A3: Android Agent Arena for Mobile GUI Agents with Essential-State Procedural Evaluation

Yuxiang Chai[1,2*], Shunye Tang[2,3*], Han Xiao[1,2], Weifeng Lin[1,2], Hanhao Li[6], Jiayu Zhang[6], Liang Liu[2†], Pengxiang Zhao[2], Guangyi Liu[2], Guozhi Wang[2], Shuai Ren[2], Rongduo Han[3], Haining Zhang[3✉], Siyuan Huang[7], Hongsheng Li[1,4,5✉]

[*]Equal contribution, [†]Project Lead, [✉]Corresponding author

[1]CUHK MMLab, [2]vivo AI Lab, [3]SEAMiLab @ NKU, [4]Shenzhen Loop Area Institute, [5]CPII under InnoHK, [6]CUHK, [7]SJTU

https://github.com/YuxiangChai/AITK

Abstract

The advancement of Large Language Models (LLMs) and Multimodal Large Language Models (MLLMs) has catalyzed the development of mobile graphic user interface (GUI) AI agents, which is designed to autonomously perform tasks on mobile devices. However, a significant gap persists in mobile GUI agent evaluation, where existing benchmarks predominantly rely on either static frame assessments such as AndroidControl or offline static apps such as AndroidWorld and thus fail to capture agent performance in dynamic, real-world online mobile apps. To address this gap, we present Android Agent Arena (A3), a novel "essential-state" based procedural evaluation system for mobile GUI agents. A3 introduces a benchmark of 100 tasks derived from 20 widely-used, dynamic online apps across 20 categories from the Google Play Store, ensuring evaluation comprehension. A3 also presents a novel "essential-state" based procedural evaluation method that leverages MLLMs as reward models to progressively verify task completion and process achievement. This evaluation approach address the limitations of traditional function based evaluation methods on online dynamic apps. Furthermore, A3 includes a toolkit to streamline Android device interaction, reset online environment and apps and facilitate data collection from both human and agent demonstrations. The complete A3 system, including the benchmark and tools, will be publicly released to provide a robust foundation for future research and development in mobile GUI agents.

1 Introduction

The rapid evolution of Large Language Models (LLMs) and Multimodal Large Language Models (MLLMs) has acted as a primary catalyst for innovation in autonomous AI agents. Within this domain, Graphical User Interface (GUI) agents represent a critical area of research. Unlike systems that interact with applications via APIs or textual representations (e.g., HTML and XML), GUI agents operate through a fundamentally visual modality, executing tasks by directly processing screen pixel information. This paper focuses on mobile GUI agents, which are specialized for the interaction paradigms of the mobile ecosystem. These agents are becoming increasingly prevalent due to their potential to autonomously execute user commands, thereby streamlining workflows and minimizing human intervention.

While research of mobile GUI agents has advanced in modeling, evaluation remains a persistent challenge. The inherently sequential nature of tasks makes ascertaining task success difficult. Pioneering benchmarks (Chai et al., 2025; Rawles et al., 2023; Li et al., 2024) address this via static frame-step evaluation, assessing an agent's ability to predict the next action from a single static screenshot. Although this successfully measures single-step accuracy, it fails to simulate real-world fidelity, where it cannot account for dynamic state changes or the cascading effects of early errors that often lead to task failure. Furthermore, static evaluation penalizes valid alternative trajectories. For instance, launching an app via the library versus using ADB commands are both correct but static evaluation would treat one of them as failure.

To overcome the limitations of static assessment, the field has shifted toward dynamic evaluation systems using interactive devices. A prominent example, AndroidWorld (Rawles et al., 2025), evaluates agents by instrumenting the source code of open-source apps to verify internal states in an interactive virtual device. However, this reliance on open-source software creates a significant gap in ecological validity: widely used closed-source applications in categories such as shopping, travel, news, ticketing, etc. are excluded. Other benchmarks (Xu et al., 2025b; Lee et al., 2025) share the same shortcomings. Consequently, agents are rarely tested on the apps users interact with most.

Findings of the Association for Computational Linguistics: ACL 2026, pages 3774–3789

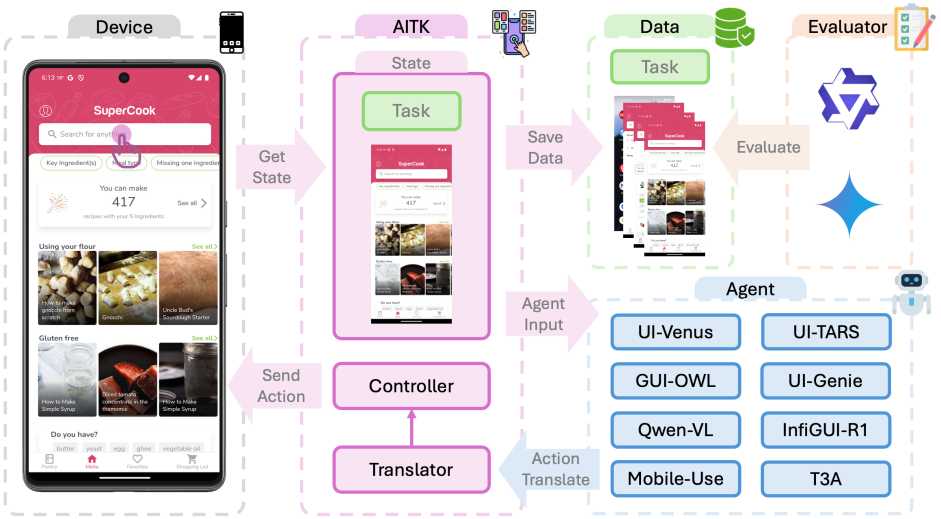

Figure 1: A3 consists of AITK (Android Interaction Toolkit), agents and evaluators. AITK has a translator to convert agent actions to a unified action space and a controller to interact with the device. AITK also gets the state of device at each step and save the data. After the task execution, evaluator output the evaluation results.

Moreover, some dynamic frameworks (Chen et al., 2024b; Xing et al., 2024) suffer from environmental instability, difficult state resets, and high reliance on manual human evaluation, leading to scalability issues and potential inaccuracies.

To address these shortcomings, we propose the Android Agent Arena (A3). Our evaluation system introduces a comprehensive benchmark comprising 100 daily-life tasks derived from 20 popular applications spanning 20 categories in the Google Play Store's top charts. Crucially, A3 incorporates dynamic and online apps previously deemed untestable due to the constraints of function-based evaluation, such as news, navigation, travel, email, shopping, etc. Because the internal states of these constantly updating proprietary apps cannot be instrumented, we introduce a novel **essential-state based procedural evaluation method**. This approach leverages MLLMs, either large commercial models or samller fine-tuned open-source alternatives, as reward models to autonomously and progressively determine task success and essential state achievements. This methodology not only verifies final objectives but also evaluates intermediate progress even when the total task is not completed, significantly reducing manual labor while maintaining high evaluation reliability on dynamic online apps and tasks. A3 pipeline is demonstrated in Figure 1.

