MOBA: Multifaceted Memory-Enhanced Adaptive Planning for Efficient Mobile Task Automation

Zichen Zhu, Hao Tang, Yansi Li, Dingye Liu, Hongshen Xu,
Kunyao Lan, Danyang Zhang, Yixuan Jiang, Hao Zhou, Chenrun Wang,
Situo Zhang, Liangtai Sun, Yixiao Wang, Yuheng Sun, Lu Chen*, Kai Yu*
X-LANCE Lab, Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence, SJTU AI Institute
Shanghai Jiao Tong University, Shanghai, China
{JamesZhutheThird,chenlusz,kai.yu}@sjtu.edu.cn

Abstract

Existing Multimodal Large Language Model (MLLM)-based agents face significant challenges in handling complex GUI (Graphical User Interface) interactions on devices. These challenges arise from the dynamic and structured nature of GUI environments, which integrate text, images, and spatial relationships, as well as the variability in action spaces across different pages and tasks. To address these limitations, we propose MOBA, a novel MLLM-based mobile assistant system. MOBA introduces an adaptive planning module that incorporates a reflection mechanism for error recovery and dynamically adjusts plans to align with the real environment contexts and action module's execution capacity. Additionally, a multifaceted memory module provides comprehensive memory support to enhance adaptability and efficiency. We also present MOBBENCH, a dataset designed for complex mobile interactions. Experimental results on MOBBENCH and AndroidArena demonstrate MOBA's ability to handle dynamic GUI environments and perform complex mobile tasks.

1 Introduction

Multimodal large language models (MLLMs) have seen significant advancements in recent years, supported by vast multimodal datasets. These models (Hu et al., 2024; Liu et al., 2024a; Ye et al., 2024, 2023; Chen et al., 2024b; Sun et al., 2024a; Liu et al., 2023; Dai et al., 2023; Chen et al., 2023; Zhu et al., 2024; Yao et al., 2024; OpenAI, 2023; Team, 2024) excel in tasks such as Chain-of-Thought (CoT) reasoning (Wei et al., 2022), In-Context Learning (ICL) (Brown et al., 2020), and various applications (Wang et al., 2024b; Wang and Zhao, 2023; Chen et al., 2024a; Liu et al., 2024b; Pan et al., 2024; Ge et al., 2024; Wu et al., 2024; Lee et al., 2024b; Qian et al., 2024b,a). Their capabilities have also enabled new MLLM-based agents for real-world tasks (Li et al., 2017, 2019; Sun et al., 2022; Zhu et al., 2023; Zhang and Zhang, 2024; Zhang et al., 2023a, 2024a; Nong et al., 2024; Ma et al., 2024; Wang et al., 2024a, 2025).

However, MLLMs face significant challenges when addressing complex GUI interactions and facing diverse user demands in real-world scenarios, particularly on devices such as smartphones (Zhang et al., 2024b) and computers (Cao et al., 2024; Xie et al., 2024). On the one hand, GUI environments are highly diverse and pose different action spaces across different apps and pages. For instance, the number and position of clickable icons can vary greatly across pages; some pages require text input, while others involve scrollable elements. Such variability makes proactive task planning hardly adapt to the real environment contexts and thus become infeasible to complete. On the other hand, the action executor can also lack capabilities enough to achieve it, even given a feasible task plan. In all these cases, agents with trivial or static planning (Zheng et al., 2024; Zhang et al., 2024a; Nong et al., 2024; Ma et al., 2024; Xing et al., 2024) will fail to align with the environment contexts and action executor's capacity and thus can fail the whole task easily caused by failure of a single sub-task. Furthermore, existing MLLM-based GUI agents (Zhang et al., 2023b,a; Wang et al., 2024a, 2025) often lack a powerful and comprehensive memory to face the need for dynamic planning at various levels and diverse user demands. These problems hinder the design of a practical mobile assistant.

To address these challenges, we propose MOBA, a novel MLLM-based mobile assistant system with

*Corresponding authors are Lu Chen and Kai Yu.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations), pages 535–549

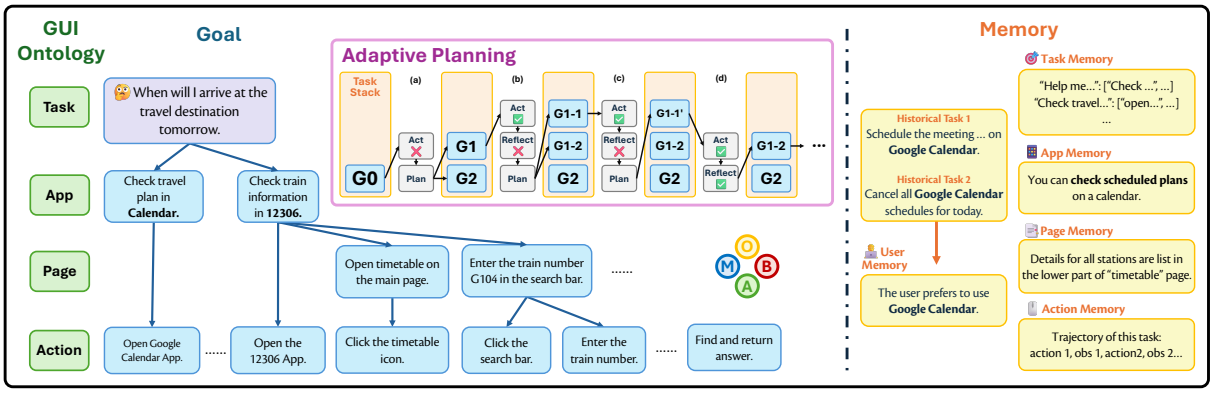

Figure 1: **The Illustration of adaptive planning and multifaceted memory structure.** There are 4 cases in adaptive planning: (a) Plan reflection failure, the goal needs to be decomposed. (b) In execution reflection failure, the goal needs to be decomposed. (c) Execution reflection failure, the goal needs to be refined. (d) Goal complete.

an adaptive planning module that dynamically adjusts task plans according to the execution results. As proactive planning often fails to accurately determine the actions required in a specific application or page or to align with the action executor's capacity, MOBA leverages reflection mechanism to recover task execution from failed sub-plans by reassessing goals or breaking tasks into more fine-grained sub-goals. To better support adaptive planning with various sub-goal granularity, a multifaceted memory module providing hierarchical memory support is proposed. We also introduce MOBBENCH, a diverse dataset for complex mobile interactions, and demonstrate MOBA's effectiveness on MOBBENCH and AndroidArena (Xing et al., 2024), showing its capability to handle dynamic GUI environments.

