A Task-State Representation for Long-Horizon Mobile GUI Agents

Anonymous ACL submission

Abstract

While long-horizon mobile GUI agents typically rely on thought-action-observation loops, they struggle to separate persistent task states from transient screen observations. As execution histories grow, this entanglement imposes a severe context burden, causing agents to forget initial requirements, hallucinate progress, or repeatedly interact with stale interfaces. To address this, we introduce Task-State Representation (TSR)—a training-free framework that explicitly decouples task state from sensory input. Acting as a lightweight external wrapper, TSR maintains three structured components: a global instruction summary, a dynamic progress tracker for subgoals, and a transition-aware action verifier. By continuously updating through pre- and post-action visual comparisons, TSR effectively guides the agent's reasoning without requiring architectural modifications. Experiments across four mobile GUI benchmarks validate TSR's effectiveness, yielding up to a 12 absolute point increase in success rate on complex cross-application and memory-intensive tasks.

1 Introduction

Automating mobile tasks via graphical user interfaces (GUIs) remains a long-standing goal in the development of intelligent agents. Recent multimodal large language models (MLLMs) (Liu et al., 2023) have facilitated prompt-based GUI actors that observe screenshots, reason about the current state, and generate executable actions, such as tapping, typing, or scrolling (Zhang et al., 2025; Hong et al., 2024). The dominant paradigm for handling long-horizon tasks adopts a thought-action-observation (Yao et al., 2022) loop: at each step, the actor receives the task instruction, a window of recent screenshots, and a history of previous reasoning and actions, and subsequently generates a new reasoning trace followed by an action (Zhang et al., 2024; Rawles et al., 2025). This append-all design relies on an implicit assumption—that the actor can reliably maintain awareness of the overall task goal and the cumulative progress from an ever-growing raw trajectory.

In practice, this assumption breaks down as the steps increases. We identify three recurring failure modes in long-horizon mobile benchmarks: (1) *goal drift*, where the actor gradually loses sight of the original task after observing numerous intermediate screens; (2) *progress hallucination*, where the actor lose access to earlier visual observations and fabricates past states when reasoning about cumulative progress; and (3) *stale-screen repetition*, where the actor misinterprets a delayed update in the user interface as a failed action and enters a localized loop.

These failures are not solely attributable to limitations in the model's capacity. Rather, they arise from a structural deficiency in how the input of the actor is organized: the standard prompt conflates two fundamentally different categories of information—*persistent task state* (the request of the user, the accomplished subgoals, and the remaining steps) and *transient observation state* (the content displayed on the current screen). Without an explicit mechanism to separate and maintain the former, the actor must re-derive the task progress from the raw history at every step—a burden that scales linearly with the length of the trajectory.

We propose a **task-state representation** that addresses this separation. The representation is maintained externally to a fixed GUI actor and is updated at each step by a training-free state updater that compares the pre-action and post-action screenshots. It comprises three functional views: a *global task-state summary* that preserves the original instruction, a *progress tracker* that records the completed and remaining subgoals, and a *transition-aware focus* that verifies the effectiveness of the preceding action and guides the next decision. The resulting state block is serialized and injected into

the prompt of the actor, requiring no model retraining and architectural modification.

We evaluate two base models across four mobile GUI benchmarks: MobileWorld (Kong et al., 2025), AndroidWorld (Rawles et al., 2025), MemGUI-Bench (Liu et al., 2026), and VenusBench-Mobile (Gong et al., 2026). Our approach improves success rates in most configurations, with gains of up to 12% on long-horizon tasks. Our contributions are:

- A task-state representation that separates persistent task state from transient observations for long-horizon mobile GUI agents.
- Empirical evaluation showing consistent improvements across four benchmarks and two base models, with ablations suggesting synergistic necessity of all three state views.
- Deeper analysis on the effect of structured task-state, identifying task horizon and state-tracking demand as key moderating factors.

