WM-R1: TRAINING GUI AGENTS TO REASON AND LEVERAGE WORLD MODELS WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI agents trained with reinforcement learning (RL) have showcased strong environment learning capabilities on mobile platforms. However, RL typically demands extensive real-environment interactions, leading to high resource costs and instability, especially in GUI scenarios. To address these, we propose WM-R1, the first reinforcement learning framework that trains mobile GUI agents with world models instead of real environments. Specifically, world models serve as the source of state transitions during all rollouts, replacing the real Android environment within the training loop. WM-R1 also embeds world models directly into the thinking process, enabling agents to reason about the consequences of candidate actions before committing to the final action. Crucially, WM-R1 eliminates the need for real-environment interaction, supports massively parallelized and step-level granularized trajectory generation grounded in world models, and introduces a multi-dimensional rule-based reward that jointly optimizes task success, trajectory efficiency, and world model utilization. For efficient training, we curate a high-quality dataset of 2000 challenging tasks. Experiments on Android mobile benchmarks demonstrate that WM-R1-trained agents significantly outperform GRPO-only baselines and inference-time simulation methods.

1 INTRODUCTION

Autonomous GUI agents that interact with mobile interfaces through visual perception and action execution have emerged as a promising direction for automating complex digital workflows (Qin et al., 2025; Hong et al., 2024; Rawles et al., 2023; Nguyen et al., 2025). Reinforcement learning (RL) has proven effective for training these agents. While PPO (Schulman et al., 2017) requires training a separate value network, GRPO (Shao et al., 2024) eliminates this overhead by computing advantages relative to a group of sampled trajectories, enabling emergent reasoning through rule-based rewards alone. UI-R1 (Lu et al., 2025c) was the first to apply GRPO to GUI action prediction, using a rule-based reward (format + action type + coordinate) without supervised reasoning traces. ARPO (Lu et al., 2025a) introduced an experience replay buffer into GRPO, reusing successful trajectories for long-horizon tasks. Recent works have extended this direction with multi-turn RL (Wang et al., 2025a), curriculum training (Nong et al., 2025) and self-evolutionary strategies (Yuan et al., 2025; Lian et al., 2026).

However, all existing RL methods require extensive real-environment interaction, introducing three bottlenecks: (1) *computational cost*: environment interaction is orders of magnitude slower than model inference; (2) *irreversibility*: a single wrong action derails long-horizon trajectories; (3) *environment noise*: non-deterministic transitions corrupt the reward signal. Parallel work has explored world models as a remedy. Text-based methods predict transitions via natural language (Chae et al., 2025) or DOM updates (Gu et al., 2024), capturing semantic intent but discarding spatial-visual details. Pixel-based approaches like ViMo (Luo et al., 2025a) and MobileDreamer (Cao et al., 2026) synthesize screenshots but struggle with structural controllability. Code-based methods such as Code2World (Zheng et al., 2026) generate renderable HTML, achieving both visual fidelity and structural precision. However, all these works use world models exclusively at inference time as a plug-and-play simulator that augments a pre-trained agent's decision-making, leaving the RL train-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

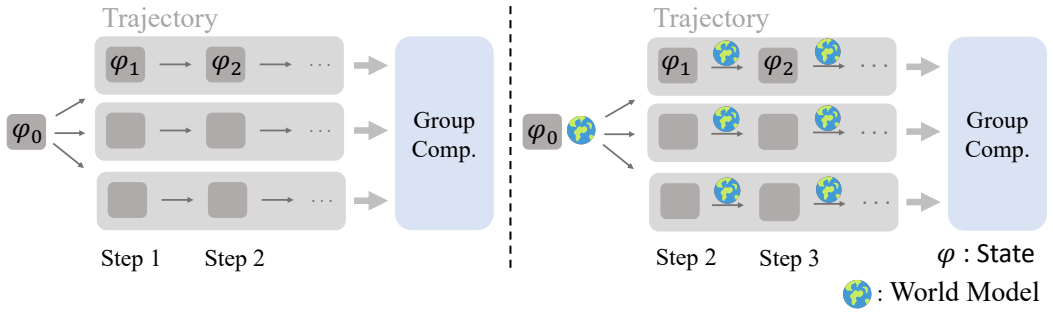


Figure 1: Comparison of training paradigms. Left: Standard GRPO generates trajectories through real-environment interaction, incurring high computational cost, environment latency, and irreversibility. Right: WM-R1 replaces the environment without world models, enabling fully simulated trajectory generation.

ing loop unchanged. The fundamental challenge remains: how to leverage the world model not just for inference, but as the training environment itself.

To address these challenges, we propose WM-R1, the first reinforcement learning framework that trains mobile GUI agents entirely within a simulated environment, namely world models. As shown in Figure 1 (right), WM-R1 replaces the real Android environment with world models that predict next GUI states via simulated screenshots. This substitution has two critical consequences: (1) training becomes massively parallelizable and step-level granularized with environmental interaction overhead, and (2) the agent's chain-of-thought reasoning naturally incorporates world model predictions, since every state transition it observes comes from the world model. Specifically, WM-R1 embeds the world model directly into the agent's reasoning stream through a `<call_wm>` mechanism, allowing the agent to simulate and refine candidate actions before committing to final outputs.

