an already ”saved” word. Instead, it redundantly clicked the ”save” button, unintentionally unsaving the word. Such errors often stem from limitations in grounding GUI elements (e.g., distinguishing icon states like filled versus hollow) or parsing dynamic UI hierarchies (e.g., overlays, scrolling content).

感知错误发生在代理误解视觉或上下文线索时。例如, 在 SPA-Bench 的 dictionary_merriam_webster_2 任务中，代理未能识别实心图标表示单词已“保存”，反而重复点击“保存”按钮，导致单词被意外取消保存。此类错误通常源于对 GUI 元素的定位能力不足 (如区分实心与空心图标状态) 或解析动态 UI 层级结构的困难 (如覆盖层、滚动内容)。

E.2 Reasoning Errors

E.2 推理错误

Reasoning errors arise from incomplete task decomposition or flawed step-by-step logic. For example, in SPA-Bench contacts_2, the agent appended ”Three” to an existing last name instead of first deleting the original text, thereby violating the task’s implicit requirement to ”replace” rather than ”modify”. These errors reflect limitations in the base model’s ability to infer nuanced constraints

推理错误源于任务分解不完整或逐步逻辑缺陷。例如，在 SPA-Bench 的 contacts_2 任务中，代理在未先删除原有文本的情况下，将“三”附加到已有的姓氏上，违反了任务隐含的“替换”而非“修改”要求。这反映了基础模型在推断细微约束

(e.g., distinguishing ”edit” from ”overwrite”) or manage multi-step dependencies (e.g., ensuring changes are saved before exiting).

(如区分“编辑”与“覆盖”) 或管理多步骤依赖关系 (如确保更改保存后再退出) 方面的局限性。

E.3 Missing Knowledge Errors

E.3 知识缺失错误

Missing knowledge errors occur when agents lack app-specific prior knowledge critical for task completion. For instance, in SPA-Bench booking.com_5, the agent exhausted its step limit searching for a ”currency” setting under ”Preferences” instead of the correct ”Payment details” section. This highlights challenges in efficiently navigating unfamiliar UIs, particularly when key functionalities are nested within non-intuitive menus.

知识缺失错误发生在代理缺乏完成任务所需的应用特定先验知识时。例如，在 SPA-Bench 的 booking.com_5 任务中，代理在“偏好设置”下寻找“货币”设置，耗尽了步骤限制，而正确位置应为“支付详情”部分。这凸显了在不熟悉的 UI 中高效导航的挑战，尤其当关键功能隐藏在不直观的菜单中时。

F Comparison with Alternative Ranking Methods

F 与其他排序方法的比较

To validate the necessity and efficiency of our pairwise knowledge ranking module, we conducted comparative experiments against three alternative ranking strategies: (1) Direct LLM judgment: The LLM directly assesses the usefulness of each knowledge piece. (2) Confidence scores: Ranking based on confidence scores assigned to knowledge. (3) Sliding-window ranking (Sun et al., 2023): A sequential ranking approach.

为验证我们成对知识排序模块的必要性和效率，我们进行了与三种替代排序策略的对比实验:(1) 直接 LLM 判断: 由 LLM 直接评估每条知识的有用性。(2) 置信度评分: 基于知识的置信度分数进行排序。(3) 滑动窗口排序 (Sun 等，2023): 一种顺序排序方法。

We evaluated these methods on the GUI-KRB benchmark using an identical experimental setup (Qwen2-VL-72B-Instruct-GPTQ-Int4). The results

我们在 GUI-KRB 基准测试上使用相同的实验设置 (Qwen2-VL-72B-Instruct-GPTQ-Int4) 评估了这些方法。结果

Ranking Method	Prior Knowledge Error Rate (%)	Dynamic Comprehension Error Rate (%)
Direct LLM judgment	8.8	8.8
Confidence score	7.6	7.0
Sliding-window (Sun et al., 2023)	6.8	6.4
Ours	6.8	6.4

排序方法	先验知识错误率 (%)	动态理解错误率 (%)
直接大型语言模型 (LLM) 判断	8.8	8.8
置信度评分	7.6	7.0
滑动窗口法 (Sun 等，2023)	6.8	6.4
本方法	6.8	6.4

Table 5: Comparison of Different Knowledge Ranking Methods on GUI-KRB. Error rates (%) are reported. Lower is better.