2 Method

2.1 Problem Formulation

We formulate mobile GUI automation as a sequential decision-making problem. Given a task instruction I and an initial screen observation o_0, a GUI actor π_θ repeatedly observes the current screen and generates a response $r_t = (T_t, a_t)$, where T_t denotes a reasoning trace and a_t represents an executable action drawn from a predefined action space $\mathcal{A}$ (detailed in Appendix B). Following the execution of a_t, the environment transitions to a new screen state o_{t+1}, and the rollout continues until task completion or a predefined step limit.

To reduce context overhead, existing systems retain only the m most recent screenshots alongside the full text history (Li et al., 2026). Formally, at step t, the actor generates:

$$a_t = \pi_\theta(I,\ O_t,\ H_{t-1}),$$

where $O_t = \{o_{t-m+1}, \ldots, o_t\}$ denotes the recent observation window and $H_{t-1} = \{(T_1, a_1), \ldots, (T_{t-1}, a_{t-1})\}$ is the interaction history comprising all prior reasonings and actions. This design preserves local visual continuity but discards earlier observations. Consequently, the actor must infer task progress from past responses, a growing burden in long trajectories that leads to the potential failures.

2.2 Task-State Representation

To address these limitations, we introduce an externally maintained task-state representation S_t, which is updated at each step and injected into the actor's input. The state comprises three functional views, which are detailed below. Rather than operating as separate modules, these views are fields within a unified state object and are updated jointly at each step.

Global Task-State Summary. When early visual contexts are truncated, the actor is prone to losing sight of the initial task goal. To counteract this intent decay, we design a summary module to record the persistent semantics of the task along with the cumulative progress. The primary role of this summary is to maintain the visibility of the original task instruction, even when recent screenshots display only a narrow slice of the interface. This mechanism prevents the actor from drifting away from the intent of the user during long sequences of interaction.

Progress Tracker. During complex long-horizon tasks, the agent often loses track of their overarching goals, creating severe ambiguity in evaluating current progress. To resolve this ambiguity, we design a progress tracker that decomposes the task into atomic requirements and tracks the execution status of each. It maintains four fields: task decomposition, completed milestones, current subgoal, and remaining requirements. By making verified progress and unresolved steps explicit, the tracker mitigates the risk of the actor hallucinating earlier observations or terminating prematurely.

Transition-Aware Focus. Furthermore, since H_{t-1} reflects past intentions rather than environmental feedback, the actor risks repetitive loops during system delays or failures. To close this loop, the transition view evaluates the previous action by comparing observations o_{t-1} and o_t. If the outcome is uncertain, the representation prevents blind repetition by generating a *next-action focus*—such as verifying the state or refreshing a list—to guide the actor's next decision.

2.3 State Update and Actor Injection

At the beginning of a task, the initial state S_0 is derived solely from the task instruction I. At each subsequent step t, following the execution of action a_{t-1} by the actor and the generation of observation o_t by the environment, the state updater $\mathcal{U}_\phi$ receives