Our approach uses Qwen2.5-VL-3/7B (Bai et al., 2025b) as the base model and trains on a curated dataset combining three Android GUI sources: AndroidCode (Zheng et al., 2026), GUI-Odyssey (Lu et al., 2025b), and GUI-R1 (Luo et al., 2025b). We filter by task difficulty. The training loop follows GRPO (Shao et al., 2024) with a composite reward (success + length + WM call count) to drive the policy update. Our contributions are:

1. WM-R1 framework: A GRPO-based training framework that replaces the real Android environment with a learned world model, enabling fully simulated, massively parallel training of mobile GUI reasoning agents without any real-environment interaction.
2. World model utilization: We demonstrate that WM-R1 substantially improves GUI agents' ability to reason with world models on mobile platforms.
3. Multi-dimensional reward design: We propose a rule-based composite reward that aligns with GUI task objectives across task success, trajectory efficiency, and world model utilization, with ablation studies confirming each component's contribution.
4. Experiments on mobile GUI benchmarks: We train on the curated Android dataset and evaluate on AndroidWorld, GUI-Odyssey, ScreenSpot-Pro/V2, and AndroidControl, demonstrating that WM-R1-trained agents outperform GRPO baselines at both 3B and 7B model scales.

2 Related Work

GUI agents. Mobile GUI agents perform multi-step interactions on Android devices from visual input (Nguyen et al., 2025; Wang et al., 2024; Tang et al., 2025). The prevailing approach trains vision-language models on large-scale state–action trajectories via SFT. For GUI grounding, CogAgent (Hong et al., 2024) introduced spatial-aware visual understanding, SeeClick (Cheng et al., 2024) provided explicit grounding supervision, and UGround (Gou et al., 2025) achieved cross-platform universal grounding. For unified action prediction, ShowUI (Lin et al., 2025) combined

vision-language-action modeling, and UI-TARS (Qin et al., 2025) added reasoning-oriented fine-tuning. Mobile-domain extensions include Mobile-Agent-v3 (Ye et al., 2025), Aria-UI (Yang et al., 2025), and MAI-UI (Zhou et al., 2025b), which target the unique challenges of Android interaction such as app diversity, dynamic layouts, and touch-based navigation. GUI-Libra (Yang et al., 2026) introduced RL-augmented supervision for grounding tasks.

Reinforcement learning for GUI agents. PPO (Schulman et al., 2017; Ouyang et al., 2022; Yao et al., 2022; Wei et al., 2022) requires a value network; GRPO (Shao et al., 2024) computes group-relative advantages instead. After DeepSeek-R1 (Guo et al., 2025) showed emergent reasoning from pure GRPO, the approach was extended to VLMs (Huang et al., 2025; Zhou et al., 2025a; Chen et al., 2025). In GUI, UI-R1 (Lu et al., 2025c) first applied rule-based GRPO without reasoning traces. Follow-ups explored multi-turn RL (Wang et al., 2025a; Liu et al., 2025), experience replay (Lu et al., 2025a), curriculum (Nong et al., 2025), decoupled training (Li et al., 2025b), mobile-specific RL (Shi et al., 2025; Xu et al., 2025), self-evolutionary grounding (Yuan et al., 2025), efficient training (He et al., 2025), and alternative formulations (Lian et al., 2026; Yang et al., 2026). Similar ideas appeared in web navigation (Wei et al., 2025) and environment-free training (Li et al., 2025d). However, all these methods rely on real-environment interaction, which is costly and unstable. WM-R1 instead uses world models as the training environment, eliminating the need for real environment.

World models for GUI. World models enable action-conditioned future prediction for planning (Ha & Schmidhuber, 2018; Ding et al., 2025; Li et al., 2025c). GUI approaches fall into three categories: *text-based* (language/DOM prediction (Chae et al., 2025; Gu et al., 2024)), *pixel-based* (diffusion-generated screenshots (Luo et al., 2025a; Cao et al., 2026)), and *code-based* (renderable HTML (Zheng et al., 2026)). UI-Simulator (Wang et al., 2025b) and SimuRA (Deng et al., 2025) use world models for agent training simulation. All, however, operate at *inference time* only (Zheng et al., 2026; Luo et al., 2025a). WM-R1 instead uses the world model as the *training environment*.

