Enhancing UI Location Capabilities of Autonomous Agents

Anonymous ACL submission

Abstract

With the growing reliance on digital devices equipped with graphical user interfaces (GUIs), such as computers and smartphones, the need for effective automation tools has become increasingly important. While multimodal large language models (MLLMs) like GPT-4V excel in many areas, they struggle with GUI interactions, limiting their effectiveness in automating everyday tasks. In this paper, we introduce ClickAgent, a novel framework for building autonomous agents. In ClickAgent, the MLLM handles reasoning and action planning, while a separate UI location model (e.g., SeeClick) identifies the relevant UI elements on the screen. This approach addresses a key limitation of current-generation MLLMs: their difficulty in accurately locating UI elements. ClickAgent significantly outperforms other prompt-based autonomous agents (CogAgent, AppAgent) on the AITW benchmark. Our evaluation was conducted on both an Android smartphone emulator and an actual Android smartphone, using the task success rate as the key metric for measuring agent performance.

1 Introduction

Autonomous agents capable of interacting with graphical user interfaces (GUIs) are becoming critical for automating tasks on digital devices such as smartphones and computers (Kapoor et al., 2024). Researchers have begun developing agent-oriented large language models (LLMs) (Chen et al., 2023a; Zeng et al., 2023), but the potential of language-only agents is limited in real-world applications, where interaction with GUIs is often required. MLLMs and visual language models (VLMs) offer a promising solution to these limitations (You et al., 2023; Rahman et al., 2024; Gur et al., 2024; Baechler et al., 2024). Unlike language-based agents that rely solely on textual data such as HTML (Nakano et al., 2021) or OCR outputs (Rawles et al., 2023), MLLM-based agents directly interpret visual signals from GUIs. However, while current-generation MLLMs demonstrate reasonable abilities in screen understanding, reasoning, and action planning, they struggle to accurately locate specific UI elements on screens (Liu et al., 2024). Previous works (Yang et al., 2023; Fan et al., 2024; Ma et al., 2024) attempt to bypass this issue, for example by using an XML file that details the interactive elements or by using separate OCR model. Yet, such multi-module approaches are error-prone due to the inherent complexity of GUIs and the inconsistencies in XML/HTML files.

Our contribution lies in the development of ClickAgent, a hybrid autonomous agent that combines MLLM-driven reasoning with a specialized UI location model. Specifically, ClickAgent leverages the InternVL2.0 MLLM (Chen et al., 2023b) for reasoning and the TinyClick UI location model for identifying the coordinates of relevant UI elements (Pawlowski et al., 2024). This approach significantly advances performance, as demonstrated by our results on the commands from AITW benchmark (Rawles et al., 2024).

2 Method

Although models like SeeClick (Cheng et al., 2024) and Auto-UI (Zhan and Zhang, 2023) excel at identifying UI elements, they lack robust action planning, leading to low success rates in real-world smartphone tasks. To overcome these challenges, ClickAgent integrates InternVL2.0 for reasoning, while a dedicated UI location model identifies the exact coordinates of the target UI elements. The inputs to the UI location model are a screenshot and a natural language command corresponding to the desired UI element. ClickAgent's hybrid approach addresses the limitations of current MLLMs, which struggle to accurately locate UI elements (Liu et al.,

[1] The code for ClickAgent is included in the supplementary materials.