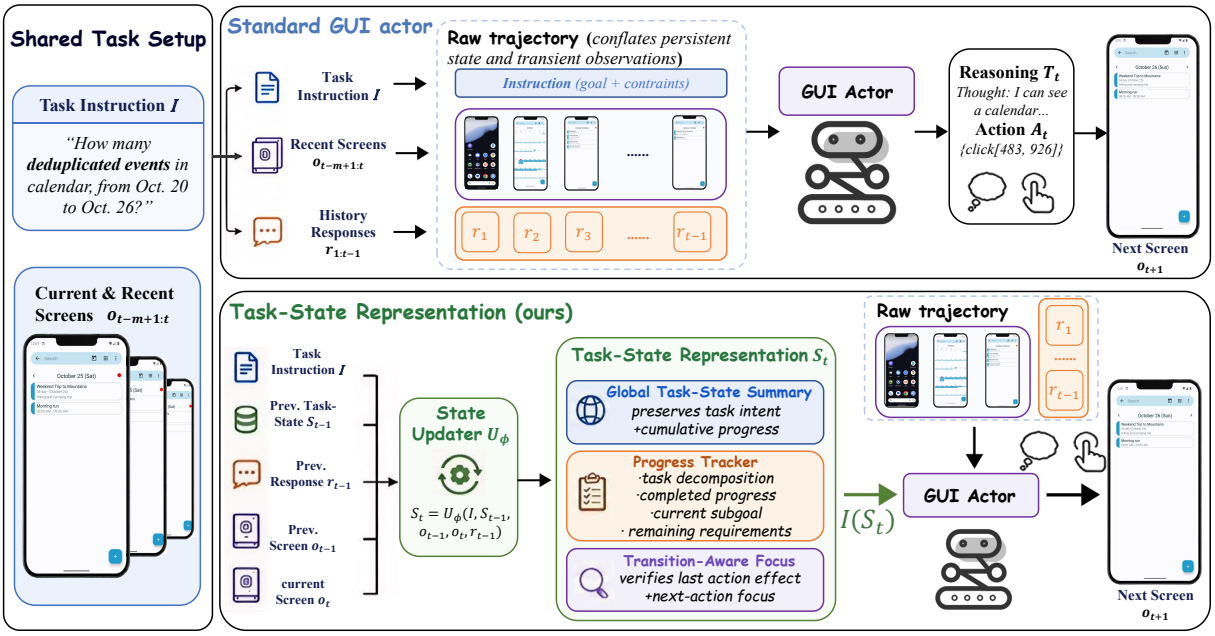

Figure 1: Overview of the task-state representation. At each step, the state updater compares pre- and post-action screenshots and updates a structured state block which is then injected into the actor's input.

the task instruction, the previous state, the previous response of the actor, and the screenshots captured before and after the action. It then produces the updated state through a single function call:

$$S_t = \mathcal{U}_\phi(I,\ S_{t-1},\ r_{t-1},\ o_{t-1},\ o_t).$$

The updater $\mathcal{U}_\phi$ is implemented via a prompted LLM that outputs a structured JSON object encompassing all three views. Subsequently, the actor receives the standard context augmented with a serialized rendering of the task state:

$$a_t = \pi_\theta(I,\ O_t,\ H_{t-1},\ \mathcal{I}(S_t)),$$

where $\mathcal{I}(S_t)$ denotes an injection function that formats the three state views into a text block appended to the prompt of the actor. Crucially, π_θ remains *fixed* throughout, so that the entire mechanism operates externally at inference time without any training or architectural modifications.

3 Experiments

3.1 Experimental Setup

We evaluate our approach on four online GUI benchmarks, including MobileWorld (Kong et al., 2025), AndroidWorld (Rawles et al., 2025), MemGUI-Bench (Liu et al., 2026) and VenusBench-Mobile (Gong et al., 2026). We implement the standard GUI actor based on previous studies (Kong et al., 2025) as baseline. More details are presented in Appendix A.

3.2 Main Results

Table 1 reports success rates and average steps across all settings. We list our findings below.

First, the most pronounced improvements appear on MobileWorld (+12% for Qwen3.5-plus, +9% for Kimi-k2.5), the benchmark with the longest average trajectories and cross-application dependencies. This is consistent with the hypothesis that explicit progress tracking is most beneficial when the agent navigate extended interaction sequences.

Besides, on memory-intensive and user-centric benchmarks (MemGUI-Memory, VenusBench-Mobile), both models benefit (+3.48% to +5.22%). However, the representation is not universally positive: Qwen3.5-plus degrades by 3.45% on AndroidWorld while Kimi-k2.5 improves by 3.45% on the same benchmark, suggesting that the utility of structured task-state tracking depends on the base model's inherent planning capability and the task complexity distribution.

Third, higher success rates do not always require more steps. In several cases (Kimi-k2.5 on VenusBench-Mobile: 41.2→38.3 steps; Qwen3.5-plus on MemGUI-Memory: 65.4→64.5 steps), the representation enables more direct task completion. Conversely, on MobileWorld the step count increases alongside SR, indicating that the representation helps agents persist through complex tasks they would otherwise abandon.