Our contributions are summarized as follows:

- We introduce the Android Agent Arena (A3), a benchmark featuring 100 common tasks derived from 20 popular, dynamic online apps. This enables the robust evaluation of agent performance in complex, real-world scenarios that were previously difficult to assess.
- We propose a novel essential-state procedural evaluation methodology that utilizes MLLMs to progressively verify task success. We demonstrate that both large commercial models and our fine-tuned lightweight alternatives (A3RM) can serve as effective reward models to assess intermediate progress and final objectives.
- We release the complete pipeline and tools to accelerate research in the field. This includes modules for streamlined agent execution, trajectory data collection (for both agents and humans), and a versatile evaluator module designed for easy customization and community adoption.

2 Related Work

2.1 GUI Agents

Recent advancements increasingly leverage the world knowledge and reasoning capabilities of modern MLLMs for GUI control tasks, aiming to build more general and autonomous interactive agents (Liu et al., 2025a; Wang et al., 2025; Hu et al., 2025). Existing approaches can be broadly grouped into two categories. The first line finetunes a single general-purpose base MLLM (Bai et al., 2025b) as the GUI agent, relying on their unified visual-textual understanding to directly plan and execute actions in diverse interfaces (Qin et al.,

Name	Eval Mode	# Tasks	# General Apps	Operation	Inf. Query	Online
AITW	static	-	-	✓	✗	✗
AndroidControl	static	-	-	✓	✗	✗
AMEX	static	-	-	✓	✗	✗
GUI-Odyssey	static	-	-	✓	✗	✗
AndroidArena	dynamic	221	4	✓	✗	✗
Mobile-Env	dynamic	74	5	✓	✗	✗
AndroidWorld	dynamic	116	15	✓	✓	✗
B-Moca	dynamic	131	4	✓	✗	✗
AndroidLab	dynamic	138	5	✓	✓	✗
SPA-bench	dynamic	170	20	✓	✗	✓
A3 (Our)	dynamic	100	20	✓	✓	✓

Table 1: GUI related datasets and benchmarks in English. The top four rows are GUI agent related datasets, which provide static frame evaluation. The middle six rows are dynamic evaluation systems, which provide different tasks from different apps in different settings. AndroidWorld provides 15 generals apps from non-mainstream open-source F-Droid. SPA-bench provides another 20 apps and 170 tasks in Chinese.

2025; Gu et al., 2025; Liu et al., 2025c; Xiao et al., 2025, 2026). The second line comprises framework style systems like Mobile-Use (Li et al., 2025) that integrate GUI-specific perception or UI-tree modules, multi-agent planning or shortcut guidance to improve robustness and task efficiency.

2.2 GUI Agent Benchmarks

Early evaluations of GUI agents rely on static benchmarks where agents predict the next action from a single screenshot. Prior works such as AITW (Rawles et al., 2023), AMEX (Chai et al., 2025), and AndroidControl (Li et al., 2024) focus on single-step action accuracy via element or coordinate matching, but fail to capture sequential decision-making in dynamic environments. ColorBench (Song et al., 2025) and Mobile-Bench-V2 (Xu et al., 2025a) partially address this by introducing multi-step, multi-path tasks, yet remain limited in modeling fully interactive settings, motivating our dynamic evaluation framework.

However, existing dynamic benchmarks exhibit constraints in their task and application design. Several systems are restricted by task simplicity and diversity, such as Mobile-Env (Zhang et al., 2023) and B-Moca (Lee et al., 2025). Others are constrained by their app selection: AndroidArena (Xing et al., 2024) focuses on system apps, failing to evaluate generalization to the third-party app ecosystem, while prominent benchmarks like AndroidWorld (Rawles et al., 2025) and AndroidLab (Xu et al., 2025b) are restricted to open-source and offline apps. ProBench (Yang et al., 2026) improves evaluation by introducing more fine-grained and accurate success metrics, yet still operates within relatively constrained environments and limited app diversity. They exclude common real-world stochastic events such as pop-up ads and dynamic content updates and also omit online dynamic apps such as shopping, travel, and news, which are common in real-life scenarios. While SPA-bench (Chen et al., 2024b) includes online apps, it suffers from instability and unreliable system resets. More fundamentally, many benchmarks rely on rigid evaluation protocols: matching predefined answers (Xu et al., 2025b) or exact states (Rawles et al., 2025), which fail to capture the nuances of task completion in dynamic environments. This highlights the need for ecologically valid tasks and more flexible, semantically-aware evaluation frameworks. The overall statistics of the existing benchmarks are listed in Table 1.

3 Android Agent Arena (A3)

3.1 Apps & Tasks

Existing benchmarks exhibit significant limitations regarding ecological validity and reproducibility. Many rely on offline, open-source applications with restricted functionality (Rawles et al., 2025; Xu et al., 2025b). While these environments enable stable testing, they prioritize low-complexity system utilities, such as starting a timer or starting a voice recording which are often trivially solvable by API-based assistants (e.g., Siri), or drawing and moving boxes on a static HTML canvas which users would never ask agents to do in the real-life. In contrast, we consider that robust mobile GUI agents should address user intents in common apps with in-the-wild tasks, such as playing a tutorial video when

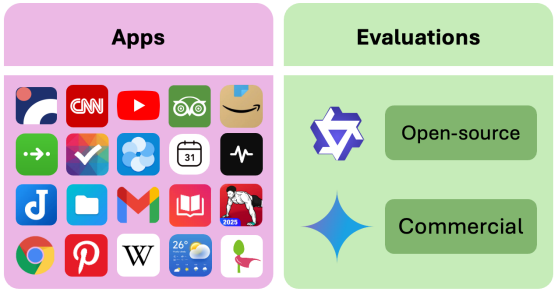

Figure 2: Overview of A3 apps and evaluations. The benchmark features tasks from 20 popular mobile applications and the framework supports essential-state based evaluation by two MLLM-based methods, utilizing either a commercial or an open-source finetuned model.

cooking, searching for the cheapest flight or finding the fast route when navigating. While Chen et al. (2024b) attempts to incorporate online apps, it fails to provide reproducible test environments or automated device-reset mechanisms. This lack of standardization necessitates extensive human intervention for initialization and cleanup, causing large-scale, reproducible testing prohibitive.

To address these challenges, we adopted a systematic top-down curation strategy. First, we identified 20 distinct categories from the Google Play Store top charts to ensure broad domain coverage. Within each category, we evaluated multiple candidate applications using a rigorous filtering protocol to select one representative app. Our final selection of 20 applications, averaging 115 million downloads each (see Figure 2 and Appendix A.1), was guided by a critical technical criterion: feasibility for automated state management, specifically requiring minimal forced logins and supporting reliable environment resets. This curation resolves the fundamental trade-off between ecological validity and experimental reproducibility. A3 establishes a unique framework that supports dynamic, online apps while maintaining reliable, programmatic resets to a consistent initial distribution, enabling the robust evaluation of agents within a stable experimental setting.