3 PRELIMINARIES

At step t in standard RL training, a GUI agent π_θ parameterized by θ observes a screenshot φ_t and interaction history $h_{t-1} = (\varphi_0, a_0, \ldots, \varphi_{t-1}, a_{t-1})$, and produces an action $a_t \in \mathcal{A}$ (e.g., click, type, scroll). The environment transitions to a new state $\varphi_{t+1} \sim \mathcal{T}(\cdot \mid \varphi_t, a_t)$, yielding a sparse reward $r_t \in \{0, 1\}$. Training proceeds via Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which eliminates the value network by computing advantages relative to a group of G sampled trajectories. For each trajectory τ_i with reward r_i, the advantage is:

$$A_i = \frac{r_i - \mu_r}{\sigma_r}, \tag{1}$$

where μ_r and σ_r are the mean and standard deviation of rewards within the group. The policy update maximizes:

$$\mathcal{J}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^{G} \min \Big(\rho_i A_i, \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon) A_i \Big) - \lambda \cdot \text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}}) \right], \tag{2}$$

where $\rho_i = \pi_\theta(\tau_i)/\pi_{\theta_{\text{old}}}(\tau_i)$ is the importance sampling ratio and λ controls the KL divergence penalty from a reference policy π_{ref}. WM-R1 replaces the real environment with world models $\mathcal{W}$ making the simulator an integral component in the training loop. Concretely, this substitution enables two key capabilities beyond standard GRPO: (1) *reasoning with simulated outcomes*: the agent invokes $\mathcal{W}$ during its chain-of-thought to preview candidate actions, observe the predicted next state $\hat{\varphi}_{t+1}$, and refine its decision before committing to a final action a_t; (2) *fully simulated state transformation*: the entire trajectory τ_i is generated via state transformation $\hat{\varphi}_{t+1} \sim \mathcal{W}(\cdot \mid \varphi_t, a_t)$.

4 METHOD: WM-R1

As illustrated in Figure 2, WM-R1 trains a GUI agent through GRPO in a fully simulated environment powered by a world model. This substitution enables three things: (1) the agent learns to reason using world model predictions during its chain-of-thought (Section 4.1), (2) trajectory generation is massively parallelized, producing states without requiring real environments (Section 4.2), and (3) the reward design can focus on reasoning performance and action correctness (Section 4.3).

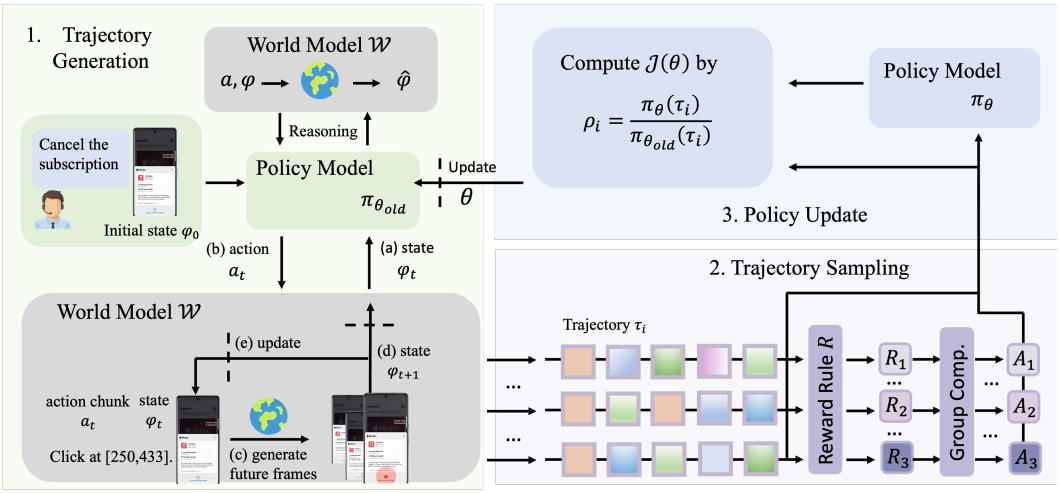


Figure 2: WM-R1 training framework. For each task, the policy model generates multiple candidate trajectories with chain-of-thought reasoning in a world-model-simulated environment. At each reasoning step, the agent can invoke the world model to predict the consequence of a candidate action. The simulated next state becomes part of the trajectory. After all trajectories complete, a rule-based Reward Model computes scores, followed by GRPO advantage computation and policy update.

4.1 Reasoning with the World Model

The core of WM-R1 is how the agent learns to invoke and reason with the world model during its chain-of-thought process. The policy model π_θ is prompted to produce responses in a structured format that interleaves reasoning tokens with actions. The system prompt is shown below:

Prompt for Training and Inference
In this GUI screenshot, I want to perform the command *instruction*. Please provide the action to perform and the coordinate where the cursor is moved to if click is performed. Output the thinking process in thinking tags and final answer in answer tags. Within the thinking process, you may invoke <call_wm>to simulate the outcome of an action before committing. Please strictly follow the format.

The world model is integrated into the reasoning process through a `<call_wm>` invocation. Unlike approaches that treat the world model as a separate conversational turn, WM-R1 embeds world model calls *within* the agent's reasoning stream. At each reasoning step, the agent can decide to invoke the world model to simulate the consequence of a proposed action:

$$\hat{\varphi}_{t+1} \sim \mathcal{W}(\cdot \mid \hat{\varphi}_t, a_t). \tag{3}$$

The simulated screenshot $\hat{\varphi}_{t+1}$ is then injected back as a *System (WM)* observation, and the agent continues reasoning from this point within the same thinking block. This multi-turn reasoning pattern allows the agent to interleave thinking and world model calls within a single thinking block, then produce a final action once it is confident.