Model	Benchmark	Baseline (SR)	+Context (SR)	Δ SR	Baseline (Avg steps)	+Context (Avg steps)
Qwen3.5-plus	MobileWorld	43.00	55.00	+12.00	25.1	27.0
Qwen3.5-plus	AndroidWorld	61.21	57.76	-3.45	12.8	13.9
Qwen3.5-plus	MemGUI-Bench	26.56	28.91	+2.35	60.5	59.8
Qwen3.5-plus	MemGUI-Memory	23.48	28.70	+5.22	65.4	64.5
Qwen3.5-plus	VenusBench-Mobile	15.25	20.34	+5.09	38.7	38.2
Kimi-k2.5	MobileWorld	49.00	58.00	+9.00	29.6	28.3
Kimi-k2.5	AndroidWorld	51.72	55.17	+3.45	15.5	15.8
Kimi-k2.5	MemGUI-Bench	42.19	42.97	+0.78	51.6	55.8
Kimi-k2.5	MemGUI-Memory	37.39	40.87	+3.48	55.5	60.6
Kimi-k2.5	VenusBench-Mobile	22.03	26.27	+4.24	41.2	38.3

Table 1: Performance comparison of baseline and our proposed task-state representation.

Method	MobileWorld	AndroidWorld
Baseline	43.00	61.21
+Context w/o summary	50.00	59.48
+Context w/o tracker	49.00	61.21
+Context w/o transition	48.00	**63.79**
+Context full	**55.00**	57.76

Table 2: Ablation on Qwen3.5-plus (Success Rate, %).

3.3 Ablation Study

We ablate on Qwen3.5-plus by separately removing the task-state summary, progress tracker, and transition-aware focus from the representation while still updating the internal state. Results are shown in Table 2.

Firstly, we discover that on MobileWorld, the full representation outperforms every ablated variant by 5–7%, suggesting that the three components function synergistically. Among individual removals, dropping the transition-aware focus causes the largest degradation (55→48), suggesting that verifying action effectiveness is the most critical signal for long-horizon tasks. However, the pattern reverses on AndroidWorld. Specifically, removing the transition module improves over the baseline (63.79 vs. 61.21), while the full representation hurts. This indicates that for shorter, single-application tasks, transition verification introduces unnecessary decision noise. The task-state representation is therefore best viewed as conditionally beneficial—most valuable when tasks demand persistent state tracking across many steps.

4 Related Work

Mobile GUI Agents. Recent multimodal LLMs (Liu et al., 2023, 2025b) have enabled prompt-based GUI actors that translate instructions into touch actions on mobile screens (Zhang et al., 2025; Hong et al., 2024). Subsequent work improves grounding accuracy (Cheng et al., 2024) and scales toward long-horizon tasks via hierarchical planning (Lu et al., 2026), history compression (Li et al., 2026) or active loading (Liu et al., 2025a). However, existing methods often require retraining, limiting their use as general wrappers. To address this, we propose a training-free task-state representation that enriches prompts without altering the base agent.

Long-Horizon Context for LLM Agents. Managing growing interaction trajectories is a central challenge for LLM agents. Existing approaches primarily address this through trajectory compression: HiAgent (Hu et al., 2025) chunks working memory by subgoals, IterResearch (Chen et al., 2025) periodically reconstructs a condensed workspace, and summarization-based methods learn to shorten history via RL (Lu et al., 2025). These methods treat the trajectory as a monolithic stream to be shortened; in contrast, we maintain a separate structured representation alongside the actor's unmodified observation history.

5 Conclusion

In this paper, we propose a task-state representation that decouples persistent task state from transient screen observations to counter goal drift, progress hallucination, and stale-screen repetition for long-horizon mobile GUI agents. Our approach augments a fixed actor with three externally maintained views: a global task-state summary, a progress tracker, and a transition-aware focus, without any architectural modification. Evaluated across four benchmarks, our method improves success rates by up to 12 % on long-horizon cross-app tasks and yields consistent 3–5 % gains on memory-intensive tasks, validating the effectiveness of our approach.

Limitations

The state updater can propagate errors: if it incorrectly marks a subgoal as completed or omits a remaining requirement, the actor conditions subsequent decisions on a flawed state, potentially compounding mistakes across steps. Additionally, the updater introduces one extra LLM call per step (1024 tokens), increasing both latency and cost; for short tasks where the baseline already succeeds, this overhead provides no benefit and may degrade performance through over-decomposition. All experiments are conducted on Android-based mobile benchmarks; whether the representation generalizes to desktop or web GUI environments remains untested. Finally, the current design updates all three state views jointly—a more selective mechanism that activates views only when needed could reduce noise on simpler tasks.