Building on this foundation, we designed 100 representative tasks aligned with the core functionalities of the selected applications (e.g., restricting Amazon tasks to shopping and CNN tasks to news retrieval). To provide a structured evaluation, tasks are classified along two primary axes. Based on their objective, they are designated as either Operation (requiring a sequence of state-changing actions) or Information Query (which additionally requires the agent to extract information to answer a question). The difficulty of each task is quantitatively defined by the number of steps (N) required by a human expert: Easy ($N < 7$), Medium ($7 \leq N \leq 11$), and Hard ($N > 11$).

3.2 Essential-State Evaluation

Existing evaluation methodologies for mobile agents often rely on metrics that lack sufficient granularity and flexibility for in-depth analysis. The widely-used AndroidWorld benchmark (Rawles et al., 2025), for instance, primarily employs a binary Success Rate (SR), calculated as the ratio of successful tasks to the total ($N_{success}/N_{total}$). While straightforward, this coarse-grained metric provides no insight into partial progress or specific failure modes, and it cannot differentiate the capabilities of agents that achieve the similar final score. To address this, AndroidLab (Xu et al., 2025b) introduced an additional, more granular sub-goal success rate. While this approach offers a more detailed view by tracking intermediate steps, its implementation has significant limitations. The sub-goals are pre-defined in a rigid JSON format (e.g., *Phone: 12345678*) for the evaluation function and are only applicable to certain operation tasks.

To address the need for a granular yet flexible metric, we introduce the core of A3's framework: the essential-state evaluation method. We define an essential state as a critical, observable semantic milestone that must be achieved for a task to be considered successful. Instead of assessing a binary final outcome, our method decomposes the execution trajectory into a sequence of these verifiable states. This approach differs fundamentally from the rigid sub-goals employed in systems like AndroidLab (Xu et al., 2025b). While AndroidLab sub-goals are typically bound to specific internal element key-value pairs (e.g., exact view IDs or string matches), essential states are defined by higher-level semantic outcomes. This abstraction makes our metric and evaluation resilient to UI variations and diverse execution paths, which is crucial for dynamic apps and open-ended information query tasks. For example, consider the task: *"Search 'marvel comics' in Pinterest. Who is the author of the first pin?"* We define three essential states: (1) The query 'marvel comics' is searched; (2) The first pin is selected; and (3) The author of the pin is identified. Crucially, because

these states are defined by what is achieved rather than how the underlying code renders it, they remain invariant even as app content dynamically refreshes or evolves. A detailed list of tasks and their corresponding essential states is provided in Appendix A.2.

Defining such semantic milestones in human-centric tasks inherently involves qualitative judgment. To ensure rigor and mitigate annotator bias, we established a strict three-stage, human-in-the-loop protocol for defining these states:

- **Trajectory Diversity:** Two human operators independently complete the same task using distinct strategies to generate diverse successful trajectories.
- **Collaborative Definition:** Based on the collected trajectories, the operators collaboratively propose a set of essential states. These states must meet three criteria: (1) *Visual Verifiability* (identifiable from consecutive screenshots and actions); (2) *Criticality* (representing a "must-be-done" step); and (3) *Sufficiency* (the set must cover the entire task logic).
- **Independent Audit:** A human evaluator audits the proposed states to verify they are logical, comprehensive, and achievable across different valid interaction methods.

We acknowledge that defining essential states incurs an initial manual design cost. However, this represents a strictly one-time investment. As established, essential states capture high-level task logic rather than transient UI elements or specific pixel coordinates. Consequently, these states remain invariant as long as the core task objective is unchanged, effectively decoupling the evaluation logic from the dynamic content refreshes inherent to online applications. This stability ensures that the benchmark remains valid over time without frequent re-annotation, offering a long-term evaluation reliability.

For essential-state evaluation metric, we define $ESAR = N_{AES}/N_{TES}$, where $ESAR$ is the **E**ssential-**S**tate **A**chieved **R**ate, N_{AES} is the number of achieved essential-states and N_{TES} is the total number of essential-states. Overall task success is then determined by the successful completion of all its associated essential states. We formalize this as:

$$A_i = \begin{cases} 1, & \text{if } eval(ES_j) = 1 \\ & \quad \text{for } \forall j \in [0, ..., N_{ES}] \\ 0, & \text{otherwise} \end{cases}$$

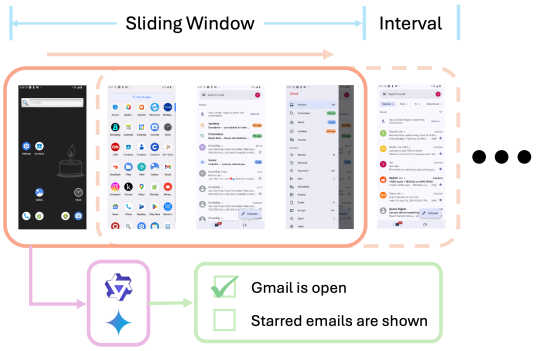

Figure 3: Illustration of the sliding window mechanism. A window of a predefined size and interval traverses the trajectory. At each position, the screen frames within the window are passed to a judge. The judge then assesses whether any of the task's essential states have been achieved within that segment. This process repeats until the entire trajectory has been evaluated.

where A_i is the success status of task i, ES_j is the j-th essential-state for the task, N_{ES} is the total number of the essential-states for task i and $eval(\cdot)$ is the method that verifies the achievement of an essential-state. Thus, the complex problem of end-to-end task evaluation is simplified into a series of more manageable essential-state assessments.

To implement the essential-state evaluation, we process the agent's full interaction trajectory using a sliding window mechanism. A window of a predefined size traverses the sequence of screen frames from start to finish, advancing at a specified interval. At each position, the contents of the window are submitted to our MLLM-based evaluator, which judges whether any of the essential-states are satisfied within that specific segment. This evaluation process is illustrated in Figure 3.

3.3 MLLM-Based Judgment

A critical challenge in our framework is how to verify the achievement of each essential state. Traditional function-based methods, which rely on parsing UI structures like the XML or View Hierarchy, are not suitable for this task of dynamic, constantly updating apps. These methods often struggle for two primary reasons: (1) many third-party proprietary apps feature mis-aligned or inconsistent UI trees that are difficult to parse reliably, and (2) they suffer from a semantic gap, as many essential states are defined by high-level concepts (e.g., *"the first product is selected"*) that cannot be verified by simply checking for a specific widget ID or text string in an constantly updating environment.

To overcome these limitations, we leverage the sophisticated visual and language understanding capabilities of modern MLLMs. Inspired by works on using MLLMs as evaluators (Chen et al., 2024a), we employ an MLLM to act as the core judgment mechanism for A3. Our approach, however, refines how the MLLM is applied. Prior work like SPA-bench (Chen et al., 2024b) pioneered the use of an MLLM to directly evaluate an entire, multi-step task trajectory based on the screenshots combination. This naive end-to-end judgment is a complex reasoning task that resulted in an accuracy of approximately only 80% in trajectory evaluations, which is unreliable. In contrast, our essential-state evaluation method is perfectly designed to address this challenge by decomposing the overall task into a series of simpler, more discrete verification steps of essential-states. This simplifies the reasoning required of the MLLM, as it makes a focused judgment on a single, much simpler milestone rather than a long sequence, thereby aiming for higher and more reliable evaluation accuracy.