表 5:GUI-KRB 上不同知识排序方法的比较。报告错误率 (%)。数值越低越好。

in Table 5 demonstrate that both direct LLM judgment and confidence-based scoring yield higher error rates than our proposed method. While sliding-window ranking achieves comparable error rates to our approach, it suffers from a time complexity of $O(n)$ due to its sequential traversal of knowledge items. In contrast, our merge-sort-inspired pairwise comparison strategy allows for $O(\log n)$ time complexity through parallelizable divide-and-conquer iterations. This inherent parallelism makes our method significantly more scalable for large knowledge sets. For instance, to reduce latency with sliding-window ranking, one would need to

表 5 中显示，直接使用大型语言模型 (LLM) 判断和基于置信度的评分方法的错误率均高于我们提出的方法。虽然滑动窗口排序的错误率与我们的方法相当，但由于其对知识项的顺序遍历，时间复杂度为 $O(n)$ 。相比之下，我们受归并排序启发的成对比较策略通过可并行的分治迭代实现了 $O(\log n)$ 的时间复杂度。这种内在的并行性使得我们的方法在处理大规模知识集时具有显著的可扩展性。例如，为了降低滑动窗口排序的延迟，必须

compromise sorting quality (e.g., by enlarging window or step sizes). Crucially, the computational overhead of merge operations in our method is negligible, as LLM-based pairwise comparisons (which incur seconds-level delays) dominate the overall runtime.

牺牲排序质量 (例如，通过增大窗口或步长)。关键是，我们方法中归并操作的计算开销可以忽略不计，因为基于 LLM 的成对比较 (耗时为秒级) 主导了整体运行时间。

Furthermore, as shown in Table 4, disabling ranking increases the Prior Knowledge error rate from 6.8% to 9.8%, highlighting the critical role of ranking in performance.

此外，如表 4 所示，禁用排序会使先验知识错误率从 6.8% 升至 9.8%，凸显了排序在性能中的关键作用。

G Generalization and Robustness Analysis

G 泛化性与鲁棒性分析

This section presents additional experimental results to address the robustness of our autonomous exploration approach with incomplete metadata and the generalization of our framework to web environments.

本节展示了额外实验结果，旨在验证我们自主探索方法在元数据不完整情况下的鲁棒性，以及我们框架向网络环境的泛化能力。

G.1 Robustness to Incomplete Metadata

G.1 对不完整元数据的鲁棒性

To assess the performance of our autonomous exploration approach when application metadata is incomplete or unavailable, we conducted supplementary experiments where exploration task goal generation relied solely on screenshot data, without access to manifest-derived structural information (Exploration Anchors as described in Section D). This scenario simulates conditions where only visual information is accessible.

为评估当应用元数据不完整或不可用时我们自主探索方法的表现，我们进行了补充实验，其中探索任务目标生成仅依赖截图数据，未使用清单 (manifest) 派生的结构信息 (即第 D 节所述的探索锚点)。该场景模拟了仅能获取视觉信息的条件。

We tested this screenshot-only configuration on the Retro Music and Broccoli applications, evaluating performance using the corresponding task sets from the AndroidWorld benchmark. The results, compared against a baseline without any exploration and our full method (which utilizes manifest metadata), are presented in Table 6.

我们在 Retro Music 和 Broccoli 应用上测试了该仅截图配置，使用 AndroidWorld 基准对应的任务集评估性能。结果与无任何探索的基线及利用清单元数据的完整方法进行了对比，详见表 6。

Method Configuration	Task Success Rate (%)
No exploration (Baseline)	16.7
Ours (Screenshots only)	22.2
Ours (Full method)	33.3

方法配置	任务成功率 (%)
无探索 (基线)	16.7
我们的 (仅截图)	22.2
我们的 (完整方法)	33.3

Table 6: Performance comparison on selected Android-World tasks (Retro Music and Broccoli apps) under varying levels of metadata availability for exploration task goal generation.

表 6: 在不同元数据可用性水平下，针对选定的 Android-World 任务 (Retro Music 和 Broccoli 应用) 进行探索任务目标生成的性能比较。

The results indicate that even when restricted to using only screenshots for generating exploration goals, our method achieves a 5.5% absolute improvement in task success rate over the nonexploratory baseline. While the integration of full metadata (Exploration Anchors) clearly yields superior performance (33.3% success rate), these experiments confirm that the core exploration strategy remains effective and provides benefits even in metadata-deficient scenarios. This demonstrates a degree of robustness in our approach, as the visual cues present in screenshots can still guide meaningful exploration, albeit less efficiently than when supplemented with structural priors from manifest files.

结果表明，即使仅限于使用截图来生成探索目标，我们的方法在任务成功率上也比非探索基线提高了 5.5 个百分点。虽然整合完整元数据 (探索锚点) 显然带来了更优的性能 (成功率为 33.3%)，但这些实验确认了核心探索策略依然有效，并且即使在元数据缺乏的情况下也能带来益处。这展示了我们方法的鲁棒性，因为截图中的视觉线索仍能引导有意义的探索，尽管效率不及结合了 manifest 文件中结构先验时。

G.2 Generalization to Web Environments

G.2 向 Web 环境的泛化

While our method, particularly the Autonomous Exploration component described in Section 3.1, leverages Android-specific structural information (i.e., Exploration Anchors from manifest files) to guide exploration, it is pertinent to investigate the potential for the acquired knowledge to generalize to other platforms, such as web applications.