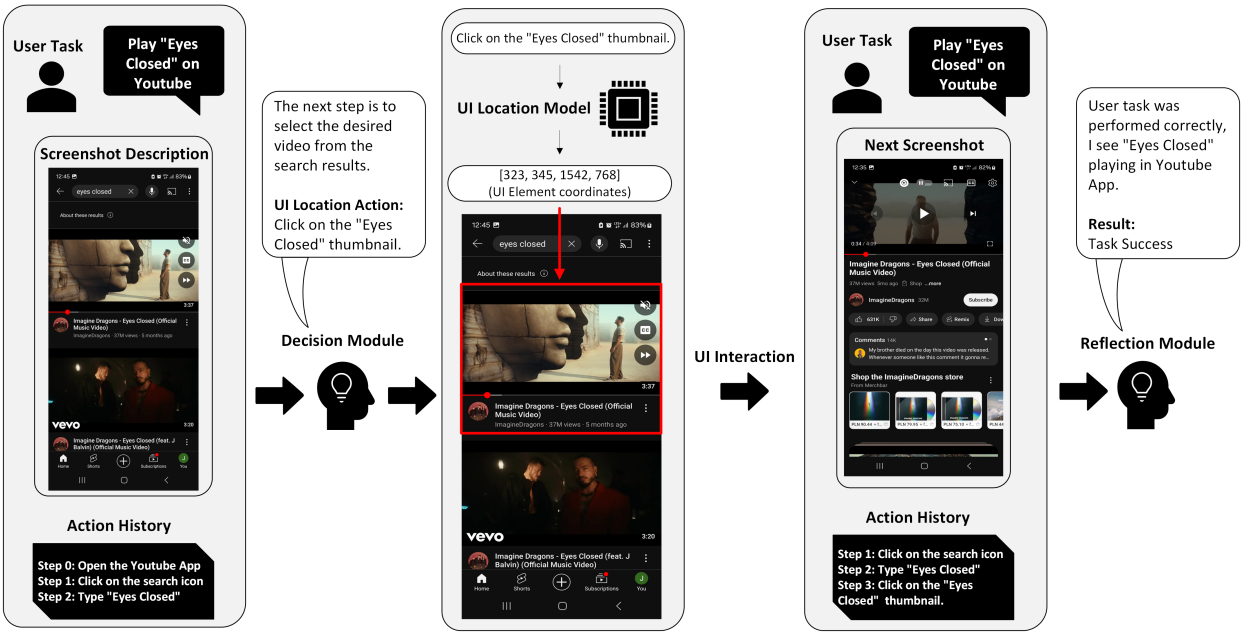

Figure 1: In ClickAgent, the MLLM is responsible for reasoning, reflection and action planning. In this example, the MLLM generates a UI command, and a specialized UI Location Model identifies the coordinates of the corresponding icon on the screen.

2024).

ClickAgent consists of three main components: Decision, UI location and Reflection. In the Decision module, the MLLM is asked to analyze the current screenshot, review the action history, and determine the next step to complete the user's task. The Decision module selects one of the predefined actions:

- `Click`: The MLLM generates a natural language command for the UI location model (e.g., "click on the Gmail icon."). The command and screenshot are passed to the UI location model, which returns the coordinates of the relevant UI element.
- `Type`: The MLLM generates the text to be typed into the text field.
- `OpenApp`: If this action is chosen, an additional query is made to the MLLM to select an app from the list retrieved from the Android device.
- `Swipe`: The agent swipes in the specified direction (up, down, left, or right).

For example, as illustrated in Figure 1, the MLLM chooses the Click action and issues a UI action command ("Click on the Eyes Closed Official Video") which, along with a screenshot, serves as input to the UI location model. The model then returns the bounding box coordinates of the relevant UI element.

After the action is executed on Android, the next screenshot is captured. In the Reflection module, the MLLM is asked to analyze content of screenshot and the entire action history. Reflection evaluates whether the user's task was successfully completed, returning either a "success" or "failure" status. If the task is marked as successful, the autonomous agent is stopped; otherwise, the agent proceeds with next decision and reflection cycle.