Commercial MLLMs The advanced capabilities of commercial MLLMs, such as Gemini-2.5-pro, make them highly effective for the role of an automated evaluator. Their robust performance in visual comprehension and instruction following provides a reliable evaluation for judging whether an essential-state has been achieved within a given set of screen frames. Though commercial MLLMs performs good as an evaluator, a significant practical challenge associated with these commercial models is the monetary cost of their API services, as each evaluation call incurs a fee. This creates a critical trade-off between evaluation granularity and cost. Therefore, to ensure our framework is both accurate and economically viable, we find an optimal sliding window parameters, specifically the window size and interval size, to minimize the total number of API calls without compromising evaluation fidelity (experiments and results detailed in Appendix A.3).

Open-source MLLMs While optimizing the sliding window parameters mitigates the expense of using commercial APIs, a persistent monetary cost remains for each evaluation. To address this fundamental challenge of cost and accessibility, we propose a lightweight, open-source alternative that eliminates API fees and allows for local deployment, thereby providing researchers with greater control and transparency. To this end, we introduce the A3RM (Android Agent Arena Reward Model), a specialized reward model fine-tuned from Qwen3-VL-8B (Bai et al., 2025a). We selected this base model due to its strong demonstrated capabilities in GUI related tasks, making it an ideal foundation for our evaluator. In summary, **A3RM** serves as an accurate, cost-effective, and transparent judge for essential-state verification.

3.4 A3 Pipeline Suite

To accompany our benchmark, we introduce a comprehensive, open-source software pipeline, as illustrated in Figure 1. The core of this suite is the AITK (Android Interaction Toolkit), a lightweight and customizable framework for managing agent-device interaction, data collection, and task execution. AITK is mainly designed for model-based agents and it features a plug-in architecture for integrating custom agents and tasks. Another component of the suite is evaluator, which implements our essential-state evaluation methods. This evaluator is designed to be agent-agnostic, allowing it to be applied to any standard trajectory data, independent of the agent and framework that produced it. The complete pipeline, including our human annotation tools, will be open-sourced to foster reproducibility and advance future research.

4 Experiments

4.1 A3 Reward Model (A3RM)

We developed the A3RM (Android Agent Arena Reward Model) as a deployable alternative reward model. We selected Qwen3-VL-8B (Bai et al., 2025a) as the base model, given its strong performance on GUI related tasks. Our preliminary experiments revealed that a small window size of 2 yielded the best performance for the 8B model. This is likely due to its more limited context capacity. Therefore, we adopted a window size of 2 and interval size of 1 for all A3RM data collection and training.

Data Construction We constructed a comprehensive training dataset by collecting an average of three distinct, successful human trajectories for each of the 100 benchmark tasks, ensuring the coverage of diverse valid trajectories. Human annotators manually identified the specific state transitions where essential states were achieved; these constitute our positive samples, while all other transitions within these expert trajectories serve as negative samples. To further improve the

Metric	Gemini-2.5-pro	A3RM
Essential State Level		
Precision	87.3	95.7
Recall	96.3	94.8
F1 Score	91.5	95.3
Accuracy	89.5	96.6
Task Level		
Precision	85.7	96.0
Recall	96.0	96.0
F1 Score	90.6	96.0
Accuracy	95.0	98.0

Table 2: A3RM evaluation metric performance across different models. Metrics are reported for both Essential State (Ess. State) and Task levels.

model's discriminative capability, we employed a negative sample mining strategy. Leveraging the high-recall, low-precision characteristic (Table 2) of the Gemini-based evaluator, we processed agent-generated trajectories and labeled additional negative samples. The final dataset comprises 8241 step-wise samples derived from expert trajectories (981 positive and 7260 negative samples), augmented with 1083 negative samples from agent trajectories.

Training We finetuned Qwen3-VL-8B using the dynamic sampling policy optimization (DAPO) algorithm (Liu et al., 2025b) on our curated dataset. Given the natural class imbalance in our data (far more negative samples than positive ones), we oversampled the positive class by a factor of four in each training epoch to prevent the model from collapsing to a trivial solution of always predicting a negative outcome.

Evaluator Performance Table 2 presents the performance of our fine-tuned A3RM compared to the Gemini-2.5-pro baseline on a held-out A3 test set. Despite its significantly smaller size, A3RM surpasses the commercial model across key metrics. At the essential state level, it achieves 95.7% Precision (+8.4 points), and at the task level, it dominates with 96.0% Precision (+10.3 points) while maintaining parity in Recall. This performance validates our data construction strategy, which conditions the model to reject ambiguous states that zero-shot models frequently hallucinate as successes. However, we emphasize that A3RM is a specialized model optimized for the A3 task distribution; while this specialization yields superior in-domain performance compared to the general-purpose Gemini, it is designed specifically for this benchmark. Ultimately, this high precision is critical for reward modeling, as it minimizes the risk of inadvertently reinforcing agents for failed trajectories.

4.2 A3 Benchmark

Experimental Setup To establish a comprehensive baseline, we evaluated a diverse suite of mobile GUI agents. We selected seven representative single-model agents: Qwen2.5-VL-7B (Bai et al., 2025b), Qwen3-VL-8B (Bai et al., 2025a), UI-TARS-1.5-7B (Qin et al., 2025), UI-Genie-7B (Xiao et al., 2025), UI-Venus-7B (Gu et al., 2025), InfiGUI-R1-3B (Liu et al., 2025c), and GUI-OWL-7B (Ye et al., 2025). Additionally, we evaluated two agent frameworks: Mobile-Use (Li et al., 2025) (powered by Qwen2.5-VL-7B) and T3A (Rawles et al., 2025) (evaluated with both Gemini-2.5-pro and Qwen2.5-VL-7B). All agents were evaluated on the A3 benchmark using their official, publicly available prompts and inference settings to ensure reproducibility. Detailed results are presented in Table 3 (A3RM evaluation) and Table 6 (Gemini evaluation in Appendix A.4).

Performance Analysis The results indicate that A3 poses a rigorous challenge for current state-of-the-art agents. Among open-source single-model agents, InfiGUI-R1 distinguishes itself as the leader with a Success Rate (SR) of 27.0%. However, the broader landscape reveals significant fragility, where most models exhibit precipitous performance declines on "Hard" tasks, where success rates frequently approach zero. In sharp contrast, the proprietary T3A + Gemini-2.5-pro agent establishes a strong upper bound with a 53.0% SR. This nearly two-fold performance gap underscores a critical reality: while specialized open-source agents are evolving, they still lag significantly behind large-scale commercial MLLMs in the complex, multi-step reasoning required for dynamic GUIs. Ultimately, with even the top-performing system failing nearly half the tasks, A3 reveals substantial room for improvement across the board, positioning the commercial model's performance as a current "skyline" for the open-source community to pursue.