4.2 Trajectory Generation

Each training trajectory is generated through an iterative loop between the policy model and the world model. Given a task instruction I and an initial screenshot φ_0, the trajectory generation proceeds as follows:

1. Observe. The agent receives the current simulated screenshot $\hat{\varphi}_t$ along with the full interaction history $h_t = (\varphi_0, a_0, \hat{\varphi}_1, a_1, \ldots, \hat{\varphi}_t)$.

2. Think. The policy model π_θ generates a reasoning block with optional <call_wm> invocations:
$$\text{thought}_t \sim \pi_\theta(\hat{\varphi}_t, I, h_t). \tag{4}$$
3. Simulate & Continue. If <call_wm> is present, the world model generates the next simulated screenshot, which is fed back as a **System (WM)** observation. The agent continues reasoning from this point. If no <call_wm> is present, the reasoning block ends and the agent produces a final action:
$$a_t \sim \pi_\theta(\hat{\varphi}_t, I, h_t, \text{thought}_t). \tag{5}$$
4. Repeat. Steps 1–3 continue until the agent produces a terminal action or the trajectory reaches the maximum length $T_{\max}$.

The complete trajectory for task i and rollout j is:
$$\tau_{i,j} = (\hat{\varphi}_0, a_0, \hat{\varphi}_1, a_1, \ldots, \hat{\varphi}_T, a_T), \tag{6}$$
The complete WM-R1 training loop is summarized in Algorithm 1 (Appendix A.2).

4.3 Reward Design

The reward function drives the GRPO optimization and determines what behaviors the agent learns. We adopt a composite reward design that combines a task success signal with a length penalty and a world model efficiency bonus:
$$R = \alpha \cdot R_{\text{success}} + \beta \cdot R_L + \gamma \cdot R_{\text{WM}}. \tag{7}$$

Success reward R_{success}. The success reward evaluates whether the agent completes the task. We use LLM-as-a-judge to compare the final trajectory against the reference solution. The reward is binary:
$$R_{\text{success}} = \begin{cases} 1, & \text{task succeeds (exact match)} \\ 0, & \text{task fails} \end{cases} \tag{8}$$
Exact match is determined by the LLM judge evaluating whether the sequence of actions and their effects are semantically equivalent to the reference trajectory.

Length reward R_L. The length reward encourages efficiency by penalizing unnecessarily long trajectories. Following DAST adopted in UI-R1 (Shen et al., 2025; Lu et al., 2025c), we define a length budget L_{budget} as a mixture of the reference trajectory length $L_{\bar{r}}$ and the maximum allowed length $L_{\max}$:
$$L_{\text{budget}} = p \cdot L_{\bar{r}} + (1-p) \cdot L_{\max}, \qquad p = \frac{c}{N}, \tag{9}$$
where c is the number of completed sub-tasks and N is the total number of sub-tasks. The normalized deviation $\lambda = (L_i - L_{\text{budget}})/L_{\text{budget}}$ is then used to compute the length reward:
$$R_L = \begin{cases} \max(-0.5\lambda + 0.5,\ 0.1) & \text{if task succeeds} \\ \min(0.9\lambda - 0.1,\ -0.1) & \text{if task fails} \end{cases} \tag{10}$$

World model call count reward R_{WM}. The WM call reward regulates the agent's usage of world model simulation. Let $N_{\text{WM}}^{(i)}$ denote the number of <call_wm> invocations in trajectory i. The group average is $\bar{N}_{\text{WM}} = \frac{1}{G}\sum_{j=1}^{G} N_{\text{WM}}^{(j)}$. The normalized deviation $\mu = (N_{\text{WM}}^{(i)} - \bar{N}_{\text{WM}})/\bar{N}_{\text{WM}}$ is used to compute:
$$R_{\text{WM}} = \begin{cases} \max(-0.5\mu + 0.5,\ 0.1) & \text{if task succeeds} \\ \min(0.9\mu - 0.1,\ -0.1) & \text{if task fails} \end{cases} \tag{11}$$

For each task, the G trajectory rewards are used to compute GRPO advantages via Eq. (1). The relative nature of GRPO advantages means that the agent learns to produce trajectories that are better than the group average, which naturally selects for more accurate and efficient reasoning strategies. Through GRPO training, the agent learns *when* and *how many times* to invoke the world model: trajectories that use simulation effectively to refine their actions receive higher rewards, while unnecessary or poorly timed world model calls do not contribute positively. Over training, the agent internalizes a policy that balances reasoning depth with action efficiency.

Table 1: Main results across all benchmarks. Bold: best per scale. AndroidWorld and GUI-Odyssey report success rate (%); Hard subsets contain tasks requiring $\geq$5 ground-truth steps. ScreenSpot-Pro/V2 report mobile grounding accuracy (%). AndroidControl reports action type prediction (Type) and coordinate grounding (Gnd) accuracy (%). CW: Code2World inference-time augmentation.