Our contributions are threefold:

- We propose an **adaptive planning** module that incorporates a reflection mechanism for error recovery and dynamically adjusts plans based on the current GUI environment and action executor's capacity.
- We develop a **multifaceted memory** module that provides hierarchical memory support to enhance task adaptability and efficiency.
- We introduce **MOBBENCH**, a diverse dataset for complex mobile interactions, and validate the effectiveness of our approach through extensive experiments on two datasets.

2 The MOBA System

The system overview of MOBA is shown in Figure 2. MOBA comprises a **Global Agent** (GA) and a **Local Agent** (LA). The Global Agent consists of a Plan Module and a Reflection Module. The

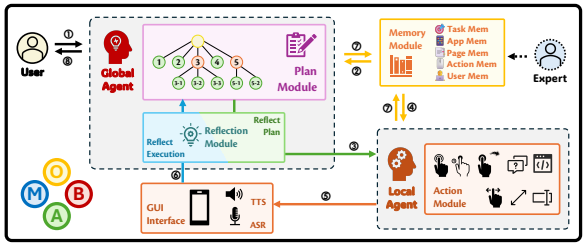

Figure 2: **System Overview of MOBA.**

Plan Module interprets the user's command (①) and resolves the task into several easier and clearer sub-tasks adaptively with the help of experiences in the multifaceted Memory Module (②), while the Reflection Module will review if the decomposition is feasible and sub-goals are achievable. Then, under the direction of a specific sub-goal (③), the Local Agent will leverage the experiences in the Memory Module (④), predict the concrete actions, and directly control the device (⑤). After LA's execution, the Reflection Module will reflect if the current sub-task has been completed (⑥) and the Plan Module can revise the plan accordingly. The Memory Module can also be updated after the invocation of the Plan Module and Local Agent (⑦) to improve MOBA's performance through execution. To ameliorate the performance at the early stage of the memory, it can also be initialized with a warm-up of some basic expert experiences. Finally, MOBA can generate a response to the user regarding the result of task execution (⑧). The remaining parts of this section will elaborate on the proposed adaptive plan module and multifaceted memory module.

2.1 Task Completion with Adaptive Planning

Facing the problem that static fixed-level task planning is deficient in aligning with real environment

contexts and the Action Module's capacity, we propose adaptive planning to react to concrete execution results of the Action Module and adjust the granularity of task decomposition adaptively. The proposed planning workflow is demonstrated in Algorithm 1. Given an established sub-goal, the reflection module is first adopted to review the sub-task feasibility. Then the Action Module will attempt to complete the reviewed sub-goal. The execution result will be inspected again by the Reflection Module. Once failure is detected, the Plan Module is invoked to revise the task plan to adapt to the current environment context or to further break the sub-goal down to match the Action Module's execution capacity. By repeating this procedure, MOBA can generate a multi-granularity task plan that well aligns with the environment contexts and the Action Module's capacity iteratively and dynamically.

Input: Global Agent GA, Local Agent LA, Goal G_0
task_stack.push(G_0)
while task_stack not empty **do**
  cur_task ← task_stack.pop()
  can_do ← GA.reflect_plan(cur_task)
  if can_do **then**
    action,obs← LA.exec_task(cur_task)
    cur_task_complete ←
     GA.reflect_exec(action,obs)
  end
  if not can_do **or not** cur_task_complete **then**
    new_subtasks ← GA.plan(cur_task)
    task_stack.push(new_subtasks)
  end
  GA.updateMemory()
end

Algorithm 1: Adaptive Planning of MOBA

2.2 Multifaceted Memory

The Memory Module serves as the backbone of MOBA's adaptability and learning capabilities, storing historical data to enhance decision-making and reduce redundant actions. It is categorized into five components:

Task Memory: Tracks the execution history of tasks, including task decomposition structures, action traces, success and failure records, and reflections. This hierarchical organization enables efficient retrieval of relevant experiences for task planning and execution.

App Memory: Maintains detailed observations and exploration histories for various applications, including functional descriptions and page-specific interactions. This helps the agent adapt to similar GUI layouts and locate target applications more effectively.

Page Memory: Encompasses the historical steps executed on this interface, such as the positioning of a particular button on the page, among other actions. This facilitates the agent's ability to perform similar operations on the page based on past interactions more effectively.

Action Memory: Incorporates the operations executed during the current task cycle, enabling the agent to more clearly capture the actions performed within this task and to more precisely define the subsequent steps required.

User Memory: Captures user-specific interaction histories, such as preferences, habitual commands, and implicit requirements. This allows MOBA to better infer user intent and personalize task execution.

3 Experiments

To comprehensively compare MOBA with other GUI agents in handling complex user instructions and executing GUI interactions on mobile devices, we evaluate them using a real-life scenario test set called MOBBENCH. Additionally, we assess our method using a widely adopted mobile benchmark, Android Arena.

3.1 The MOBBENCH Test Set

The MOBBENCH comprises a diverse test set of 50 tasks designed to evaluate the performance of MOBA in real-world mobile application scenarios. The test set includes 10 applications widely used in China, each with four tasks of varying difficulty: Easy, Medium, Hard, and Indirect Comprehension, totaling 40 tasks. The tasks are categorized by the complexity and steps required to complete them. Indirect Comprehension is designed for common cases where the user gives a vague instruction without detailing which application or specific steps are required. The agent is expected to decide target application and find an effective approach. Additionally, there are 10 Cross-Application tasks, which involve interacting with two applications and are more close to Hard level in difficulty. These tasks focus on evaluating the ability of information extraction and retrieval, as well as the awareness of sub-goal completion and application switching.

Compared with several similar task sets mentioned in other papers (Zhang et al., 2023a; Wang et al., 2024a, 2025; Zhang et al., 2024a; Lee et al.,

2024a), which only get a score when it finishes the task, we assign several milestone scores for sub-tasks in MOBBENCH. This allows for a more precise process assessment, in the cases where the task is partially finished. We also include a detailed preparation instruction for tasks when a more reproducible, fair, and stable start is needed.

To establish a human expert baseline, three human operators independently perform the tasks on three different mobile phones, documenting their execution steps. The average number of steps taken is used as the human expert baseline.