Frameworks vs. Foundation Models Our ablation of frameworks versus base models reveals critical insights. First, structured frameworks significantly enhance weak base models: Qwen2.5-VL fails almost completely as a standalone agent (3.0% SR), but when integrated into the Mobile-Use or

Agent	Metric	Easy	Medium	Hard	Operation	Inf. Query	Overall
UI-TARS-1.5	SR	20.0	10.0	4.0	13.9	7.1	12
	ESAR	35.3	21.8	23.9	31.2	20.9	28.2
UI-Venus	SR	28.5	15.0	16.0	20.8	17.8	20
	ESAR	41.2	29.7	27.1	33.9	27.5	32.0
UI-Genie	SR	25.8	10.0	0.0	16.7	3.6	13
	ESAR	41.2	25.8	32.3	34.9	25.3	32.1
InfiGUI-R1	SR	34.3	30.0	12.0	34.7	7.1	27
	ESAR	55.3	50.8	51.0	54.6	46.2	52.1
GUI-OWL	SR	31.4	7.5	0.0	18.1	3.6	14
	ESAR	49.4	26.6	23.9	35.8	23.1	32.0
Qwen2.5-VL	SR	5.7	0.0	4.0	4.2	0.0	3
	ESAR	10.5	10.9	21.8	15.6	11.0	14.2
Qwen3-VL	SR	34.3	12.5	0.0	20.8	7.1	17
	ESAR	50.6	35.2	31.3	44.0	24.2	38.2
Mobile-Use + Qwen2.5-VL	SR	34.3	10.0	0.0	22.2	0.0	16
	ESAR	48.2	39.9	31.3	43.1	30.8	39.5
T3A + Qwen2.5-VL	SR	31.4	10.0	4.0	19.4	3.8	15
	ESAR	37.6	28.1	28.1	29.4	34.1	30.7
T3A + Gemini-2.5-pro	SR	57.1	55.0	44.0	55.6	46.6	53
	ESAR	68.2	67.9	62.5	72.9	50.5	66.4

Table 3: Benchmark results evaluated by our A3RM. We report Task Success Rate (SR) and Essential State Achieved Rate (ESAR) across task categories and difficulties.

T3A frameworks, its performance jumps to 16.0% and 15.0% respectively. This suggests that well designed frameworks can compensate for limited reasoning capabilities. However, the backbone model remains the upper bound of performance. When the exact same T3A framework is powered by Gemini-2.5-pro, performance skyrockets to 53.0% (compared to 15.0% with Qwen2.5). This demonstrates that while frameworks provide necessary structure, they cannot fully mitigate the reasoning deficits of the underlying vision-language model. We also observe rapid evolution in base models; Qwen3-VL (17.0% SR) naturally outperforms its predecessor Qwen2.5-VL (3.0% SR) by a wide margin without any framework assistance.

Essential State Evaluation The Essential State Achieved Rate (ESAR) provides a more granular view of agent capabilities than the binary SR metric. A consistent trend across all agents is that ESAR is substantially higher than SR. This discrepancy highlights the primary failure mode of current agents: they possess sufficient semantic understanding to initiate tasks and navigate early stages (high ESAR) but lack the long-horizon robustness required to reach the final state without encountering a terminal error (low SR). This phenomenon also corresponds to our case studies in Appendix A.5, where current agents mostly fail in progress awareness.

We also provide more A3RM and essential state evaluation generalization in Appendix A.6, sliding window analysis in Appendix A.3 and Gemini evaluation experiments and results in Appendix A.4.

5 Conclusion

We introduced the Android Agent Arena (A3), a benchmark that bridges the gap between static evaluations and real-world utility by incorporating dynamic tasks from popular online applications. To overcome the opacity of closed-source apps, we proposed the essential-state evaluation method and the fine-tuned A3RM, offering a scalable, visual-based metric that captures granular agent progress beyond binary success. Our experiments reveal that while current agents struggle with long-horizon robustness, our open-sourced pipeline provides the necessary infrastructure to accelerate the development of truly autonomous mobile assistants.

Limitations

First, our benchmark is confined to the Android ecosystem. Expanding to iOS remains effectively unsolvable due to restrictive system policies that prevent the virtualization and programmatic control necessary for reproducible testing. Second, despite our rigorous curation, we exclude specific high-frequency categories like instant messaging, where strict authentication protocols and privacy concerns make automated account resets impractical. Furthermore, while our A3RM evaluator demonstrates high precision, it is inherently a probabilistic model and still has hallucinations. Consequently, while A3 significantly advances ecological validity, these constraints highlight the trade-offs required to balance reproducibility with the closed, secure nature of modern mobile platforms.

Acknowledgement

This project is funded in part by Shenzhen Loop Area Institute, by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)'s InnoHK, in part by NSFC-RGC Project N_CUHK498/24, and in part by Guangdong Basic and Applied Basic Research Foundation (No. 2023B1515130008, XW).

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. 2025. AMEX: Android multi-annotation expo dataset for mobile GUI agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2138–2156, Vienna, Austria. Association for Computational Linguistics.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yinuo Liu, Yaochen Wang, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024a. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. *Preprint*, arXiv:2402.04788.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, and 1 others. 2024b. Spabench: A comprehensive benchmark for smartphone agent evaluation. In *NeurIPS 2024 Workshop on Open-World Agents*.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, Yue Wen, Jingya Dou, Fei Tang, Jinzhen Lin, Yulin Liu, Zhenlin Guo, Yichen Gong, Heng Jia, Changlong Gao, and 5 others. 2025. Ui-venus technical report: Building high-performance ui agents with rft. *Preprint*, arXiv:2508.10833.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shenzhi Wang, Xinchen Xu, Shuofei Qiao, Zhaokai Wang, Kun Kuang, Tieyong Zeng, Liang Wang, and 10 others. 2025. OS agents: A survey on MLLM-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7436–7465, Vienna, Austria. Association for Computational Linguistics.

Juyong Lee, Taywon Min, Minyong An, Dongyoon Hahm, Haeone Lee, Changyeon Kim, and Kimin Lee. 2025. Benchmarking mobile device control agents across diverse configurations. *Preprint*, arXiv:2404.16660.

Ning Li, Xiangmou Qu, Jiamu Zhou, Jun Wang, Muning Wen, Kounianhua Du, Xingyu Lou, Qiuying Peng, Jun Wang, and Weinan Zhang. 2025. Mobileuse: A hierarchical reflection-driven GUI agent for autonomous mobile operation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154.

Guangyi Liu, Pengxiang Zhao, Yaozhen Liang, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Yue Han, Shuai Ren, Hao Wang, Xiaoyu Liang, WenHao Wang, Tianze Wu, Zhengxi Lu, Siheng Chen, LiLinghao, Hao Wang, Guanjing Xiong, and 2 others. 2025a. LLM-powered GUI agents in phone automation: Surveying progress and prospects. *Transactions on Machine Learning Research*.