尽管我们的方法，特别是第 3.1 节描述的自主探索组件，利用了 Android 特有的结构信息 (即来自 manifest 文件的探索锚点) 来指导探索，但有必要研究所获得的知识是否能泛化到其他平台，如 Web 应用。

To this end, we conducted additional experiments to evaluate the cross-platform applicability of the transition-aware knowledge mined from Android environments. Specifically, we applied our framework, using knowledge acquired solely from Android app exploration, to tasks in a web environment. The evaluation was performed on the ”website” split of the Multimodal-Mind2Web test set, comprising 20 distinct samples. We compared the performance of a baseline agent without any augmented knowledge against an agent augmented with static knowledge derived from our Android exploration phase. The results are presented in Table 7.

为此，我们进行了额外实验，评估从 Android 环境挖掘的转移感知知识的跨平台适用性。具体来说，我们将仅从 Android 应用探索中获得的知识应用于 Web 环境中的任务。评估在 Multimodal-Mind2Web 测试集的“网站”分割上进行，包含 20 个不同样本。我们比较了无任何增强知识的基线代理与使用从 Android 探索阶段获得的静态知识增强的代理的性能。结果见表 7。

Knowledge Configuration	Macro Element Accuracy (%)	Macro Step Accuracy (%)
No augmented knowledge (Baseline)	45.97	42.31
+ Static Android-derived knowledge	47.26	43.05

知识配置	宏观元素准确率 (%)	宏观步骤准确率 (%)
无增强知识 (基线)	45.97	42.31
+ 静态 Android 派生知识	47.26	43.05

Table 7: Performance on the Multimodal-Mind2Web ”website” test split (20 samples) with and without leveraging static knowledge acquired from Android app exploration.

表 7: 在 Multimodal-Mind2Web“网站”测试集 (20 个样本) 上，利用与不利用从 Android 应用探索中获得的静态知识的性能表现。

The results show a modest improvement in both macro element accuracy (+1.29%) and macro step accuracy (+0.74%) when leveraging knowledge acquired from Android applications. This suggests that some fundamental aspects of UI interaction logic, such as understanding hierarchical structures, common action sequences (e.g., ”select then confirm”), and visual-textual correlations, possess a degree of cross-platform generalizability. While these improvements are not as substantial as those

结果显示，利用从 Android 应用获得的知识，宏观元素准确率提升了 1.29%，宏观步骤准确率提升了 0.74%。这表明 UI 交互逻辑的一些基本方面，如理解层级结构、常见操作序列 (例如“先选择再确认”) 以及视觉与文本的关联，具有一定的跨平台泛化能力。尽管这些提升不及 Android 领域内观察到的显著，但

observed within the Android domain, they indicate that the core principles of transition-aware knowledge are not strictly confined to mobile environments and can offer some benefit in web contexts without any web-specific exploration or fine-tuning.

它们表明转移感知知识的核心原理并非严格局限于移动环境, 在未进行任何针对网页的探索或微调的情况下，也能在网页环境中带来一定的益处。

It is important to note that web environments present unique challenges, such as highly dynamic content, complex DOM structures, and browser-specific interactions, which are not fully addressed by knowledge solely derived from Android apps. Future work will focus on adapting the autonomous exploration and knowledge mining mechanisms specifically for web and desktop platforms to achieve more significant performance gains. However, these initial findings provide encouraging evidence of the underlying generalizability of the learned GUI interaction patterns.

需要注意的是，网页环境存在独特挑战，如高度动态的内容、复杂的 DOM 结构以及浏览器特有的交互，这些仅凭从 Android 应用获得的知识无法完全解决。未来工作将重点针对网页和桌面平台，调整自主探索和知识挖掘机制，以实现更显著的性能提升。然而，这些初步发现为所学 GUI 交互模式的潜在泛化能力提供了鼓舞人心的证据。

H Distribution of Transition-aware Knowledge

H 转移感知知识的分布

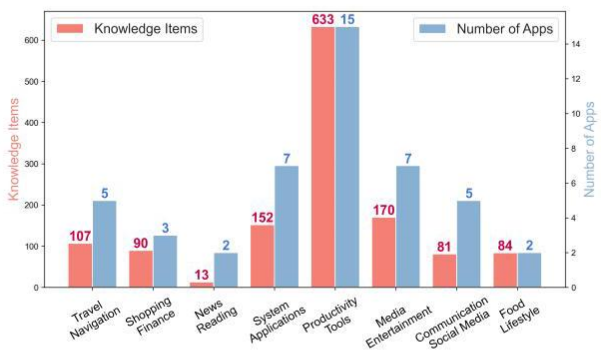


Figure 7: Distribution of transition-aware knowledge gained through autonomous exploration.

图 7: 通过自主探索获得的转移感知知识的分布。

Figure 7 shows the distribution of transition-aware knowledge gained through autonomous exploration.