3.2 Metrics

Three metrics are designed to better compare the capability of GUI agents thoroughly.

Milestone Score (MS): Scoring milestones are assigned to several sub-tasks, evenly distributed during the task completion process. Since each task contains 1 to 6 milestones, the agent will get a score as it reaches each milestone. We sum up all milestone scores of 50 tasks as the primary metric.

Complete Rate (CR): If the agent gets all milestone scores in one task, it is considered as task complete. This is the most common and straightforward metric for GUI agent evaluation.

Execution Efficiency (EE): We record the effective number of steps for each task and the corresponding milestone scores, that is, the total number of steps executed at the time of getting the last effective milestone score, and calculate the average number of steps required to obtain each effective milestone score. The lower this number, the more efficient the execution; the higher it is, the more it includes ineffective actions.

The average milestone scores and execution steps for each task type are summarized in Table 1.

Task Type	# Tasks	# MS	Avg. Steps	EE
Easy	10	10	4.3	4.30
Medium	10	23	7.3	3.17
Hard	10	41	15.2	3.71
Indirect	10	28	9.4	3.36
Cross-App	10	31	10.8	3.48
Overall	50	133	9.4	3.53

Table 1: **Milestone scores and expert execution steps for different task types of MOBBENCH.**

3.3 Setups

To provide a comprehensive evaluation, MOBA is compared against several baselines from basic manual operations to several sophisticated agent-based automation.

Human Baseline as mentioned in § 3.1 are considered as the optimal solution for each task.

GPT-4o + Human Baseline utilizes an iterative process where the GPT model (OpenAI, 2023) provides guidance for manual task execution.

AppAgent (Zhang et al., 2023a) uses both view hierarchy and screenshot for planning and choosing target actions. All interactive elements are marked with bounding boxes and a unique index for better grounding performance.

Mobile Agent (v2) (Wang et al., 2024a, 2025) uses only visual information from screens as inputs. Target elements are selected with the guidance of OCR and CLIP (Radford et al., 2021) modules.

MOBA is evaluated under several settings by disabling the Memory Module or/and Plan Module to assess its performance and the impact of these two modules. We disable the Plan Module by replacing the Global Agent with a plain agent, and no sub-tasks are provided to the Action and Reflection Module. We disable the Memory Module by removing all in-context examples and historical experience information (including observations, thoughts, previous actions, and their execution status), focusing on assessing the core capability in zero-shot task execution.

All experiments are conducted using `gpt-4o-2024-05-13` API. The primary evaluation metric is the first attempt complete rate, directly measuring the effectiveness of each system in completing tasks on the first try without retries.

3.4 Results and Analysis

The overall experiment results are as listed in Table 2. And for more detailed results categorized by task type please refer to Figure 5.

Model	CR	MS	EE
Human	50/50	133	3.53
GPT-4o + Human	49/50	131 (98.5%)	3.82 (108.2%)
AppAgent	6/50	35 (26.3%)	4.43 (125.5%)
MobileAgent (v2)	17/50	63 (47.4%)	4.84 (137.1%)
MOBA w/o M & P	13/50	52 (39.1%)	4.42 (125.2%)
MOBA w/o P	15/50	65 (48.9%)	4.17 (118.1%)
MOBA w/o M	22/50	72 (54.1%)	3.81 (107.9%)
MOBA	28/50	88 (66.2%)	3.44 (97.5%)

Table 2: **Overall Performance on MOBBENCH.** M: Memory Module. P: Planning Module.

Table 2 shows the performance of four baselines.

Due to the complexity of mobile interfaces and the technical limitations encountered during task execution, the overall task completion rates (Complete Rate, CR) are relatively low for all agents. Consequently, the Milestone Score (MS) serves as a finer metric to more accurately reflect the performance of each agent by considering partial task completion. While there are notable differences in Milestone Scores among the baseline models, the gap in Execution Efficiency (EE) is less significant. This is because most agents can smoothly complete simpler sub-goals, whereas, for more complex sub-goals, the agents either complete them or fail entirely, resulting in closer performance regarding execution efficiency.

3.4.1 Performance Comparison

The performance of *MobileAgent* is notably higher than that of *AppAgent*. This improvement is mainly due to the inclusion of both Memory and Reflection modules in *MobileAgent*, which enhance reasoning capacity and utilize more computational resources, such as tokens. Additionally, *MobileAgent* keeps a record of all historical actions, allowing it to learn from the entire sequence of operations, whereas *AppAgent* can only track the most recent action. Furthermore, *MobileAgent* relies on OCR and CLIP modules for target localization, offering greater flexibility and avoiding the technical limitations that *AppAgent* faces when dependent on XML files. By adopting a twice-reflection strategy, the ineffective execution steps are slightly reduced, where the sub-tasks that are not able to be completed with a single action are decomposed finer before executed. This gives clearer guidance for the Local Agent to decide the target actions.

3.4.2 Ablation Study

The lower part of Table 2 presents the results of the ablation study, where we experimented with four different configurations by selectively enabling or disabling the Memory and Plan modules. The results indicate that incorporating both Memory and Plan modules significantly enhances the agent's overall performance.

The Plan module alone shows a much stronger effect than the Memory module alone, validating one of the core contributions of this paper—the effectiveness of task decomposition planning. By decomposing tasks into manageable sub-tasks, MOBA can perform global planning, avoid redundant actions, and minimize overlooked details, effectively managing its historical actions (since in a tree-structured task, previously completed subtasks are inherently tracked). Unlike *MobileAgent*, which focuses solely on the next specific action, MOBA first determines the next abstract task and then plans the specific execution steps, closely mirroring human reasoning patterns and providing a more structured approach.

When the Memory module is introduced, MOBA's performance further improves, particularly in cross-application tasks (see Figure 5 (b)). This enhancement is due to the Memory module's ability to retain crucial information over longer periods, such as `"the day I am traveling to Shenzhen"`, allowing it to reference previous screens' key content. In contrast, without the Memory module, the agent is limited to short-term memory of only the current and the immediately preceding steps, resulting in less effective task execution.

3.5 Results on Android Arena

Model	SR(single-app)	SR(cross-app)
GPT-3.5	0.449	0.048
GPT-4	0.759	0.571
MOBA(ours)	0.783	0.714

Table 3: The performance of LLMs and MOBA on the Android Arena dataset.