Jiacai Liu, Chaojie Wang, Chris Yuhao Liu, Liang Zeng, Rui Yan, Yiwen Sun, and Yang Liu. 2025b. DAPO : Improving multi-step reasoning abilities of large language models with direct advantage-based policy

optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. 2025c. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *Preprint*, arXiv:2504.14239.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, and 1 others. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*.

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

Yuanyi Song, Heyuan Huang, Qiqiang Lin, Yin Zhao, Xiangmou Qu, Jun Wang, Xingyu Lou, Weiwen Liu, Zhuosheng Zhang, Yong Yu, Weinan Zhang, and Zhaoxiang Wang. 2025. Colorbench: Benchmarking mobile agents with graph-structured framework for complex long-horizon tasks. *ArXiv*, abs/2510.14621.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, Bin Wang, Chuhan Wu, Yasheng Wang, Ruiming Tang, and Jianye Hao. 2025. Gui agents with foundation models: A comprehensive survey. *Preprint*, arXiv:2411.04890.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, and 1 others. 2025. Ui-genie: A self-improving approach for iteratively boosting mllm-based mobile gui agents. *arXiv preprint arXiv:2505.21496*.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. 2026. Ui-mem: Self-evolving experience memory for online reinforcement learning in mobile gui agents. *arXiv preprint arXiv:2602.05832*.

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. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6061–6072.

Weikai Xu, Zhizheng Jiang, Yuxuan Liu, Pengzhi Gao, Wei Liu, Jian Luan, Yuanchun Li, Yunxin Liu, Bin Wang, and Bo An. 2025a. Mobile-bench-v2: A more realistic and comprehensive benchmark for vlm-based mobile agents. *ArXiv*, abs/2505.11891.

Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. 2025b. AndroidLab: Training and systematic benchmarking of android autonomous agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2166, Vienna, Austria. Association for Computational Linguistics.

Leyang Yang, Ziwei Wang, Xiaoxuan Tang, Sheng Zhou, Dajun Chen, Wei Jiang, and Yong Li. 2026. Probench: Benchmarking gui agents with accurate process information. In *AAAI Conference on Artificial Intelligence*.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *Preprint*, arXiv:2508.15144.

Danyang Zhang, Hongshen Xu, Zihan Zhao, Lu Chen, Ruisheng Cao, and Kai Yu. 2023. Mobile-env: an evaluation platform and benchmark for llm-gui interaction. *arXiv preprint arXiv:2305.08144*.

Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, and Yuwen Xiong. 2025. Easyr1: An efficient, scalable, multi-modality rl training framework. `https://github.com/hiyouga/EasyR1`.

A Appendix

A.1 App List

We select 20 apps in 20 categories from Google Play Store top charts as listed in Table 4.

App	Category
CNN	News
TripAdvisor	Events
Amazon	Shopping
Omio	Travel
Bluecoins	Finance
Tasks	Productivity
N Calendar	Calendar
File	Tools
eboox	Read
YouTube	Video
Home Workout	Fitness
Pinterest	Lifestyle
Chrome	Browser
CityMapper	Navigation
Joytify	Music
Joplin	Notes
Wikipedia	Entertainment
Weather Forecast	Weather
Supercook	Food
Gmail	Business

Table 4: List of apps and corresponding categories.

A.2 Tasks & Essential States

For clearer demonstration of our tasks and essential states, we list 10 examples in Table 8.

A.3 Sliding Window and Interval Study

We conduct experiments on the sliding window size and interval size (number of steps) for the commercial MLLMs (Gemini) evaluation to identify the optimal parameters to minimize the number of API calls without compromising evaluation fidelity. Table 5 summarizes the evaluation results across different sliding window sizes and interval settings. Specifically, the sliding window size ranges from 2 to 6, with interval sizes set to half of the window size. From the results, we observe that sliding window sizes of 3 and 4 yield the highest evaluation accuracies for both essential states and overall task performance. Among these, the configuration with a sliding window size of 4 and an interval size of 2 achieves the lowest average computational cost, making it the preferred setting for the evaluation method.

Further analysis of our results reveals a distinct trade-off in the selection of the sliding window size, with the MLLM evaluator's performance degrading at both small and large extremes. When a small window is combined with a non-overlapping stride (e.g., size of 2, stride of 2), evaluation accuracy suffers. This occurs because a critical state transition can be split across the boundary of two consecutive windows. For example, an agent's action and its immediate on-screen result might fall into separate windows, depriving the MLLM of the necessary context to make a correct judgment. Conversely, when the window size becomes too large (e.g., greater than 4), performance degrades due to loss of visual fidelity. To be processed, frames in the window are concatenated into a single composite image. With larger window sizes, each frame is significantly downscaled, causing a severe loss of resolution that can render small text and UI elements illegible. This image compression impairs the MLLM's ability to accurately identify essential states, leading to misjudgments.

A.4 Gemini Evaluation

Gemini-Based Evaluation Analysis The results from the Gemini-based evaluation (Table 6) highlight the critical importance of foundational model capability and agentic scaffolding. The proprietary T3A + Gemini-2.5-pro agent establishes a dominant upper bound with an Overall Success Rate (SR) of 58.0%, significantly outperforming all open-source alternatives. Among single-model agents, InfiGUI-R1 leads with a 29.0% SR, demonstrating that specialized fine-tuning can yield competitive results. We also observe a substantial generational leap in the Qwen family: Qwen3-VL achieves a 23.0% SR, a nearly five-fold improvement over the baseline Qwen2.5-VL (5.0%). Furthermore, the data underscores the efficacy of agent frameworks; wrapping the weak Qwen2.5-VL base model in the T3A framework boosts its performance from 5.0% to 24.0%, effectively bridging the gap to the stronger Qwen3-VL base model. However, despite these gains, the sharp

Window Size	Interval Size	Essential-State Acc	Task Acc	Average Cost ($)
2	1	0.94	**0.91**	0.196
	2	0.88	0.86	0.099
3	1	**0.95**	**0.91**	0.195
	2	0.94	0.90	0.098
4	2	**0.95**	**0.91**	0.098
	3	**0.95**	0.90	0.071
5	2	0.93	0.89	0.097
	3	0.92	0.89	0.069
6	3	0.91	0.88	0.070
	4	0.89	0.87	0.050

Table 5: Sliding window and interval size study for essential-state evaluation. The highest accuracy is in bold. The average cost is computed by the total API cost of three MLLMs over 30 tasks.

decline in performance on "Hard" tasks—where most open-source agents drop to single-digit success rates—indicates that complex, multi-step reasoning remains a significant bottleneck.