图 7 展示了通过自主探索获得的转移感知知识的分布情况。

I Prompting Templates of GUI-explorer

I GUI-explorer 的提示模板

I.1 Prompting Template of Function-aware Task Goal Generator

I.1 功能感知任务目标生成器的提示模板

Given the screenshot of app name and its available activities, generate a comprehensive list of practical user tasks that:

给定应用名称的截图及其可用活动，生成一份实用用户任务的全面列表，要求:

1. Start from the current screen shown in the screenshot 2. Can be completed within 10-30 steps

1. 从截图中显示的当前界面开始 2. 可在 10-30 步内完成

3. Utilize the app's full feature set based on the activity list

3. 基于活动列表，充分利用应用的全部功能

4. Are concrete and specific (like searching for a particular item rather than just ”search”)

4. 具体且明确 (例如搜索特定项目，而非仅仅“搜索”)

5. Cover different user interaction patterns (viewing, editing, sharing, etc.)

5. 涵盖不同的用户交互模式 (浏览、编辑、分享等)

7. Represent realistic user behaviors and goals

7. 代表真实的用户行为和目标

8. Avoid excessive steps on form-filling or scrolling pages

8. 避免在填写表单或滚动页面上花费过多步骤

Important context:

重要背景:

- App name: app name
  - 应用名称:app name
- Package name: package name
  - 包名:package name
- Available activities (app screens/features): activity list
  - 可用活动 (应用界面/功能):activity list

Format requirements:

格式要求:

1. List only the tasks without explanations or commentary

1. 仅列出任务，不要解释或评论

2. Each task should be a single, clear directive

2. 每个任务应为单一明确的指令

3. Use specific examples (e.g., concrete search terms, actions, settings)

3. 使用具体示例 (如具体搜索词、操作、设置)

4. Include the expected outcome where relevant

4. 相关时包含预期结果

5. Tasks should follow this pattern: [Starting action] + [Specific steps] + [End goal]

5. 任务应遵循此模式:[起始操作] + [具体步骤] + [最终目标]

Example tasks from other apps (for reference only):

其他应用示例任务 (仅供参考):

1. Search for ”ocean waves” white noise, then sort results by most played

1. 搜索“海浪”白噪音，然后按播放次数排序结果

2. Open the first recommended video, then post ”Great content!” as a comment

2. 打开第一个推荐视频，然后发表评论“内容很棒!”

3. Play the trending video, then add it to your ”Watch Later” playlist

3. 播放热门视频，然后添加到“稍后观看”播放列表

4. Navigate to the comments section of a featured video, then like the top comment

4. 进入精选视频的评论区，然后点赞置顶评论

Generate diverse tasks that would help a user explore and utilize all major features visible in the screenshot and implied by the activity list.

生成多样化任务，帮助用户探索并利用截图中显示及活动列表暗示的所有主要功能。

I.2 Prompting Template of Unsupervised Mining of Transition-aware Knowledge

I.2 无监督挖掘具备转变感知知识的提示模板

Objective: Describe the functionality of a specific UI element in a mobile app screenshot. Input:

目标: 描述移动应用截图中某个特定 UI 元素的功能。输入:

- Two screenshots: Before and after interacting with a UI element

  - 两张截图: 交互前和交互后

- UI element marked with a numeric tag in the top-left corner

  - 左上角带数字标签的 UI 元素

- Element number: numeric tag of element

- 元素编号: 元素的数字标签

- Broader task context: task description

  - 更广泛的任务背景: 任务描述

- Action taken: action

  - 执行动作: 动作

- UI Element Attributes:

  - UI 元素属性:

ui element attributes

ui 元素属性

...
Requirements for Functionality Description:

功能描述要求:

1. Concise: 1-2 sentences

1. 简洁:1-2 句

2. Focus on general function, not specific details

2. 侧重一般功能，不涉及具体细节

3. Avoid mentioning the numeric tag

3. 避免提及数字标签

4. Use generic terms like ”UI element” or appropriate pronouns

4. 使用通用术语如“UI 元素”或适当代词

Example:

示例:

- Incorrect: ”Tapping the element #3 displays David’s saved recipes in the results panel”

  - 错误示范:“点击元素 #3 会在结果面板显示 David 保存的食谱”

and displays matching results”

并显示匹配结果”

Guidance:

指导:

- Describe the core action and immediate result of interacting with the UI element

  - 描述与 UI 元素交互的核心动作及其直接结果

- Prioritize clarity and generality in the description

  - 优先保证描述的清晰性和通用性

I.3 Prompting Template of Knowledge Ranker

I.3 知识排序器提示模板

Given the user instruction: task goal, determine which of the following two knowledge entries is more useful.

给定用户指令: 任务目标，判断以下两条知识条目中哪一条更有用。

Respond ONLY with a integer value:

仅以整数值回复:

1 means Knowledge A is strictly better.