We also performed evaluations on Android Arena (Xing et al., 2024), comprising 157 single-app tasks and 21 cross-app tasks. As shown in Table 3, MOBA achieves success rates (SR) of 0.783 on single-app tasks and 0.714 on cross-app tasks, outperforming GPT-4 by 2.4% and 14.3%, respectively. The notable improvement in cross-app tasks is attributed to MOBA's subtask decomposition capability, which enables better app-switching decisions during tasks requiring more steps. Additionally, MOBA's reflection module encourages exploration, reducing repetitive actions and improving task success rates.

The Android Arena evaluation also highlights limitations in task completion judgment with GPT-4, with 11.8% of tasks being misclassified, compared to the results checked by humans. This is partly due to MOBA's tendency to execute redundant actions after completing tasks, complicating GPT-4's evaluation process. Despite this, MOBA's performance gains emphasize its strength in handling complex multi-step tasks, especially

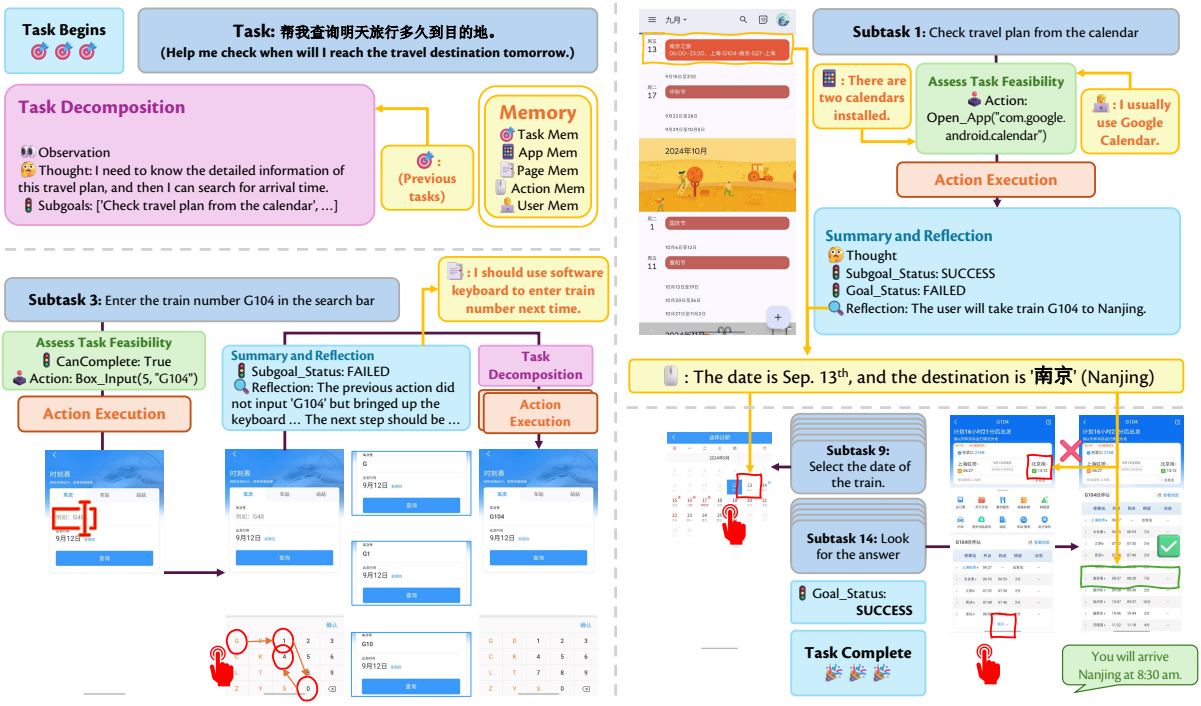

Figure 3: **The Example Case of MOBA.** Please note that several unimportant stages during the execution of a sub-task are omitted for clarity. The key features for each part are as follows. Task: MOBA supports cross-application tasks and can interpret indirect commands. Sub-task 1: Memories are retrieved to select target applications and updated to track the trace. Sub-task 3: MOBA will reflect and try other approaches if the attempt is failed. Sub-task 9 and sub-task 13: Memories are used to choose correct actions.

in scenarios requiring extensive exploration and app-switching, as evidenced by the significant improvements in cross-app success rates.

4 Case Study

Figure 3 demonstrates how the adaptive planning and multifaceted memory support task completion in MOBA. MOBA can accurately interpret user intent from command *"Help me check when will I reach the travel destination tomorrow."* and give decomposed sub-tasks based on historical commands. For sub-task 1, MOBA retrieves relevant details from App and User Memory, extracts key information (train schedule and destination), and stores it in Action Memory. When encountering failures, MOBA uses historical experiences to reflect and adapt. During sub-task 3, when MOBA initially failed to input the train number using the `Box_Input` function, it reflects on its previous operations and employs a character-by-character input method, completing the task. The key feature of this page will be saved into Page Memory, thus MOBA is unlikely to encounter the same failure. Additionally, memory retrieval is crucial for handling contextual tasks. In sub-tasks 9 and 13, although the user doesn't explicitly specify the travel date or destination in the task request. MOBA can rely on previously stored Action Memory data to provide an accurate response.

5 Related Work

5.1 LLM Agents

The advancements in M/LLMs have significantly influenced the development of agents. LLM-based agents leverage the autonomy, reactivity, proactiveness, and social ability of these models to perceive external environments and make decisions (Xi et al., 2023). Emerging abilities, such as CoT reasoning (Wei et al., 2022; Wang et al., 2023b; Zhang et al., 2023c) and in-context learning (Brown et al., 2020; Min et al., 2022). Recent studies have explored LLM-based approaches for reflection (Yao et al., 2023; Madaan et al., 2023; Shinn et al., 2023; Xu et al.), planning (Sun et al., 2024b; Qian et al., 2024c; Huang et al., 2024), and memory mechanisms (Zhang et al., 2024d,c; Li et al., 2023; Maharana et al., 2024; Lan et al., 2024).

At the same time, the agents that utilize M/LLMs to interact with the environments are quickly developed. These agents possess significantly en-

hanced capabilities for environment observation, task decomposition, and action decision-making, which enable M/LLMs to solve complex tasks across social simulations (Park et al., 2023; Aher et al., 2023; Jo et al., 2023; Lan et al., 2024), embodied robots (Wu et al., 2023), software development (Qian et al., 2024b,a) and virtual assistants (Wang et al., 2023a).