Gemini VS A3RM Comparing these results with the A3RM evaluation reveals a systematic "optimism bias" in the commercial Gemini evaluator. Across virtually all agents, Gemini assigns higher success rates than our fine-tuned A3RM. For instance, the leading T3A + Gemini agent is scored at 58.0% by the Gemini evaluator but drops to 53.0% under A3RM. Similarly, InfiGUI-R1 decreases from 29.0% to 27.0%, and Mobile-Use drops from 19.0% to 16.0%. We attribute this discrepancy to the design of A3RM: explicitly trained with human labels and hard negative mining, A3RM is more aligned to human judgment and Gemini suffers from the low precision situation. This suggests that while commercial MLLMs offer high recall, they lack the domain-specific precision required for rigorous GUI evaluation. Crucially, however, the relative rankings of the agents remain largely consistent across both evaluators (e.g., T3A > InfiGUI > Qwen2.5).

A.5 Case Study

Through observation and analysis of the ten agents performance on A3 benchmark, we noticed some representative and remarkable cases where researchers would like to solve in the future study.

A.5.1 General Agent Issues

Progress Unawareness A critical failure mode observed across most model-based agents is Progress Unawareness, a fundamental inability to track their position within a task sequence. This limitation typically manifests in two ways:

- **Redundant Actions**: Agents often fail to recognize that a required step has already been completed. We observed numerous instances where an agent would get stuck in a loop, repeatedly attempting to achieve an essential state that was already satisfied, even when its action history was provided in the prompt (Figure 4). This suggests a failure to effectively parse its own operational history.

- **Failure to Terminate**: A related issue is the inability to recognize overall task completion. Many agents, despite having successfully achieved all essential states, do not issue a stop action. Instead, they continue to perform irrelevant operations until the AITK framework terminates them for exceeding the maximum step limit.

This lack of progress awareness points to a deeper deficiency in the agents' planning and state-tracking capabilities. An agent that cannot reliably determine what it has already done or recognize when its final goal has been met will struggle with complex, multi-step tasks. Improving this self-awareness and goal recognition is therefore a crucial direction for future research in developing more robust and efficient GUI agents.

Screen misunderstanding Another commonly observed failure mode is a lack of Screen Comprehension, where an agent fails to accurately ground

Agent	Metric	Easy	Medium	Hard	Operation	Inf. Query	Overall
UI-TARS-1.5	SR	14.3	5.0	4.0	9.7	3.5	8
	ESAR	29.4	17.2	9.3	20.2	13.2	18.1
UI-Venus	SR	31.4	17.5	12.0	22.2	17.8	21
	ESAR	47.1	27.3	31.2	36.2	28.6	34.0
UI-Genie	SR	28.6	15.0	0.0	20.8	3.6	16
	ESAR	47.1	28.1	31.3	37.2	27.5	34.3
InfiGUI-R1	SR	40.0	32.5	8.0	36.1	10.7	29
	ESAR	58.8	53.1	52.1	56.4	49.5	54.4
GUI-OWL	SR	25.7	5.0	4.0	15.3	3.6	12
	ESAR	51.7	27.3	21.8	36.2	23.1	32.4
Qwen2.5-VL	SR	8.6	2.5	4.0	6.9	0.0	5
	ESAR	25.9	14.8	26.0	22.0	19.8	21.4
Qwen3-VL	SR	40.0	17.5	8.0	25.0	17.9	23
	ESAR	55.3	40.6	36.5	50.5	37.4	46.7
Mobile-Use + Qwen2.5-VL	SR	34.3	15.0	4.0	25.0	3.6	19
	ESAR	47.1	43.0	30.2	43.1	33.0	40.1
T3A + Qwen2.5-VL	SR	40.0	17.5	12.0	26.4	17.9	24
	ESAR	44.7	32.8	31.3	32.1	43.9	35.6
T3A + Gemini-2.5-pro	SR	62.8	57.5	52.0	62.5	46.4	58
	ESAR	71.76	70.3	65.6	76.6	51.6	69.3

Table 6: Benchmark results evaluated by the commercial Gemini-2.5-pro model. We report Task Success Rate (SR) and Essential State Achieved Rate (ESAR) across task categories and difficulties.

its intended action to the correct visual element on the screen. This issue, which primarily affects `click` actions, manifests in two distinct ways:

- **Incorrect Element Identification:** In some cases, the agent's reasoning is sound (e.g., "click the menu button") but it visually misidentifies the target, instead clicking a different element like a profile icon. This represents a failure in high-level visual recognition, as shown in Figure 5 left.
- **Inaccurate Coordinate Localization:** In other instances, the agent correctly identifies the target element but fails to predict its precise coordinates. This results in a "shifted click" that lands on an adjacent area instead of the intended element, as illustrated in Figure 5 right. This points to a limitation in fine-grained spatial reasoning.

A.5.2 Dynamic Online Apps Specialty

We analyzed failure cases specific to the dynamic nature of online apps and identified three primary categories of error:

- **Information Similarity and Ranking:** Online apps often present search results where the correct item is buried among highly similar distractions. Agents frequently struggle to discern the target from these visual distractors. For instance, in YouTube tasks, agents often select the top-ranked video rather than performing the necessary scrolling actions to locate the specific target content further down the list.
- **Dynamic Interference:** Unlike static offline benchmarks, online environments feature unpredictable elements such as pop-up ads and sponsored content. In apps like Chrome and Home Workout, agents often fail to distinguish between organic results and sponsored links, or lack the logic to identify and close sudden whole-screen ads, leading to task deadlocks.
- **Rich Information and Layout Complexity:** The dense information presentation and deep hierarchical designs of modern apps hinder effective planning. In complex interfaces like Amazon and Omio, agents are often overwhelmed by the volume of product details or struggle to navigate nested filtering and sorting menus, resulting in incorrect navigation paths.

Metric	Gemini-2.5-pro	A3RM	A3RM-Continued
Essential State Level			
Precision	87.5	93.1	**97.7**
Recall	93.3	91.1	**95.6**
F1 Score	90.3	92.1	**96.6**
Accuracy	87.1	90.0	**95.7**
Task Level			
Precision	93.4	94.1	**100.0**
Recall	88.2	94.1	**94.1**
F1 Score	90.7	94.1	**97.0**
Accuracy	88.0	92.0	**96.0**

Table 7: Performance comparison on 25 newly introduced tasks.

A.6 Generalization Analysis

To evaluate the generalization capability of the proposed evaluator and the essential state representation, we conducted experiments on 25 newly introduced tasks outside the original training distribution, comparing Gemini-2.5-pro, A3RM, and A3RM-Continued trained with additional data from the new tasks. As shown in Table 7, A3RM still outperforms Gemini-2.5-pro on unseen tasks across all metrics. This result highlights the advantage of a specialized evaluation model trained with structured supervision, demonstrating that A3RM is able to generalize beyond its training distribution to a meaningful extent, benefiting from structured supervision rather than solely relying on the broad reasoning capabilities of general purpose models.

More importantly, the performance gains of A3RM-Continued highlight the extensibility of the essential state representation. With continued training on new tasks, A3RM-Continued achieves consistent improvements in Precision and Accuracy while maintaining stable Recall, indicating that essential states capture transferable structural regularities of GUI task execution. By modeling semantically coherent state transitions and task-relevant state dependencies, essential states provide stable and well-aligned supervision that naturally generalizes to previously unseen task distributions. Consequently, the essential-state framework supports systematic expansion to new tasks, establishing a general and scalable foundation for GUI-Agent evaluation.