3 Evaluation Method

We evaluated ClickAgent on both an Android smartphone emulator and a real Android smartphone, using the task success rate (in percentage) as the primary performance metric. This metric assesses whether the agent successfully executed the user's task, making it the most critical measure in autonomous agent evaluation. Unlike other metrics, such as step success rate or action accuracy (Zhan and Zhang, 2023), the task success rate provides a clear and direct reflection of the agent's ability to accomplish user commands. Evaluation was performed manually due to imprecision of current automatic evaluation methods (Pan et al., 2024). We utilized 4 x NVIDIA A100 80GB GPU for running InternVL-2.0 and one NVIDIA A100 40GB GPU

	AITW General	AITW WebShopping	Overall
AppAgent (GPT-4V)	15.6	13.5	14.0
AutoUI	12.5	18.8	17.2
CogAgent	25.0	42.6	38.3
ClickAgent (ours)	72.5	**75.8**	**73.5**
+ *Android cache removal (ours)*	**73.1**	69.9	72.0

Table 1: Performance comparison of autonomous agents on the AITW General and AITW WebShopping benchmark subsets. Task success rates [%] were calculated based on manual evaluations. The main ClickAgent results were obtained using an actual Android smartphone, while the results with cache removal were performed on an Android smartphone emulator to simulate a first-time user experience.

for the UI location models (SeeClick, TinyClick). Each experiment was repeated three times, showing no significant deviation in accuracy across runs. We report the average of these three runs.

3.1 Test Environment

The tests were conducted on both an Android smartphone and an Android emulator under two scenarios: with and without cache removal. In the cache removal scenario, the emulator's cache was cleared before each test case, ensuring that popups and first-time user interactions appeared for each website, simulating a first-time user experience. In the no-cache removal scenario, conducted on the real Android smartphone, the cache was retained to mimic a user who had previously visited the websites, thereby minimizing or eliminating popups and other initial distractions.

We conducted an evaluation of ClickAgent on a subset of the AITW dataset (Rawles et al., 2023). In summary, our agent was evaluated on 154 unique webshop tasks and 432 general tasks. The AITW General consists of tasks related to interacting with everyday smartphone applications, while WebShopping focuses on tasks specific to e-commerce platforms.

4 Main Results

Table 1 presents the main results from the AITW benchmark. ClickAgent consistently outperforms other agents (AppAgent, Auto-UI, and CogAgent), achieving a significantly higher task success rate, regardless of whether the Android cache was cleared or not. As shown in Table 2, the accuracy of the UI location model plays a crucial role in determining the overall task success rate, making it a key factor in the ClickAgent's performance.

4.1 UI Location Model Analysis

Our primary insight is that TinyClick excels in OCR-related UI location. Therefore, we adjusted the prompt to encourage the MLLM to generate UI commands that incorporate textual information when possible. For instance, rather than producing commands like "Click on the first email," the MLLM is prompted to return more specific commands such as "Click on the email with the subject 'Meeting Agenda.'". This single prompt modification led to an improvement of around 10 percentage points in performance on the AITW benchmark.

4.2 ClickAgent Fails Analysis

On the AITW benchmark, the most common failures of ClickAgent were distributed across the following areas, with the percentages indicating the proportion of total errors attributed to each component:

- `Reflection Module` (47%): In some cases, the agent stops the action too late or too early, even though the task has not been completed.
- `UI Location Model` (15%): Some UI elements are unique to specific applications, and certain web pages have outdated designs, making it challenging for the UI Location Model to accurately identify desired elements.
- `Decision Module` (38%): Similar to the UI location model, most decision failures result from the MLLM's limited understanding of certain UIs and their functionalities.

Our experiments indicate that the performance of the Reflection and Decision is closely linked to the reasoning and general UI understanding capabilities of the MLLM. As more advanced MLLMs are developed, the performance of these modules should improve further.

	AITW General	AITW WebShopping
InternVL2-76B	0	0
SeeClick-9.6B	47.6	48.8
TinyClick-0.27B	72.5	75.8

Table 2: Performance comparison of ClickAgent using different UI location models on the AITW General and AITW WebShopping benchmarks (Task success rates [%]).

5 Ablation Study

We conduct an ablation study to understand the impact of two main components (MLLM and UI location model) on the overall performance of ClickAgent. Table 2 presents the evaluation of the UI location model's impact on ClickAgent's performance, comparing three different models. As expected, the MLLM (InternVL-2.0-76B) shows poor performance in the UI location task, resulting in ClickAgent failing all test cases. The most significant improvement comes from using the recently released TinyClick, which results in a substantially higher success rate than SeeClick.