5.2 GUI Agents

5.2.1 Traditional GUI Agents

Controlling GUI screens based on user commands is a complex task that involves both GUI understanding and command interpretation. Early approaches to GUI agents focused on embedding and modular systems. For example, several agents (Li et al., 2017, 2019) combined natural language and programming demonstrations, allowing users to define tasks via descriptions and demonstrations. This method relied on text and image matching for script-based control of the interface. Traditional GUI agents were largely limited by their reliance on pre-defined rules and manual programming. These agents were effective within controlled environments but struggled with dynamic, real-world GUI contexts due to their lack of flexibility and adaptability. They required specific scripts or rules for each task, making them less robust when handling the diverse and evolving nature of real-world applications.

5.2.2 Advancements with Multimodal Pretrain Models

The advent of multimodal pretraining models (Bai et al., 2021; Li et al., 2021b; Li and Li, 2023; He et al., 2021; Li et al., 2021a; Wang et al., 2021; Fu et al., 2024) for GUI understanding marked a significant shift in the development of GUI agents. Pretrained agents (Sun et al., 2022; Zhu et al., 2023; Zhang and Zhang, 2024; Xu et al., 2024) integrated multimodal information, such as dialogue history, screenshots, and action history, through pretraining. Unlike earlier methods that relied on rigid scripts, these end-to-end models adopted a more human-like approach to interacting with interfaces, enhancing their efficiency in information retrieval and task execution by mapping visual observations and text commands directly into actions.

5.2.3 MLLM-Empowered GUI Agents

The integration of MLLMs in GUI agents has introduced new opportunities to further enhance their capabilities. With the rise of larger scale models, GUI agents (Zhang et al., 2023a, 2024a; Lee et al., 2024a) began to leverage advanced reasoning and decision-making processes. These models utilized structural information provided in the view hierarchy (VH) to annotate and locate UI elements, guiding a sequence of atomic actions to achieve specific goals. VH-only agents (Wen et al., 2024) depend on the structural information to reason and make decisions, which greatly lowers the cost of inference making it suitable for deployment on the device. Image-only agents (Wang et al., 2024a, 2025; Gao et al., 2024; Yan et al., 2023), which employs OCR, CLIP (Radford et al., 2021) module, and object detection methods to identify operation targets. This image-only approach is particularly effective when the view hierarchy is inaccessible or noisy, but it may also encounter challenges, e.g. opening a target application by clicking when names are hidden, or logos vary across screens.

6 Conclusion and Future Works

This paper presented MOBA, an innovative **Mob**ile phone **A**ssistant system empowered by MLLMs. Utilizing a two-level agent structure, comprising a Global Agent and a Local Agent, MOBA effectively understands user commands, plans tasks, and executes actions. The combination of Memory and Plan Modules enhances its ability to learn from previous interactions, improving efficiency and accuracy. Our evaluations demonstrated that MOBA surpasses existing mobile assistants in handling complex tasks, leveraging multi-level memory, task decomposition, and action-validation mechanisms. These features enable precise task execution even with intricate or indirect commands. Future work will focus on improving the performance on image-only scenarios where the view hierarchy is unattainable, deploying an end-side model on mobile phones for faster response and secured privacy. We will continue to expand MOBBENCH by adding more popular applications from different regions and languages. We hope MOBA illustrates the potential of MLLMs-empowered mobile assistants and provides valuable insights for future works.

Acknowledgment

This work is funded by the China NSFC Projects (92370206, 62120106006, U23B2057) and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

References

Gati Aher, Rosa I. Arriaga, and Adam Tauman Kalai. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In Proceedings of the 40th International Conference on Machine Learning, ICML'23. JMLR.org.

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Aguera y Arcas. 2021. Uibert: Learning generic multimodal representations for ui understanding. Preprint, arXiv:2107.13731.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc.

Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Wenjing Hu, Yuchen Mao, Tianbao Xie, Hongshen Xu, Danyang Zhang, Sida Wang, Ruoxi Sun, Pengcheng Yin, Caiming Xiong, Ansong Ni, Qian Liu, Victor Zhong, Lu Chen, Kai Yu, and Tao Yu. 2024. Spider2-v: How far are multimodal agents from automating data science and engineering workflows?

Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. Preprint, arXiv:2310.09478.

Zhanpeng Chen, Chengjin Xu, Yiyan Qi, and Jian Guo. 2024a. Mllm is a strong reranker: Advancing multimodal retrieval-augmented generation via knowledge-enhanced reranking and noise-injected training. Preprint, arXiv:2407.21439.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. 2024b. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 24185–24198.

Wenliang Dai, Junnan Li, DONGXU LI, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. In Advances in Neural Information Processing Systems, volume 36, pages 49250–49267. Curran Associates, Inc.

Jingwen Fu, Xiaoyi Zhang, Yuwang Wang, Wenjun Zeng, and Nanning Zheng. 2024. Understanding mobile gui: From pixel-words to screen-sentences. Neurocomputing, 601:128200.

Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. 2024. Assistgui: Task-oriented pc graphical user interface automation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13289–13298.

Yuying Ge, Sijie Zhao, Chen Li, Yixiao Ge, and Ying Shan. 2024. Seed-data-edit technical report: A hybrid dataset for instructional image editing. Preprint, arXiv:2405.04007.

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

Jinyi Hu, Yuan Yao, Chongyi Wang, SHAN WANG, Yinxu Pan, Qianyu Chen, Tianyu Yu, Hanghao Wu, Yue Zhao, Haoye Zhang, Xu Han, Yankai Lin, Jiao Xue, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. Large multilingual models pivot zero-shot multimodal learning across languages. In The Twelfth International Conference on Learning Representations.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. Preprint, arXiv:2402.02716.

Eunkyung Jo, Daniel A. Epstein, Hyunhoon Jung, and Young-Ho Kim. 2023. Understanding the benefits and challenges of deploying conversational ai leveraging large language models for public health intervention. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23, New York, NY, USA. Association for Computing Machinery.

Kunyao Lan, Bingui Jin, Zichen Zhu, Siyuan Chen, Shu Zhang, Kenny Q. Zhu, and Mengyue Wu. 2024. Depression diagnosis dialogue simulation: Self-improving psychiatrist with tertiary memory. Preprint, arXiv:2409.15084.

Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steven Y. Ko, Sangeun Oh, and Insik Shin. 2024a. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation. Preprint, arXiv:2312.03003.