	AndroidWorld		GUI-Odyssey		ScreenSpot-Pro		ScreenSpot-V2		AndroidControl	
Method	**Full**	**Hard**	**Full**	**Hard**	**Icon**	**Text**	**Icon**	**Text**	**Type**	**Gnd**
3B scale										
Qwen2.5-VL-3B (ZS)	15.2	11.8	8.5	5.2	28.4	38.6	64.5	88.2	76.2	69.5
Qwen2.5-VL-3B + CW	20.6	16.3	14.2	9.8	31.2	41.5	67.8	89.1	78.5	72.3
GUI-R1-3B	17.4	13.5	10.2	6.8	35.6	44.3	76.8	90.5	82.3	75.6
UI-TARS-3B	19.8	15.6	13.4	8.9	39.2	48.5	80.4	91.8	84.7	77.8
UI-R1-3B	18.6	14.2	11.3	7.4	38.7	48.2	81.2	92.5	85.6	78.4
WM-R1-3B	**29.5**	**23.1**	**22.8**	**16.4**	**43.5**	**52.8**	**83.6**	**93.8**	**89.2**	**82.4**
7B scale										
Qwen2.5-VL-7B (ZS)	22.4	17.6	12.3	7.8	35.2	44.8	72.3	91.4	79.8	74.1
Qwen2.5-VL-7B + CW	28.7	23.1	19.6	14.2	38.6	48.1	76.8	92.1	82.5	77.6
GUI-R1-7B	24.5	19.2	15.8	10.5	40.2	49.6	82.5	92.8	85.2	78.5
UI-TARS-7B	27.3	21.5	18.2	12.8	43.5	53.2	85.8	93.5	87.6	80.4
Aguvis-7B	26.8	20.9	17.5	12.1	44.2	54.5	86.5	94.2	88.2	81.5
OS-Genesis-7B	29.2	22.8	19.8	14.5	41.8	51.2	84.2	93.5	86.5	79.8
UI-R1-7B	30.8	24.2	21.5	15.3	42.8	52.4	84.5	93.6	87.4	81.2
MobileGUI-RL-7B	30.0	23.5	20.8	15.1	40.5	50.2	82.1	92.8	85.8	79.5
OS-Atlas-7B	33.5	27.2	24.8	19.2	45.8	55.6	88.2	94.8	90.5	84.2
UI-TARS 1.5-7B	34.2	28.1	25.6	20.1	46.2	56.8	89.5	95.1	91.2	85.5
WM-R1-7B	**39.8**	**32.7**	**31.6**	**24.1**	**48.6**	**58.2**	**90.4**	**95.2**	**93.5**	**87.8**

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets. We train the Qwen2.5-VL-3/7B model (Bai et al., 2025b) on a curated dataset combining AndroidCode (Zheng et al., 2026), GUI-Odyssey (Lu et al., 2025b), and GUI-R1 (Luo et al., 2025b). Tasks with zero-shot success rates above 80% (trivially easy) or below 5% (impossibly hard) are removed, and 2000 examples are randomly sampled, ensuring the training distribution focuses on intermediate-complexity tasks where world-model feedback is most informative.

Benchmarks. We evaluate on two categories of benchmarks. *Long-horizon tasks*: AndroidWorld (Rawles et al., 2023), testing multi-step mobile device control with a maximum execution length of 15; and GUI-Odyssey (Lu et al., 2025b), evaluating cross-app navigation on Android. For both AndroidWorld and GUI-Odyssey, we define a *Hard* subset consisting of tasks that require $\geq$5 ground-truth steps to complete. *Action prediction and grounding*: ScreenSpot-Pro (Li et al., 2025a) and ScreenSpot-V2 (Wu et al., 2025) evaluate GUI grounding accuracy on the mobile subset; AndroidControl (Rawles et al., 2023) measures action type prediction and coordinate grounding.

Implementation details. We use the pretrained Code2World-8B model (Zheng et al., 2026), frozen during training. It generates renderable HTML from the current screenshot and action, rendered to a 1080×2400 screenshot using a headless browser. The results for Qwen3-VL and other details are listed in the Appendix.

5.2 MAIN RESULTS

Long-horizon tasks. The results in Table 1 demonstrate the advantages of world-model-simulated training for long-horizon mobile tasks. WM-R1 establishes new state-of-the-art results at both model scales. WM-R1-3B (29.5) substantially outperforms all 3B-scale baselines including UI-R1-3B (Lu et al., 2025c) (18.6) and GUI-R1-3B (Luo et al., 2025b) (17.4). At 7B scale, WM-R1-7B reaches 39.8 on AndroidWorld, outperforming UI-R1-7B (Lu et al., 2025c) (30.8) by +9.0 points and UI-TARS 1.5-7B (34.2) by +5.6 points. On GUI-Odyssey, WM-R1-7B achieves 31.6, surpassing UI-R1-7B (21.5) by +10.1 points and OS-Atlas-7B (24.8) by +6.8 points.