Figure 2 illustrates the effect of MLLM general quality on ClickAgent's performance by evaluating four versions of InternVL-2.0 (1B, 7B, 26B, and 76B), alongside Qwen2.0-VL-72B (Wang et al., 2024). The results show that the quality of the MLLM plays a critical role in the ClickAgent's performance. Larger models, such as InternVL-2.0-76B, result in significantly higher success rates compared to the smaller variants. Further improvements in the MLLM quality should continue to enhance the ClickAgent's performance (especially in terms of the accuracy of the Reflection module).

6 Future Work

Our failure analysis reveals that the constantly changing nature of UIs presents a significant challenge for autonomous agents. As UIs change, both MLLMs and UI location models require continuous retraining to maintain their ability to recognize and interact with new UI elements across apps and websites (Chen et al., 2024; Gao et al., 2024). Another direction is the Retrieval-Augmented Generation (RAG) method introduced in AppAgent (Yang et al., 2023). In this approach, apps and websites are first explored automatically, generating detailed documentation of their functionalities. During inference, RAG retrieves relevant sections from this documentation, acting as a real-time reference manual for the agent. Both continuous retraining and RAG are promising avenues for enhancing ClickAgent. Combined, they could enable the agent to better adapt to quickly changing UIs and improve its performance on less commonly encountered graphical interfaces.

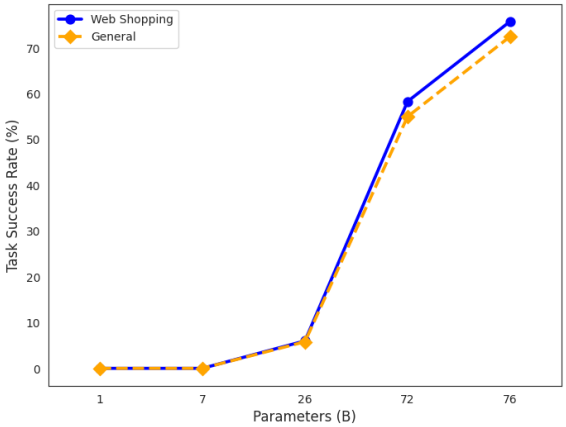

Figure 2: ClickAgent performance on the AITW General and AITW WebShopping benchmarks using different MLLMs. In all cases, the TinyClick model is employed for UI location.

7 Conclusion

In this paper, we introduced ClickAgent, a hybrid autonomous agent that combines MLLM-driven reasoning with a specialized UI location model. By addressing the limitations of previous approaches in identifying UI elements, ClickAgent demonstrates superior performance compared to baseline agents. The failures observed in the Reflection and Decision modules highlight the need for further advancements in MLLM capabilities, particularly in understanding UIs of less popular apps and websites.

Limitations

The primary limitation of ClickAgent is its high task completion time, averaging around 60 sec. per task on the AITW benchmark. Such latency is a common issue among prompt-based autonomous agents (Hong et al., 2023; Yang et al., 2023), largely because MLLM needs to generate detailed descriptions of screenshots. This prolonged inference time poses a challenge for latency-sensitive applications like voice assistants or other interactive AI systems. Figure 2 illustrates that while the smaller MLLMs have lower latency, their reasoning capabilities fall short of the requirements for building a high-quality

autonomous agent.

While alternative paradigms for building autonomous agents, such as Auto-UI, can reduce latency, they come with a significant trade-off in task success rates, as seen in Table 1. These non-MLLM-based approaches prioritize speed but fail to match the multi-turn reasoning capabilities of prompt-based agents.

Reduction of MLLM inference time can be achieved through advancements in model efficiency, caching mechanisms, or specialized hardware. For instance, the NVIDIA H100 offers up to a 3x speed improvement over the A100, increasing throughput from 20 to 60 tokens per second. FPGA solutions, such as DFX, can push performance even further to 76 tokens per second (Li et al., 2024). Another approach involves using real-time screen streaming (via MLLMs with video modalities) to take action instantly after screen changes (e.g., upon website load completion) (Durante et al., 2024). Additionally, a promising direction is to hide the MLLM's reasoning and planning (e.g., screen description) using techniques like Implicit Chain-of-Thought (Deng et al., 2024; Pfau et al., 2024).