Unggi Lee, Minji Jeon, Yunseo Lee, Gyuri Byun, Yoorim Son, Jaeyoon Shin, Hongkyu Ko, and Hyeoncheol Kim. 2024b. Llava-docent: Instruction tuning with multimodal large language model to support art appreciation education. Preprint, arXiv:2402.06264.

Gang Li and Yang Li. 2023. Spotlight: Mobile ui understanding using vision-language models with a focus. In The Eleventh International Conference on Learning Representations.

Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. Sugilite: Creating multimodal smartphone automation by demonstration. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI '17, page 6038–6049, New York, NY, USA. Association for Computing Machinery.

Toby Jia-Jun Li, Lindsay Popowski, Tom Mitchell, and Brad A Myers. 2021a. Screen2vec: Semantic embedding of gui screens and gui components. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, pages 1–15.

Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M Mitchell, and Brad A Myers. 2019. Pumice: A multi-modal agent that learns concepts and conditionals from natural language and demonstrations. In Proceedings of the 32nd annual ACM symposium on user interface software and technology, pages 577–589.

Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey Gritsenko. 2021b. Vut: Versatile ui transformer for multi-modal multi-task user interface modeling. Preprint, arXiv:2112.05692.

Yang Li, Yangyang Yu, Haohang Li, Zhi Chen, and Khaldoun Khashanah. 2023. Tradinggpt: Multi-agent system with layered memory and distinct characters for enhanced financial trading performance. Preprint, arXiv:2309.03736.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In Advances in Neural Information Processing Systems, volume 36, pages 34892–34916. Curran Associates, Inc.

Xinyu Liu, Yingqing He, Lanqing Guo, Xiang Li, Bu Jin, Peng Li, Yan Li, Chi-Min Chan, Qifeng Chen, Wei Xue, Wenhan Luo, Qifeng Liu, and Yike Guo. 2024b. Hiprompt: Tuning-free higher-resolution generation with hierarchical mllm prompts. Preprint, arXiv:2409.02919.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. CoCo-agent: A comprehensive cognitive MLLM agent for smartphone GUI automation. In Findings of the Association for Computational Linguistics ACL 2024, pages 9097–9110, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. Mobileflow: A multimodal llm for mobile gui agent. Preprint, arXiv:2407.04346.

OpenAI. 2023. Gpt-4v(ision) system card. https://openai.com/research/gpt-4v-system-card.

Xichen Pan, Li Dong, Shaohan Huang, Zhiliang Peng, Wenhu Chen, and Furu Wei. 2024. Kosmos-g: Generating images in context with multimodal large language models. In The Twelfth International Conference on Learning Representations.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST '23, New York, NY, USA. Association for Computing Machinery.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie, YiFei Wang, Weize Chen, Cheng Yang, Xin Cong, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. 2024a. Experiential co-learning of software-developing agents. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5628–5640, Bangkok, Thailand. Association for Computational Linguistics.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024b. ChatDev: Communicative agents for software development. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 15174–15186, Bangkok, Thailand. Association for Computational Linguistics.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. 2024c. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. Preprint, arXiv:2401.13996.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 8748–8763. PMLR.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In Advances in Neural Information Processing Systems, volume 36, pages 8634–8652. Curran Associates, Inc.

Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022. META-GUI: Towards multi-modal conversational agents on mobile GUI. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 6699–6712, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Quan Sun, Yufeng Cui, Xiaosong Zhang, Fan Zhang, Qiying Yu, Yueze Wang, Yongming Rao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. 2024a. Generative multimodal models are in-context learners.

Simeng Sun, Yang Liu, Shuohang Wang, Dan Iter, Chenguang Zhu, and Mohit Iyyer. 2024b. PEARL: Prompting large language models to plan and execute actions over long documents. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 469–486, St. Julian's, Malta. Association for Computational Linguistics.

Gemini Team. 2024. Gemini: A family of highly capable multimodal models. Preprint, arXiv:2312.11805.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile ui summarization with multimodal learning. In The 34th Annual ACM Symposium on User Interface Software and Technology, UIST '21, page 498–510, New York, NY, USA. Association for Computing Machinery.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. Preprint, arXiv:2305.16291.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2025. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. Advances in Neural Information Processing Systems, 37:2686–2710.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. Preprint, arXiv:2401.16158.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In The Eleventh International Conference on Learning Representations.

Yiqi Wang, Wentao Chen, Xiaotian Han, Xudong Lin, Haiteng Zhao, Yongfei Liu, Bohan Zhai, Jianbo Yuan, Quanzeng You, and Hongxia Yang. 2024b. Exploring the reasoning abilities of multimodal large language models (mllms): A comprehensive survey on emerging trends in multimodal reasoning. Preprint, arXiv:2401.06805.

Yuqing Wang and Yun Zhao. 2023. Gemini in reasoning: Unveiling commonsense in multimodal large language models. Preprint, arXiv:2312.17661.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems, volume 35, pages 24824–24837. Curran Associates, Inc.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android. In Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, ACM MobiCom '24, page 543–557, New York, NY, USA. Association for Computing Machinery.

Jiannan Wu, Muyan Zhong, Sen Xing, Zeqiang Lai, Zhaoyang Liu, Wenhai Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, Ping Luo, Yu Qiao, and Jifeng Dai. 2024. Visionllm v2: An end-to-end generalist multimodal large language model for hundreds of vision-language tasks. Preprint, arXiv:2406.08394.

Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. 2023. Embodied task planning with large language models. Preprint, arXiv:2307.01848.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. Preprint, arXiv:2309.07864.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Advances in Neural Information Processing Systems, volume 37, pages 52040–52094.

Mingzhe Xing, Rongkai Zhang, Hui Xue, Qi Chen, Fan Yang, and Zhen Xiao. 2024. Understanding the weakness of large language model agents within a complex android environment. pages 6061–6072.

Hongshen Xu, Lu Chen, Zihan Zhao, Da Ma, Ruisheng Cao, Zichen Zhu, and Kai Yu. 2024. Hierarchical multimodal pre-training for visually rich webpage understanding. In Proceedings of the 17th ACM International Conference on Web Search and Data Mining, WSDM '24, page 864–872, New York, NY, USA. Association for Computing Machinery.