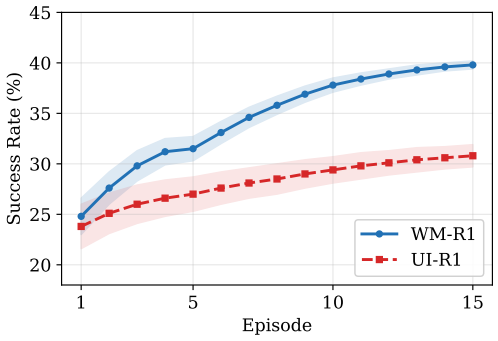


Figure 3: Training curves. WM-R1 reaches 39.8% success rate by episode 15, outperforming UI-R1 (30.8%) by 9.0 points with lower variance.

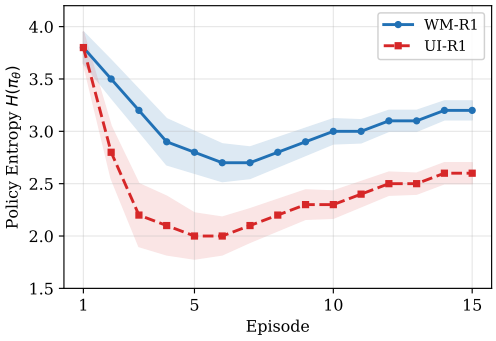


Figure 4: Policy entropy. Both methods show a decrease-then-recover pattern, but WM-R1 maintains higher entropy throughout, indicating sustained exploration from the noise-free world model.

WM-R1 achieves substantial gains over the inference-time augmentation baseline (Qwen2.5-VL + CW, i.e., using Code2World for propose-simulate-select at inference only) at both scales: +8.9 and +11.1 on AndroidWorld for 3B and 7B respectively. The improvements are particularly strong on the Hard subsets (+6.8 at 3B, +9.6 at 7B for AndroidWorld; +6.6 at 3B, +9.9 at 7B for GUI-Odyssey), indicating that world-model-simulated transitions provide especially strong signal for difficult, multi-step reasoning tasks that require multiple steps.

A key question is whether WM-R1's improvements stem from memorizing training patterns or from learning generalizable reasoning strategies. To investigate this, we categorize our evaluation benchmarks into in-distribution (ID) and out-of-distribution (OOD) settings relative to the training data. AndroidControl and GUI-Odyssey serve as ID benchmarks while AndroidWorld and ScreenSpot-Pro/V2 are OOD. WM-R1 achieves an average OOD improvement of +13.9 relative to the ZS baseline, which is *1.80*× that of UI-R1 (+7.7). This demonstrates that world-model-based reasoning delivers stronger generalization to novel environments through two complementary mechanisms: (i) World-model-based reasoning over memorization. Standard GRPO trains the agent to produce correct actions on the training distribution. When faced with unfamiliar UI elements or app combinations, the agent must rely on surface-level pattern matching, which degrades rapidly under distribution shift. WM-R1, by contrast, trains the agent to use the world model as an internal simulator: before committing to an action, the agent proposes a candidate, simulates its consequence via `<call_wm>`, and revises if the predicted outcome does not match the intent. This "think-before-act" loop is a *reasoning strategy*, not a memorized mapping. It transfers naturally to OOD situations because the agent can simulate and evaluate actions even on UI patterns it has never seen during training. (ii) Diverse training distribution. The world model enhances the visual and structural diversity of the training distribution: by simulating the consequences of actions on diverse screenshots, the agent effectively trains on a broader and more novel distribution of state transitions than the raw dataset provides. Together, these mechanisms explain why WM-R1 is expected to close the gap between ID and OOD performance more effectively than GRPO-only baselines: the agent learns *how to reason with a world model* rather than *what actions to take on specific UI patterns*.

Figure 3 shows the training dynamics over 15 episodes on AndroidWorld. Both methods start from the same zero-shot baseline (22.4%), but WM-R1 exhibits a steeper learning curve, reaching 31.5% by episode 5. By episode 15, WM-R1 reaches 39.8%, a 9.0 point advantage over UI-R1. WM-R1 also shows lower variance across seeds, which we attribute to the fully simulated environment eliminating the stochasticity inherent in real GUI interactions. The policy entropy $H(\pi_\theta)$ (Figure 4) reveals the exploration mechanism behind this advantage: both methods exhibit an initial entropy drop followed by recovery, but WM-R1's entropy decreases much more gradually (3.8→2.9 vs. UI-R1's 3.8→2.1 in the first 3 episodes). This sustained exploration is enabled by the noise-free world model, which avoids the stochastic punishment of exploratory actions that causes real-environment agents to prematurely collapse. At convergence, WM-R1 retains higher entropy (∼3.2 vs. ∼2.6).

Table 2: Ablation study on AndroidWorld (7B). Chain-of-thought reasoning is the most critical component; removing it causes the largest degradation (-5.2 on AW, -4.9 on AW-H). R_L and R_{WM} contribute smaller but complementary gains.