Ethics

Autonomous agents can be misused for malicious purposes, such as misinformation bots or automated surveillance. Implementing regulatory safeguards and raising societal awareness are essential for mitigating negative impacts. Furthermore, it is crucial to strengthen the robustness of autonomous agents against adversarial attacks from malicious actors (Yang et al., 2024), so that benevolent agents are not manipulated into performing harmful actions.

References

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. In *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023a. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. 2024. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*.

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. 2023b. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Computer Vision and Pattern Recognition*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Annual Meeting of the Association for Computational Linguistics*.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*.

Zane Durante, Bidipta Sarkar, Ran Gong, Rohan Taori, Yusuke Noda, Paul Tang, Ehsan Adeli, Shrinidhi Kowshika Lakshmikanth, Kevin Schulman, Arnold Milstein, et al. 2024. An interactive agent foundation model. *arXiv preprint arXiv:2402.05929*.

Yue Fan, Lei Ding, Ching-Chen Kuo, Shan Jiang, Yang Zhao, Xinze Guan, Jie Yang, Yi Zhang, and Xin Eric Wang. 2024. Read anywhere pointed: Layout-aware gui screen reading with tree-of-lens grounding. *arXiv preprint arXiv:2406.19263*.

Yuan Gao, Kunyu Shi, Pengkai Zhu, Edouard Belval, Oren Nuriel, Srikar Appalaraju, Shabnam Ghadar, Zhuowen Tu, Vijay Mahadevan, and Stefano Soatto. 2024. Enhancing vision-language pre-training with rich supervisions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13480–13491.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A real-world webagent with planning, long context understanding, and program synthesis. *Preprint*, arXiv:2307.12856.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2023. Cogagent: A visual language model for gui agents. In *Computer Vision and Pattern Recognition*.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakhutdinov. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. *arXiv preprint arXiv:2402.17553*.

Jinhao Li, Jiaming Xu, Shan Huang, Yonghua Chen, Wen Li, Jun Liu, Yaoxiu Lian, Jiayi Pan, Li Ding, Hao Zhou, et al. 2024. Large language model inference acceleration: A comprehensive hardware perspective. *arXiv preprint arXiv:2410.04466*.

Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. 2024. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *arXiv preprint arXiv:2404.05955*.

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.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024. Autonomous evaluation and refinement of digital agents. In *First Conference on Language Modeling*.

Pawel Pawlowski, Krystian Zawistowski, Wojciech Lapacz, Adam Wiacek, Marcin Skorupa, Sebastien Postansque, and Jakub Hoscilowicz. 2024. Tinyclick: Single-turn agent for empowering gui automation.

Jacob Pfau, William Merrill, and Samuel R Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*.

Abdur Rahman, Rajat Chawla, Muskaan Kumar, Arkajit Datta, Adarsh Jha, Mukunda NS, and Ishaan Bhola. 2024. V-zen: Efficient gui understanding and precise grounding with a novel multimodal llm. *Preprint*, arXiv:2405.15341.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Android in the wild: a large-scale dataset for android device control. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 59708–59728.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao 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. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Yulong Yang, Xinshan Yang, Shuaidong Li, Chenhao Lin, Zhengyu Zhao, Chao Shen, and Tianwei Zhang. 2024. Security matrix for multimodal agents on mobile devices: A systematic and proof of concept study. *arXiv preprint arXiv:2407.09295*.

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

Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. 2023. Ferret: Refer and ground anything anywhere at any granularity. *Preprint*, arXiv:2310.07704.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

Zhuosheng Zhan and Aston Zhang. 2023. You only look at screens: Multimodal chain-of-action agents. In *Annual Meeting of the Association for Computational Linguistics*.