1 表示知识 A 明显更优。

2 means Knowledge B is strictly better.

2 表示知识 B 明显更优。

Knowledge A: knowledge a

知识 A: 知识 a

Knowledge B: knowledge b

知识 B: 知识 b

Please provide your response:

请提供您的回答:

I.4 Prompting Template of Reasoning

I.4 推理提示模板

##Role Definition

角色定义

You are an Android operation AI that fulfills user requests through precise screen interactions.

您是一款通过精确屏幕操作满足用户请求的 Android 操作 AI。

The current screenshot and the same screenshot with bounding boxes and labels added are also given to you. ##Action Catalog

当前截图及带有边界框和标签的同一截图也一并提供给您。## 操作目录

Available actions (STRICT JSON FORMAT REQUIRED):

可用操作 (严格 JSON 格式要求):

1. Status Operations:

1. 状态操作:

- Task Complete: {”action_type”: ”status”, ”goal_status”: ”complete”}

  - 任务完成:{”action_type”: ”status”, ”goal_status”: ”complete”}

- Task Infeasible: {”action_type”: ”status”, ”goal_status”: ”infeasible”}

  - 任务不可行:{”action_type”: ”status”, ”goal_status”: ”infeasible”}

2. Information Actions:

2. 信息操作:

- Answer Question: {”action_type”: ”answer”, ”text”: ”<answer_text>”}

  - 回答问题:{”action_type”: ”answer”, ”text”: ”<answer_text>”}

3. Screen Interactions:

3. 屏幕交互:

- Tap Element: {”action_type”: ”click”, ”index”: <visible_index>

- 点击元素:{”action_type”: ”click”, ”index”: <visible_index>

- Long Press: {”action_type”: ”long_press”, ”index”: <visible_index>}

- 长按:{”action_type”: ”long_press”, ”index”: <visible_index>}

- Scroll: Scroll the screen or a specific scrollable UI element. Use the ’index’ of the target element if scrolling a specific element, or omit ’index’ to scroll the whole screen. ”action_type”: ”scroll”, ”direction”: <”up”|”down”|”left”|”right”>, ”index”: <optional_target_index>

- 滚动: 滚动屏幕或特定的可滚动 UI 元素。若滚动特定元素，使用目标元素的“index”，否则省略“index”以滚动整个屏幕。”action_type”: ”scroll”, ”direction”: <”up”|”down”|”left”|”right”>, ”index”: <optional_target_index>

4. Input Operations:

4. 输入操作:

- Text Entry: {”action_type”: ”input_text”, ”text”: ”<content>”, ”index”: <text_field_index>}

- 文本输入:{”action_type”: ”input_text”, ”text”: ”<content>”, ”index”: <text_field_index>}

- Keyboard Enter: {”action_type”: ”keyboard_enter”} 5. Navigation:

- 键盘回车:{”action_type”: ”keyboard_enter”} 5. 导航:

- Home Screen: {”action_type”: ”navigate_home”}

- 主页:{”action_type”: ”navigate_home”}

- Back Navigation: {”action_type”: ”navigate_back”} 6. System Actions:

- 返回导航:{”action_type”: ”navigate_back”} 6. 系统操作:

- Launch App: {”action_type”: ”open_app”, ”app_name”:

- 启动应用:{”action_type”: ”open_app”, ”app_name”:

- Wait Refresh: {”action_type”: ”wait”}

- 等待刷新: {”action_type”: ”wait”}

User Goal: task goal

用户目标: 任务目标

##Execution Context

执行环境

Action History:

操作历史:

history

历史记录

Visible UI Elements (Only interact with *visible=true elements):

可见的界面元素 (仅与 *visible=true 的元素交互):

ui elements

界面元素

##Core Strategy

核心策略

1. Path Optimization:

1. 路径优化:

- Prefer direct methods (e.g., open_app > app drawer navigation)

  - 优先选择直接方法 (例如，打开应用 > 应用抽屉导航)

- Always use the ’input_text’ action for entering text into designated text fields.

  - 输入文本时始终使用“input_text”操作，针对指定文本框。

- Verify element visibility (’visible=true’) before attempting any interaction (click, long_press, input_text). Do not interact with elements marked ’visible=false’.

  - 在尝试任何交互 (点击、长按、输入文本) 前，确认元素可见 (’visible=true’)。不要与标记为’visible=false’ 的元素交互。

- Use ’scroll’ when necessary to bring off-screen elements into view. Prioritize scrolling specific containers (’index’ provided) over full-screen scrolls if possible.