Config	AW	AW-H	Δ
WM-R1 (full)	**39.8**	**32.7**	—
− CoT	34.6	27.8	−5.2 / −4.9
− R_L	36.5	30.1	−3.3 / −2.6
− R_{WM}	37.2	30.8	−2.6 / −1.9

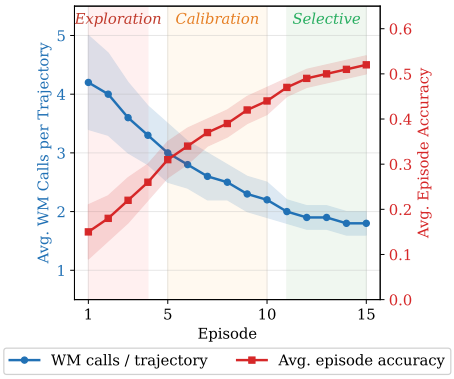


Figure 5: WM utilization dynamics.

Action prediction and grounding. WM-R1 delivers substantial improvements across all mobile grounding benchmarks. At 3B scale, WM-R1-3B improves icon grounding on ScreenSpot-Pro by +15.1 points (43.5 vs. 28.4) and text grounding by +14.2 points (52.8 vs. 38.6) over the zero-shot baseline, while surpassing UI-R1-3B (Lu et al., 2025c) across all metrics. On AndroidControl, WM-R1-3B achieves 89.2 action type accuracy and 82.4 coordinate grounding (avg 85.8), exceeding UI-R1-3B (85.6 / 78.4, avg 82.0) by +3.8 points. At 7B scale, WM-R1-7B further extends these gains: ScreenSpot-Pro icon reaches 48.6 (+13.4 over ZS), ScreenSpot-V2 icon reaches 90.4 (+18.1 over ZS), and AndroidControl grounding reaches 87.8 (+13.7 over ZS). The grounding improvement is driven by the world model's dense visual feedback: by simulating the consequence of each candidate action, the agent learns to associate visual features with spatial locations more precisely.

5.3 Ablation Studies

To understand the contribution of each component, we conduct an ablation study on AndroidWorld (7B scale). We evaluate three variants of WM-R1, each removing a single design choice from the full system.

Table 2 reveals the relative importance of WM-R1's components. Chain-of-thought reasoning is the most impactful, with a 5.2 point degradation on AndroidWorld and 4.9 on the Hard subset when removed. Without intermediate reasoning, the model produces actions directly from the observation, bypassing the reflective loop where the world model's simulated feedback can be used to evaluate and revise candidate actions before execution. The length reward R_L contributes a 3.3 point gain: without it, the agent learns to solve tasks correctly but with redundant steps that accumulate error over longer trajectories. The WM call reward R_{WM} has the smallest individual impact (-2.6 on the full set, -1.9 on Hard), but its role is stabilizing: without regulation, some training trajectories converge to excessive world model calls that waste inference budget, while others underuse the simulator entirely. Together, the three components address distinct failure modes, and the full system achieves a 2.6 point advantage over the strongest single-component variant.

5.4 World Model Utilization Dynamics

A natural question is how the agent's use of the world model evolves during training. We track two quantities across training episodes: (1) the average number of `<call_wm>` invocations per trajectory, and (2) the average episode reward.

As shown in Figure 5, a clear inverse relationship emerges: as training progresses, the agent makes *fewer* world model calls while achieving *higher* rewards. We identify three phases of world model mastery:

Exploration phase (episodes 1–4): The agent invokes the world model frequently ($\sim$4 calls per trajectory), often redundantly simulating obvious actions. Many of these calls produce predictions that the agent could have anticipated without simulation, reflecting an undifferentiated reliance on the world model.

Calibration phase (episodes 5–10): The agent begins to distinguish between actions that benefit from simulation (e.g., clicking on ambiguous UI elements, navigating to unfamiliar screens) and actions that do not (e.g., typing text, pressing the home button). World model calls drop to $\sim$2.5 per trajectory as the agent learns to allocate simulation budget strategically.

Selective utilization phase (episodes 11–15): The agent invokes the world model primarily at critical decision points—when it is uncertain about the consequence of an action or when a mistake would derail the remaining trajectory. Calls stabilize at $\sim$1.8 per trajectory, indicating that the agent has internalized a policy of *targeted simulation*: using the world model where it provides the most value while relying on its own learned policy for routine actions.

This dynamic is consistent with the role of R_{WM} in our composite reward design: by penalizing both over-use and under-use relative to the group average, the reward guides the agent toward an efficient equilibrium.