References

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, and 1 others. 2025. Iterresearch: Rethinking long-horizon agents with interaction scaling. *arXiv preprint arXiv:2511.07327*.

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

Yichen Gong, Zhuohan Cai, Sunhao Dai, Yuqi Zhou, Zhangxuan Gu, Changhua Meng, and Shuheng Shen. 2026. Venusbench-mobile: A challenging and user-centric benchmark for mobile gui agents with capability diagnostics. *arXiv preprint arXiv:2604.06182*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and 1 others. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14281–14290.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, and 1 others. 2025. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. *arXiv preprint arXiv:2512.19432*.

Daiqiang Li, Zihao Pan, Zeyu Zhang, Ronghao Chen, Huacan Wang, Honggang Chen, and Haiyun Jiang. 2026. Rethinking token pruning for historical screenshots in gui visual agents: Semantic, spatial, and temporal perspectives. *arXiv preprint arXiv:2603.26041*.

Guangyi Liu, Pengxiang Zhao, Yaozhen Liang, Qinyi Luo, Shunye Tang, Yuxiang Chai, Weifeng Lin, Han Xiao, WenHao Wang, Siheng Chen, and 1 others. 2026. Memgui-bench: Benchmarking memory of mobile gui agents in dynamic environments. *arXiv preprint arXiv:2602.06075*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Zikang Liu, Junyi Li, Wayne Xin Zhao, Dawei Gao, Yaliang Li, and Ji-rong Wen. 2025a. Pal-ui: Planning with active look-back for vision-based gui agents. *arXiv preprint arXiv:2510.00413*.

Zikang Liu, Kun Zhou, Wayne Xin Zhao, Dawei Gao, Yaliang Li, and Ji-Rong Wen. 2025b. Less is more: High-value data selection for visual instruction tuning. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 3712–3721.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. 2025. Scaling llm multi-turn rl with end-to-end summarization-based context management. *arXiv preprint arXiv:2510.06727*.

Zhengxi Lu, Fei Tang, Guangyi Liu, Kaitao Song, Xu Tan, Jin Ma, Wenqi Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and 1 others. 2026. Ui-copilot: Advancing long-horizon gui automation via tool-integrated policy optimization. *arXiv preprint arXiv:2604.13822*.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, and 1 others. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pages 406–441.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, and 1 others. 2026. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024. Android in the zoo: Chain-of-action-thought for gui agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031.

A Experimental Details

Datasets. We evaluate on four benchmarks spanning different difficulty axes. **MobileWorld** (Kong et al., 2025): long-horizon, cross-application workflows; we exclude agent-user interaction and MCP-augmented tasks, retaining 100 GUI-only tasks. **AndroidWorld** (Rawles et al., 2025): reproducible Android tasks with programmatic reward verification. **MemGUI-Bench** (Liu et al., 2026): 115 memory-intensive tasks requiring cross-temporal and cross-spatial recall (denoted MemGUI-Memory for the memory-specific subset). **VenusBench-Mobile** (Gong et al., 2026): realistic user-centric tasks with capability diagnostics; we exclude unsupported tasks, leaving 118 tasks.

Baselines. Our baseline is the standard MobileWorld actor, which instantiates π_θ as defined in Section 2.1. We compare this baseline against the same actor augmented with the task-state representation S_t.

Implementation Details. We use two multimodal LLMs as the underlying model for both the actor and the state updater: Qwen3.5-plus (Yang et al., 2025) and Kimi-k2.5 (Team et al., 2026), accessed via Bailian API with temperature 0.

B Action Space Definition

The action space $\mathcal{A}$ differs between benchmarks. Table 3 lists all available actions.