  - 必要时使用“滚动”操作将屏幕外元素带入视野。优先滚动特定容器 (提供“index”) 而非全屏滚动。

2. Error Handling Protocol:

2. 错误处理协议:

- Switch approach after $\geq$ 2 failed attempts

  - 在 $\geq$ 2 次失败尝试后切换策略

- Prioritize scrolling (’scroll’ action) over force-acting on invisible elements

  - 优先使用滚动 (“scroll”操作)，而非强制操作不可见元素

- If an element is not visible, use ’scroll’ in the likely direction (e.g., ’down’ to find elements below the current view).

  - 如果元素不可见，使用“滚动”朝可能的方向 (例如，向下滚动以查找当前视图下方的元素)。

- Try opposite scroll direction if initial fails (up/down, left/right)

  - 如果初始滚动失败，尝试相反方向滚动 (上下、左右)。

- If the ’open_app’ action fails to correctly open the app, find the corresponding app in the app drawer and open it.

  - 如果“open_app”操作未能正确打开应用，找到应用抽屉中的对应应用并打开它。

3. Information Tasks:

3. 信息任务:

- MANDATORY: Use answer action for questions

  - 必须: 对问题使用回答操作。

- Verify data freshness (e.g., check calendar date) ##Expert Techniques

  - 验证数据的新鲜度 (例如，检查日历日期)## 专家技巧

Here are some tips for you:

这里有一些提示给你:

knowledge

知识

##Response Format

响应格式

STRICTLY follow:

严格遵循:

Reasoning: [Step-by-step analysis covering:

推理:[逐步分析涵盖:

- Visibility verification

  - 可见性验证

- History effectiveness evaluation

  - 历史有效性评估

- Alternative approach comparison

  - 替代方法比较

- Consideration of scrolling if needed] - Consideration of scrolling if needed] Action: [SINGLE JSON action from catalog] Action: [SINGLE JSON action from catalog] Generate response: Generate response:

  - 如有需要考虑滚动] - 如有需要考虑滚动] 操作:[目录中的单一 JSON 操作] 操作:[目录中的单一 JSON 操作] 生成响应: 生成响应:

J Prompting Templates of GUI-KRB

GUI-KRB 的 J 提示模板

J.1 Prompting Template of Prior Knowledge Task

J.1 先验知识任务的提示模板

Objective: Describe the functionality of a specific UI element in a mobile app screenshot. Input:

目标: 描述移动应用截图中某个特定 UI 元素的功能。输入:

- One screenshot: Before interacting with a UI element

  - 一张截图: 在与 UI 元素交互之前

- UI element marked with a numeric tag in the top-left corner

  - UI 元素左上角标有数字标签

- Element number: numeric tag of element

  - 元素编号: 元素的数字标签

- Broader task context: task description

  - 更广泛的任务背景: 任务描述

- UI Element Attributes:

  - UI 元素属性:

ui element attributes

ui 元素属性

“

Requirements for Functionality Description:

功能描述要求:

1. Concise: 1-2 sentences

1. 简洁:1-2 句

2. Focus on general function, not specific details

2. 侧重于一般功能，不涉及具体细节

3. Avoid mentioning the numeric tag

3. 避免提及数字标签

4. Use generic terms like ”UI element” or appropriate pronouns

4. 使用通用术语如“UI 元素”或适当的代词

Example:

示例:

- Incorrect: ”Tapping the element #3 displays David’s saved recipes in the results panel”

  - 错误:“点击元素 #3 会在结果面板显示 David 保存的食谱”

- Correct: ”Tapping this element will initiates a search and displays matching results”

- 正确:“点击该元素将启动搜索并显示匹配结果”

Guidance:

指导:

- Describe the core action and immediate result of interacting with the UI element

- 描述与 UI 元素交互的核心操作及其直接结果

- Infer functionality based on the current screen context

- 根据当前屏幕上下文推断功能

- Prioritize clarity and generality in the description

- 优先保证描述的清晰性和通用性

J.2 Prompting Template of Dynamic Comprehension Task

J.2 动态理解任务的提示模板

Same as Appendix I.2.

同附录 I.2。

J.3 Prompting Templates for GUI-explorer (w/o Ranker)

J.3 GUI 探索器 (无排序器) 的提示模板

J.3.1 Prompting Template of Prior Knowledge Task

J.3.1 先验知识任务的提示模板

Objective: Describe the functionality of a specific UI element in a mobile app screenshot. Input:

目标: 描述移动应用截图中特定 UI 元素的功能。输入:

- One screenshot: Before interacting with a UI element

- 一张截图: 交互前的 UI 元素

- UI element marked with a numeric tag in the top-left corner

  - 左上角带数字标签的 UI 元素

- Element number: numeric tag of element

  - 元素编号: 元素的数字标签

- Broader task context: task description

  - 更广泛的任务背景: 任务描述

- UI Element Attributes:

  - UI 元素属性:

ui element attributes

ui 元素属性

44

- Similar UI Elements' Functionalities (retrieved based on visual similarity):

  - 相似 UI 元素的功能 (基于视觉相似性检索):

“

similar element functionalities

类似元素的功能

Requirements for Functionality Description:

功能描述要求:

1. Concise: 1-2 sentences

1. 简洁:1-2 句

2. Focus on general function, not specific details

2. 侧重一般功能，不涉及具体细节

3. Avoid mentioning the numeric tag

3. 避免提及数字标签

4. Use generic terms like ”UI element” or appropriate pronouns

4. 使用通用术语如“界面元素”或适当的代词

5. Consider similar elements’ functionalities as reference, but prioritize:

5. 参考类似元素的功能，但优先考虑:

- Current screen context
  - 当前屏幕上下文
- Task description
  - 任务描述

6. Only incorporate relevant patterns from similar elements if they align with the current context Example:

6. 仅在与当前上下文一致时，采纳类似元素的相关模式示例:

- Incorrect: ”Tapping the element #3 displays David’s saved recipes in the results panel”
  - 错误:“点击元素 #3 会在结果面板显示 David 保存的食谱”
- Correct: ”Tapping this element will initiates a search and displays matching results”
  - 正确:“点击此元素将启动搜索并显示匹配结果”

Guidance:

指导:

- Describe the core action and potential result of interacting with the UI element
  - 描述与界面元素交互的核心动作及可能结果
- Infer functionality based on the current screen context
  - 根据当前屏幕上下文推断功能
- Prioritize clarity and generality in the description
  - 优先保证描述的清晰性和通用性
- Use similar elements’ functionalities to validate and refine your description, not to simply copy them
  - 利用相似元素的功能来验证和完善你的描述，而不是简单复制它们

J.3.2 Prompting Template of Dynamic Comprehension Task

J.3.2 动态理解任务的提示模板

Objective: Describe the functionality of a specific UI element in a mobile app screenshot. Input:

目标: 描述移动应用截图中特定 UI 元素的功能。输入:

- Two screenshots: Before and after interacting with a UI element

    - 两张截图: 交互前和交互后

- UI element marked with a numeric tag in the top-left corner

    - 左上角带有数字标签的 UI 元素

- Element number: numeric tag of element

    - 元素编号: 元素的数字标签

- Broader task context: task description

    - 更广泛的任务背景: 任务描述

- UI Element Attributes:

    - UI 元素属性:

ui element attributes

ui 元素属性

- Similar UI Elements' Functionalities (retrieved based on visual similarity):

    - 相似 UI 元素的功能 (基于视觉相似性检索):

similar element functionalities

相似元素功能

"

Requirements for Functionality Description:

功能描述要求:

2. Focus on general function, not specific details

2. 关注一般功能，不涉及具体细节

3. Avoid mentioning the numeric tag

3. 避免提及数字标签

4. Use generic terms like ”UI element” or appropriate pronouns

4. 使用通用术语如“UI 元素”或适当的代词

5. Consider similar elements’ functionalities as reference, but prioritize:

5. 参考相似元素的功能，但优先考虑:

- Current screen context
  - 当前屏幕上下文
- UI element attributes
  - 界面元素属性
- Task description
  - 任务描述

6. Only incorporate relevant patterns from similar elements if they align with the current context Example:

6. 仅当相似元素的相关模式与当前上下文一致时，才将其纳入参考示例:

- Incorrect: ”Tapping the element #3 displays David’s saved recipes in the results panel”
  - 错误示例:“点击元素 #3 会在结果面板中显示 David 保存的食谱”
- Correct: ”Tapping this element will initiates a search and displays matching results”
  - 正确示例:“点击此元素将启动搜索并显示匹配结果”

Guidance:

指导:

- Describe the core action and immediate result of interacting with the UI element
  - 描述与界面元素交互的核心操作及其即时结果
- Infer functionality based on the current screen context

- 根据当前屏幕上下文推断功能

- Prioritize clarity and generality in the description

- 优先保证描述的清晰性和通用性

- Use similar elements' functionalities to validate and refine your description, not to simply copy them

- 利用相似元素的功能来验证和完善描述，而非简单复制

J.4 Prompting Templates of Prior Knowledge Task for GUI-explorer

J.4 GUI-explorer 先验知识任务的提示模板

J.4.1 Prompting Template of Prior Knowledge Task

J.4.1 先验知识任务的提示模板

Objective: Describe the functionality of a specific UI element in a mobile app screenshot. Input:

目标: 描述移动应用截图中特定界面元素的功能。输入:

- One screenshot: Before interacting with a UI element

- 一张截图: 在与界面元素交互之前

- UI element marked with a numeric tag in the top-left corner

- 左上角带有数字标签的界面元素

- Element number: numeric tag of element

- 元素编号: 元素的数字标签

- Broader task context: task description

- 更广泛的任务背景: 任务描述

- UI Element Attributes:

- 界面元素属性:

ui element attributes

界面元素属性

- Similar UI Elements’ Functionalities (ranked by relevance to task description):