A.7 Risks & License

1. Since the apps are online and some of them requires user login, agent's wrong actions may lead to content malfunctioning.

2. MIT license: GUI-OWL, UI-Genie. Apache: UI-TARS, UI-Venus, InfiGUI-R1, Qwen2.5/3-VL. All use are consistent with the license.

A.8 Training Details

We use L40s (48G) to train A3RM for 600 GPU-hour. We use EasyR1 (Zheng et al., 2025) as the training codebase and use DAPO as the RL training algorithm.

A.9 AI Declaration

We use Gemini to help paper writing and one of the evaluation method.

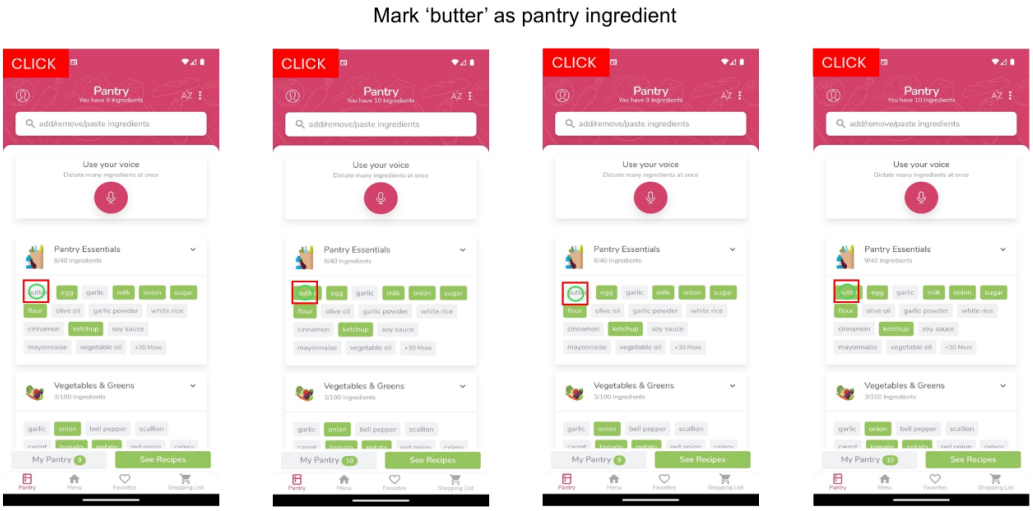

Figure 4: Example for redundant actions in an essential state.

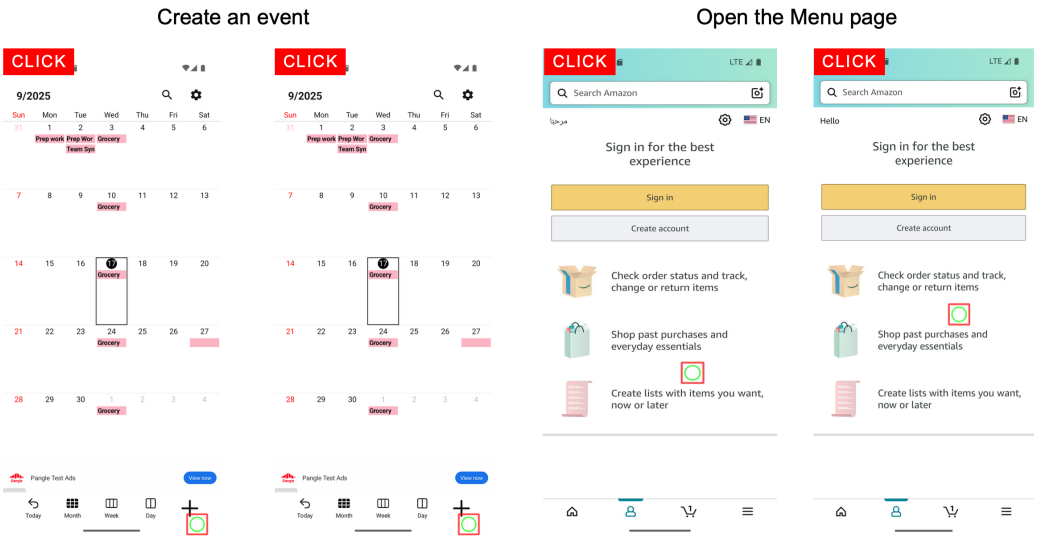

Figure 5: Examples for screen misunderstanding in an essential state.

Table 8: Sample tasks and their corresponding essential states from the A3 Benchmark.

Task Instruction	**Essential States**
Navigate to CNN's Science section and check the top headline news. What is the title?	• CNN Science section is selected • The top headline news in Science section is selected • The title of the article is answered
Search for 'hotels in Seoul' on Tripadvisor for 1 room and 2 adults, sorted by traveler ranking.	• Room is set to 1 room • Guest details are set to 2 adults • 'hotels in Seoul' is searched • Results are sorted by traveler ranking
Open Amazon and search for 'Laptop Sleeve'. Filter the material by 'carbon fiber'. What is the price of the first result?	• 'Laptop Sleeve' is searched • Results are filtered by material 'carbon fiber' • Price of the first result is answered
Find the cheapest flight from Paris to Rome for 2 adults on Omio departing tomorrow. What is the total price?	• Departure location is set to Paris • Arrival location is set to Rome • Passengers set to 2 adults • Departure date set to tomorrow • Flights are searched • Results for flights are sorted by 'Cheapest price' • Cheapest flight price is answered
Open Citymapper and search route from London Bridge to Oxford Street. How long is the estimated walking time?	• Route starting location is set to London Bridge • Route destination is set to Oxford Street • Estimated walking time is answered
Open Gmail and send an email to 'stock_notify_01@163.com' with subject 'Meeting time' and body 'When is the meeting?'	• New email page is opened • Recipient set to 'stock_notify_01@163.com' • Subject 'Meeting time' is added • Body 'When is the meeting?' is added • Email is sent
Search 'How to cook steak' in Youtube, play video by 'Hodder Books' titled 'Gordon Ramsay's Ultimate Cookery Course'	• 'How to cook steak' is searched • Video by 'Hodder Books' with title 'Gordon Ramsay's Ultimate Cookery Course' is selected • Video is played
Open Home Workout, search Abs workouts, select 'abs beginner', and start training.	• Abs workouts are searched • Workout 'abs beginner' is selected • Training is started
Remove butter from shopping list in supercook, mark it as pantry ingredient, and tell me how many recipes can be made.	• Shopping list is accessed • Butter is removed from shopping list • Butter is marked as pantry ingredient • Number of recipes possible is answered
Search 'Quantum Computing' in Wikipedia, select first article, open table of contents, and go to Algorithms chapter.	• 'Quantum Computing' is searched • First article is selected • Table of contents is opened • Algorithms chapter is accessed