Hongshen Xu, Zichen Zhu, Situo Zhang, Da Ma, Shuai Fan, Lu Chen, and Kai Yu. Rejection improves reliability: Training llms to refuse unknown questions using rl from knowledge feedback. In First Conference on Language Modeling.

An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, Zicheng Liu, and Lijuan Wang. 2023. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. Preprint, arXiv:2311.07562.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Minicpm-v: A gpt-4v level mllm on your phone. Preprint, arXiv:2408.01800.

Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, Chenliang Li, Yuanhong Xu, Hehong Chen, Junfeng Tian, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. 2024. mplug-owl: Modularization empowers large language models with multimodality. Preprint, arXiv:2304.14178.

Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Anwen Hu, Haowei Liu, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. 2023. mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration. Preprint, arXiv:2311.04257.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024a. Ufo: A ui-focused agent for windows os interaction. Preprint, arXiv:2402.07939.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023a. Appagent: Multimodal agents as smartphone users. Preprint, arXiv:2312.13771.

Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. 2023b. Large language models are semi-parametric reinforcement learning agents. In Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.

Danyang Zhang, Zhennan Shen, Rui Xie, Situo Zhang, Tianbao Xie, Zihan Zhao, Siyuan Chen, Lu Chen, Hongshen Xu, Ruisheng Cao, and Kai Yu. 2024b. Mobile-env: Building qualified evaluation benchmarks for llm-gui interaction. Preprint, arXiv:2305.08144.

Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. 2024c. Training language model agents without modifying language models. ICML'24.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024d. A survey on the memory mechanism of large language model based agents. Preprint, arXiv:2404.13501.

Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In Findings of the Association for Computational Linguistics ACL 2024, pages 3132–3149.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023c. Automatic chain of thought prompting in large language models. In The Eleventh International Conference on Learning Representations.

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, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2024. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In The Twelfth International Conference on Learning Representations.

Zichen Zhu, Liangtai Sun, Jingkai Yang, Yifan Peng, Weilin Zou, Ziyuan Li, Wutao Li, Lu Chen, Yingzi Ma, Danyang Zhang, et al. 2023. Camgui: A conversational assistant on mobile gui. In National Conference on Man-Machine Speech Communication, pages 302–315. Springer.

```
Input: xml file of the current screen
Output: the annotated screen
// First pass: Filter the small elements and all useless attributes
elements ← (sort(filter(elements), key=area)
selected_elements ← ∅
// Second pass: select elements whose overlapping area with former ones is small
foreach element in elements do
    if element is interactive then
        is_valid ← True foreach selected_element in selected_elements do
            if overlapping_area is large then
                is_valid ← False
            end
        end
        if is_valid is True then
            selected_elements ← selected_elements + element
        end
    end
end
// Third pass: Add the texts and merge the information of text into interactive elements
foreach element in elements do
    foreach selected_element in selected_elements do
        if element is contained in selected_element then
            merge(element, selected_element)
        end
    end
end
// Final pass: Sort the elements from left to right, top to bottom
Sort(elements, key=(y,x))
Plot all the interactive elements with their index
```

Algorithm 2: The Logic of View-Hierarchy Process Algorithm

A Several Useful Links

Code of MOBA:
https://github.com/OpenDFM/MobA
Prompts used in MOBA:
https://github.com/OpenDFM/MobA/blob/main/moba/prompts/prompts.py
Complete MOBBENCH:
https://huggingface.co/datasets/OpenDFM/MobA-MobBench

B View hierarchy processing

Given that (1) large models still exhibit limitations in processing visual information and (2) certain elements of the mobile phone interface cannot be obtained through visual means alone, the view hierarchy (VH) plays a crucial role in enabling agents to effectively interpret the mobile interface. However, the XML files representing mobile interfaces contain a substantial amount of redundant information. This redundancy increases token counts and complicates the agent's task of identifying key UI elements.

To address this issue, we developed an algorithm designed to filter UI elements. The algorithm consists of four steps: (1) parsing UI elements from the XML file, (2) filtering user-interactable UI elements based on their attributes, and adding them in ascending order of size, unless they exhibit significant overlap with previously added elements, (3) for UI elements containing text, merging the text content with interactive elements if the text is largely contained within those elements, thus enriching the interactive element with explanatory information, and (4) assigning an index to each UI element according to its central coordinates, from left to right and top to bottom, while plain text elements are assigned an index of -1. This ensures that the index ordering aligns more closely with the user's natural visual scanning behavior.

In summary, the core of our algorithm is the preservation of key interactive elements and their associated textual information, while minimizing occlusion in the image. For example, in the case of the "plane ticket" element demonstrated in Figure 4, the UI element itself does not contain text, and the text information associated with the plane ticket is non-clickable. By merging the two, the agent can infer that clicking the UI element corresponds to selecting the plane ticket.

However, limitations remain in this approach.

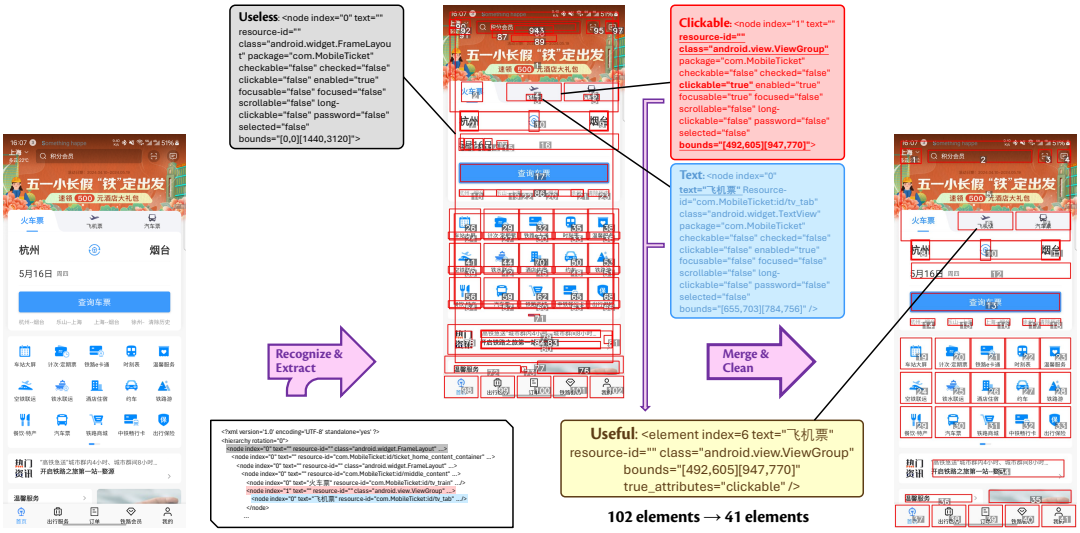

Figure 4: **An Example Diagram of View-Hierarchy Processing.** From left to right are the original image, unprocessed image and processed image. The underlined parts are the properties that are retained after the merge.