  - 类似界面元素的功能 (按与任务描述的相关性排序):

“

similar element functionalities

类似元素功能

“

Note: Elements are sorted by relevance, with most task-relevant functionalities listed first

注意: 元素按相关性排序，最相关的功能优先列出

Requirements for Functionality Description:

功能描述要求:

1. Concise: 1-2 sentences

1. 简洁:1-2 句

2. Focus on general function, not specific details

2. 侧重一般功能，不涉及具体细节

3. Avoid mentioning the numeric tag

3. 避免提及数字标签

4. Use generic terms like ”UI element” or appropriate pronouns

4. 使用通用术语如“界面元素”或适当代词

5. Consider similar elements’ functionalities as reference, with priority:

5. 参考类似元素功能，优先考虑:

- Higher-ranked (more relevant) reference functionalities

  - 排名更高 (更相关) 的参考功能

- Current screen context

  - 当前屏幕上下文

- UI element attributes

- 界面元素属性

- Task description

- 任务描述

6. Only incorporate relevant patterns from similar elements if they align with the current context Example:

6. 仅在与当前上下文一致时，才整合来自相似元素的相关模式示例:

- Incorrect: ”Tapping the element #3 displays David’s saved recipes in the results panel”

- 错误示例:“点击元素 #3 会在结果面板中显示 David 保存的食谱”

- Correct: ”Tapping this element will initiates a search and displays matching results”

- 正确示例:“点击此元素将启动搜索并显示匹配结果”

Guidance:

指导:

- Describe the core action and potential result of interacting with the UI element

- 描述与 UI 元素交互的核心操作及可能的结果

- Infer functionality based on the current screen context

- 根据当前屏幕上下文推断功能

- Prioritize clarity and generality in the description

- 优先保证描述的清晰性和通用性

- Pay special attention to higher-ranked similar functionalities as they are more likely to be relevant

- 特别关注排名较高的相似功能，因为它们更可能相关

- Use similar elements’ functionalities to validate and refine your description, not to simply copy them

- 利用相似元素的功能来验证和完善描述，而非简单复制

J.4.2 Prompting Template of Dynamic Comprehension Task

J.4.2 动态理解任务提示模板

Objective: Describe the functionality of a specific UI element in a mobile app screenshot.

目标: 描述移动应用截图中特定 UI 元素的功能。

Input:

输入:

- Two screenshots: Before and after interacting with a UI element

  - 两张截图: 交互前和交互后

- UI element marked with a numeric tag in the top-left corner

  - UI 元素左上角标有数字标签

- Element number: numeric tag of element

  - 元素编号: 元素的数字标签

- Broader task context: task description

  - 更广泛的任务背景: 任务描述

- UI Element Attributes:

  - 界面元素属性:

ccc

ccc

ui element attributes

界面元素属性

- Similar UI Elements' Functionalities (ranked by relevance to task description):

  - 类似界面元素的功能 (按与任务描述的相关性排序):

similar element functionalities

类似元素功能

Note: Elements are sorted by relevance, with most task-relevant functionalities listed first

注意: 元素按相关性排序，最相关的功能优先列出

Requirements for Functionality Description:

功能描述要求:

1. Concise: 1-2 sentences

1. 简洁:1-2 句

2. Focus on general function, not specific details

2. 关注一般功能，不涉及具体细节

3. Avoid mentioning the numeric tag

3. 避免提及数字标签

4. Use generic terms like ”UI element” or appropriate

4. 使用通用术语如“界面元素”或适当表达

5. Consider similar elements’ functionalities as

5. 考虑类似元素的功能作为

Higher-ranked (more relevant) reference functionalities

更高排名 (更相关) 的参考功能

- Current screen context

  - 当前屏幕上下文

- UI element attributes

  - 界面元素属性

- Task description

  - 任务描述

6. Only incorporate relevant patterns from similar elements if they align with the current context Example:

6. 仅在与当前上下文一致时，才整合来自相似元素的相关模式示例:

- Incorrect: ”Tapping the element #3 displays David’s saved recipes in the results panel”

  - 错误示例:“点击元素 #3 会在结果面板中显示 David 保存的食谱”

- Correct: ”Tapping this element will initiates a search and displays matching results”

  - 正确示例:“点击此元素将启动搜索并显示匹配结果”

Guidance:

指导:

- Describe the core action and potential result of interacting with the UI element

  - 描述与 UI 元素交互的核心操作及可能的结果

- Infer functionality based on the current screen context

  - 根据当前屏幕上下文推断功能

- Prioritize clarity and generality in the description

  - 优先保证描述的清晰性和通用性

- Pay special attention to higher-ranked similar functionalities as they are more likely to be relevant

  - 特别关注排名较高的相似功能，因为它们更可能相关

- Use similar elements’ functionalities to validate and refine your description, not to simply copy them

  - 利用相似元素的功能来验证和完善描述，而非简单复制