Action	Parameters	Scope
`click`	coordinate $[x, y]$	All
`double_tap`	coordinate $[x, y]$	All
`long_press`	coordinate $[x, y]$	All
`drag`	start, end coords	All
`input_text`	text string	All
`answer`	text string	All
`navigate_home`	—	All
`navigate_back`	—	All
`scroll`	direction	All
`status`	complete/infeasible	All
`wait`	—	All
`keyboard_enter`	—	All
`open_app`	app name	AW
`swipe`	direction	AW

Table 3: Action space $\mathcal{A}$. AW = AndroidWorld only. Coordinates are normalized to $[0, S]$ where S is a model-specific scale factor (1000 for Qwen3.5-plus, 1 for Kimi-k2.5). `scroll` moves content; `swipe` performs system-level finger gestures. `answer` and `status` terminate the episode.

C LLM Configuration

Parameter	Actor π_θ	Updater $\mathcal{U}_\phi$
Temperature	0.0	0.0
Max output tokens	2048	1024
Obs. window m	3	2 (pre + post)
Model-specific		
Scale S (Qwen3.5-plus)	1000	
Scale S (Kimi-k2.5)	1	
API	Bailian	

Table 4: Representation and inference configurations. Both components use the same underlying model (Qwen3.5-plus (Yang et al., 2025) or Kimi-k2.5 (Team et al., 2026)) accessed via provider APIs.

The maximum number of interaction steps per task is set to 50 for MobileWorld; for AndroidWorld, MemGUI-Bench, and VenusBench-Mobile, we use the benchmark-specific preset limits defined by each environment. The state updater is invoked at every step regardless of the benchmark.

D Actor Prompt Template

The actor receives a structured prompt comprising a role definition, action framework, execution principles, the task instruction, and the injected task-state representation $\mathcal{I}(S_t)$. The template below shows the MobileWorld variant; AndroidWorld omits `open_app` and `swipe`. Coordinates are normalized to $[0, S]$ where S is the model-specific scale factor.

```
# Role: Android Phone Operator AI
You are an AI that controls an Android
phone to complete user requests.

# Action Framework
Respond with EXACT JSON format for one
of these actions:
| Action       | JSON Format              |
|--------------|--------------------------|
| click        | {"action_type":"click", |
|              |  "coordinate":[x,y]}    |
| long_press   | {"action_type":          |
|              |  "long_press",           |
|              |  "coordinate":[x,y]}    |
| drag         | {"action_type":"drag",  |
|              |  "start_coordinate":     |
|              |  [x1,y1],               |
|              |  "end_coordinate":       |
|              |  [x2,y2]}               |
| input_text   | {"action_type":          |
|              |  "input_text",           |
|              |  "text":"content"}      |
| answer       | {"action_type":"answer",|
|              |  "text":"response"}     |
| scroll       | {"action_type":"scroll",|
|              |  "direction":"down"}    |
| open_app     | {"action_type":          |
|              |  "open_app",             |
|              |  "app_name":"Calendar"} |
| status       | {"action_type":"status",|
|              |  "goal_status":          |
|              |  "complete"}             |

# Execution Principles
1. Communication: ALWAYS use 'answer'
   to reply to user questions.
2. Efficiency: Choose the simplest path.
3. Navigation: scroll = content move;
   swipe = system gesture.
4. Text input: Click input box first.

# Task Instruction
{I}

# Task-State Representation
## Global Summary
{cumulative task state}

## Progress Tracker
Task Decomposition:
- {decomposition items}
Completed Progress:
- {completed items}
Current Subgoal: {current subgoal}
Remaining Requirements:
- {remaining items}

## Transition-Aware Focus
Last Step Result:
- Effectiveness: {effective/uncertain}
- Outcome: {last_step_result}
Next Action Focus: {next_action_focus}

# Output Format
Thought: [Your analysis of current state]
Action: [Single JSON action]
```

E State Updater Prompt Template

The state updater $\mathcal{U}_\phi$ receives the following prompt along with before/after screenshots (o_{t-1}, o_t) as image inputs.