There are cases where all elements in the XML file are marked as "clickable=false", despite the presence of interactive elements in practice. Additionally, technical limitations sometimes prevent the XML file from accurately reflecting the current state of the interface.

C Action Space

We provide all actions supported in MOBA in Table 4.

D MOBBENCH

We provide five examples of the tasks included in MOBBENCH as shown in Table 5. You can get the complete collection of 50 tasks in both Chinese and English on Huggingface.

E Detailed Results Comparison

While the performance of all models is relatively similar on simpler tasks, MOBA demonstrates superior results in more challenging tasks, outperforming other models except for Human and GPT-4o + Human. This suggests that MOBA is more efficient in handling complex cases. Additionally, the incorporation of both the Memory Module and Plan Module enhances performance, highlighting their respective contributions to the system's overall capability.

E.1 Human is more adaptive and robust to screen interactions

While the human baseline is considered the optimal solution for each task, the *GPT-4o + Human* method achieves performance very close to that of human operators on all metrics. In the evaluation of *GPT-4o + Human*, the agent only provides textual task descriptions and an initial screenshot, and the GPT-4o generates detailed step-by-step instructions, which are then executed manually by a human operator.

The eye-catching performance of *GPT-4o + Human* can be attributed to several factors: (1) a relatively lenient standard in task execution, allowing human operators to interpret GPT-4o's general instructions flexibly; (2) human operators automatically completing tasks such as OCR, target detection, and localization, ensuring more precise actions; (3) GPT-4o provides a global plan, avoiding redundant or missed steps; (4) technical issues (e.g., inability to retrieve XML files or missing information in the files) do not affect task completion.

Action	Type	Usage	Description
Click	single	Click(element_index: int)	This function clicks the center of the UI element with the specified element index.
Click by Coordinate	single	Click_by_Coordinate(x: double, y: double)	This function simulates a click at the specified x and y coordinates on the screen.
Double Click	single	Double_Click(element_index: int)	This function double clicks the center of the UI element with the specified element index.
Long Press	single	Long_Press(element_index: int)	This function long-presses the center of the UI element with the specified element index.
Scroll	single	Scroll(element_index: int, direction: str, distance: str or int)	This function swipes from the center of the UI element with the specified element index.
Swipe	single	Swipe(direction: str, distance: str)	This function swipes from the center of the screen.
Type	single	Type(text: str)	This function inputs text on the current input box.
Back	single	Back()	This function presses the back key to return to the previous screen or status.
Box Input	combination	Box_Input(element_index: int, text: str)	This function clicks the input box, inputs given text, and confirms it.
Open App	system	Open_App(description: Optional[str])	This function locates and opens an app with a short description.
Close App	system	Close_App(package_name: Optional[str])	This function closes the specified app by its package name.
Error	system	Failed()	This function indicates that the task cannot be completed.
Finish	system	Finish()	This function indicates that the task is completed.

Table 4: **Available Actions and Descriptions**

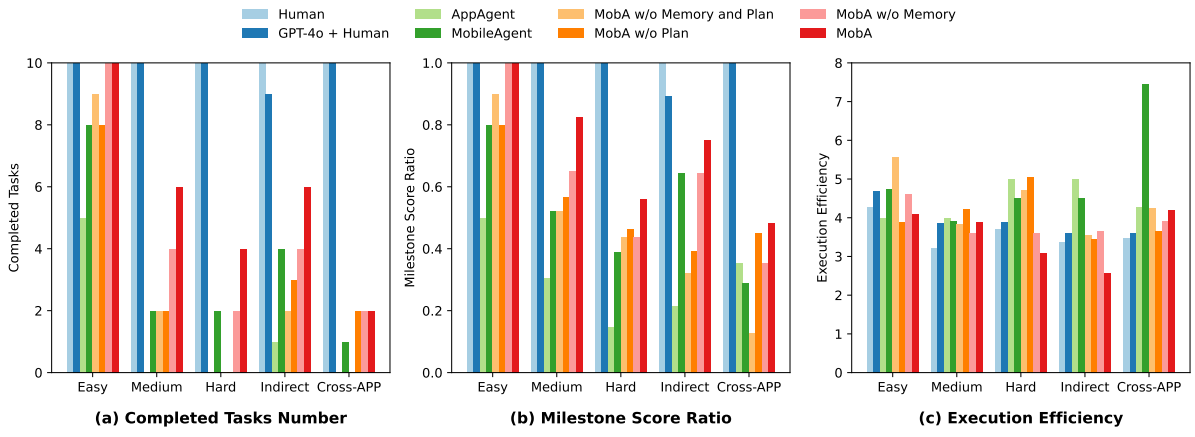

Figure 5: **Performance on MobBench Categorized by Task Type.**

Type	Application	Task	Preparation	Scoring Milestones	Steps
Easy	McDonald's	Switch the language of the McDonald's app to English.	Switch to Chinese.	1. Task completion.	6.7
Medium	12306 (China Railway)	Check the schedule for train G104 from Shanghai to Beijing tomorrow, and find out what time it is expected to arrive in Nanjing.	-	1. Enter the timetable screen, 2. Correct train number, 3. Task completion.	11.7
Hard	Douban	Search for the movie "The Shawshank Redemption" on Douban, mark it as "watched", rate it five stars, and leave a positive review.	Remove the previous mark, rating, and review of this movie.	1. Correct movie, 2. Correct mark, 3. Correct rating, 4. Positive review.	9.7
Indirect	BiliBili	If I'm out of mobile data, what videos can I still watch on the phone?	Download several videos in advance.	1. Open BiliBili, 2. Check downloads.	3.3
Cross-APP	JD.com, WeChat	Share the product link of the most recent JD.com order with a WeChat friend, and write a recommendation message.	There is an existing order.	1. Enter the order list, 2. Correct order, 3. Suitable message, 4. Task completion.	10.3

Table 5: **Several example tasks in MOBBENCH.** The content is translated from Chinese.