System message.

```
You maintain a structured task-state
representation for a mobile GUI actor.
Return strict JSON only.
```

User message.

```
Task Instruction: {I}

Previous Task-State Representation (JSON):
{S_{t-1}}

Last Step:
- Actor reasoning: {T_{t-1}}
- Action executed: {a_{t-1}}
- Screenshot before action: [image o_{t-1}]
- Screenshot after action: [image o_t]

Instructions:
1. Compare the before/after screenshots
   to determine action effectiveness.
2. Update task decomposition if the task
   requires multiple steps.
3. Preserve all verified completed
```

```
   progress from previous state.
4. Update the current subgoal.
5. Align remaining requirements.
6. Write next_action_focus to guide
   the actor's next decision.

Rules:
- Treat the previous state as persistent
  working memory across steps.
- Base judgments only on visible UI state;
  do not invent hidden information.
- Keep each field concrete and relevant.
- Return JSON only, no explanation.

Output JSON schema:
{
  "action_effective": true|false|null,
  "task_summary": "...",
  "task_decomposition": ["..."],
  "completed_progress": ["..."],
  "current_subgoal": "...",
  "remaining_requirements": ["..."],
  "last_step_result": "...",
  "next_action_focus": "..."
}
```

F Case Study

Figure 2 illustrates a bookmark-management task where the baseline enters a repetition loop after a delayed UI update; the transition-aware focus detects the uncertain effect and guides the actor to verify before retrying. Figure 3 shows a failure mode: when the updater decomposes a task too aggressively, the actor inspects items individually instead of leveraging global visual inference, exceeding the step budget. The task-state representation should therefore be treated as guidance rather than an override of current-screen evidence.

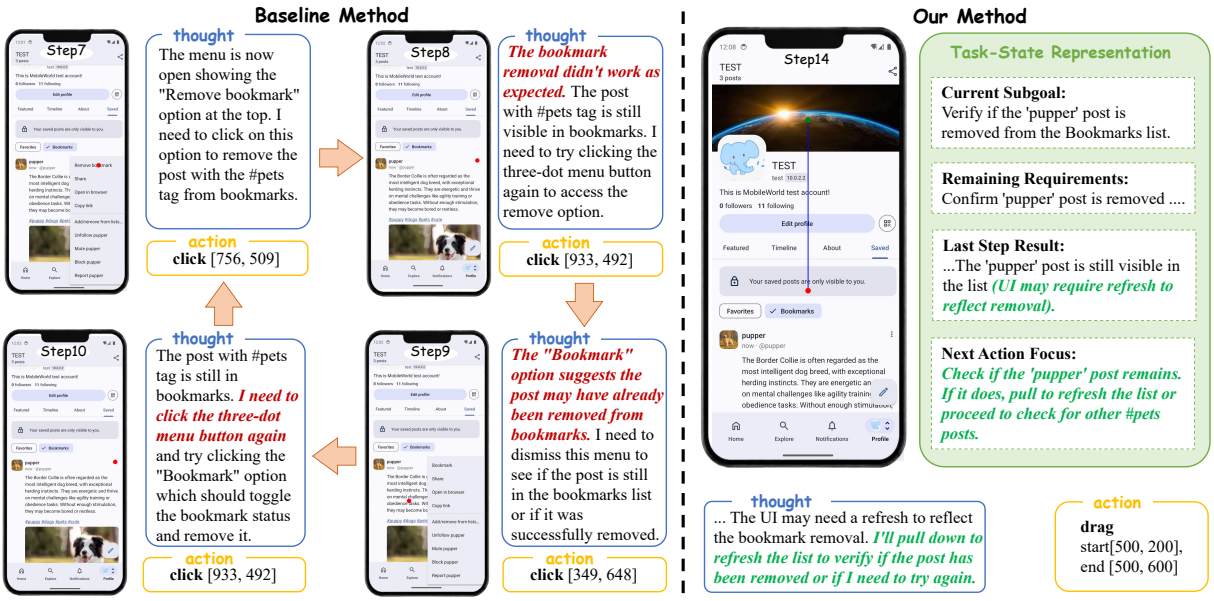

Figure 2: Stale-screen recovery. The baseline repeats the same removal action against a delayed UI update. The task-state representation detects the uncertain transition and redirects the actor to verify before retrying.

Figure 3: Over-decomposition failure. The state updater decomposes the task too aggressively, causing the actor to inspect items individually rather than using global visual inference, exceeding the step budget.