5.5 Case Study: World-Model-Guided Self-Correction

To illustrate how WM-R1 agents leverage the world model during inference, we present a representative reasoning trace requiring the agent to click a search bar and enter a query.

```
<think>
I need to click the search bar to enter the query.  It appears on
the left side of the top bar.  Let me simulate clicking there.
<call_wm>(click(0.7, 0.3))
</think>

System (WM): [simulated screenshot φ̂_{t+1}:  nothing changed --- the
click landed on an empty area to the right of the search bar]

<think>
The click did not hit the target.  The coordinate was too far to
the right.  The search bar starts around x=0.4.  Let me adjust.
<call_wm>(click(0.5, 0.3))
</think>

System (WM): [simulated screenshot φ̂_{t+2}:  search bar focused with
cursor visible]

<think>
The search bar is now focused and the cursor is active.  This is
the correct action.
</think>

Action:  click(0.5, 0.3)
```

The agent demonstrates a three-stage reasoning process: (1) Propose. It identifies the search bar and proposes an initial click coordinate; (2) Simulate and Evaluate. It invokes the world model to preview the consequence, and recognizes that the click landed on an empty area; (3) Revise and Confirm. It adjusts the coordinate, simulates again, and only commits once the predicted state matches the intent. This self-correction loop is impossible for GRPO-only baselines, which must execute actions directly and suffer irreversible consequences from initial mistakes. The world model effectively serves as a "scratchpad" that allows the agent to make and learn from mistakes before acting in the environment.

6 Conclusion

We presented WM-R1, a reinforcement learning framework that trains GUI agents entirely within a world-model-simulated environment. Rather than using the world model only at inference time as a post-hoc proposal-evaluation tool, WM-R1 uses world models as the source of all state transitions during GRPO training. This design enables massively parallel, zero-cost training while allowing the agent's chain-of-thought reasoning to naturally incorporate simulated visual feedback at every step. Our experiments on mobile benchmarks demonstrate that WM-R1 improves over zero-shot and GRPO baselines, confirming that world-model-simulated training provides a practical and effective path toward generalizable mobile GUI reasoning agents.

AI-USE STATEMENT

During the preparation of this manuscript, the authors used a large language model solely to polish the language of the text. All scientific content, experiments, and conclusions were produced by the authors, who take full responsibility for the integrity of the work.

REPRODUCIBILITY STATEMENT

We provide the following resources to facilitate reproduction of our results. Implementation details: the complete WM-R1 training loop is given in Algorithm 1 (Appendix A.2), and the full hyperparameter configuration, distributed training setup, world model configuration, and reward function details are reported in Appendix A.1. Models and data: we use publicly available base models (Qwen2.5-VL-3B/7B), reward model (GPT-4o) and the pretrained Code2World-8B world model; the training data is curated from three public Android GUI datasets (AndroidCode, GUI-Odyssey, GUI-R1), with the curation procedure described in Appendix A.1. Evaluation: all benchmarks (AndroidWorld, GUI-Odyssey, ScreenSpot-Pro/V2, AndroidControl) are public, and we follow their official evaluation protocols as detailed in Appendix A.1. Code: code and training scripts will be made publicly available upon acceptance.

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, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025a. URL https://arxiv.org/abs/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, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Yilin Cao, Yufeng Zhong, Zhixiong Zeng, Liming Zheng, Jing Huang, Haibo Qiu, Peng Shi, Wenji Mao, and Wan Guanglu. Mobiledreamer: Generative sketch world model for gui agent. *arXiv preprint arXiv:2601.04035*, 2026.

Hyungjoo Chae, Namyoung Kim, Kai Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations*, volume 2025, pp. 63707–63738, 2025.

Yan Chen, Long Li, Teng Xi, Long Zeng, and Jingdong Wang. Perception before reasoning: Two-stage reinforcement learning for visual reasoning in vision-language models. *arXiv preprint arXiv:2509.13031*, 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 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)*, pp. 9313–9332, 2024.

Mingkai Deng, Jinyu Hou, Zhiting Hu, and Eric Xing. Simura: A world-model-driven simulative reasoning architecture for general goal-oriented agents. *arXiv preprint arXiv:2507.23773*, 2025.

Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, et al. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, 58(3):1–38, 2025.

Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. In *International Conference on Learning Representations*, volume 2025, pp. 30851–30883, 2025.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, et al. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Yanheng He, Jiahe Jin, and Pengfei Liu. Efficient agent training for computer use. *arXiv preprint arXiv:2505.13909*, 2025.

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

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 8778–8786, 2025a.

Pengxiang Li, Zechen Hu, Zirui Shang, Jingrong Wu, Yang Liu, Hui Liu, Zhi Gao, Chenrui Shi, Bofei Zhang, Zihao Zhang, et al. Efficient multi-turn rl for gui agents via decoupled training and adaptive data curation. *arXiv preprint arXiv:2509.23866*, 2025b.

Xinqing Li, Xin He, Le Zhang, Min Wu, Xiaoli Li, and Yun Liu. A comprehensive survey on world models for embodied ai. *arXiv preprint arXiv:2510.16732*, 2025c.

Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025d.

Shuquan Lian, Yuhang Wu, Jia Ma, Yifan Ding, Zihan Song, Bingqi Chen, Xiawu Zheng, Hui Li, and Rongrong Ji. Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8575–8584, 2026.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19498–19508, 2025.

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

Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Arpo: End-to-end policy optimization for gui agents with experience replay. *arXiv preprint arXiv:2505.16282*, 2025a.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22404–22414, 2025b.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning, 2025c. URL https://arxiv.org/abs/2503.21620.

Dezhao Luo, Bohan Tang, Kang Li, Georgios Papoudakis, Jifei Song, Shaogang Gong, Jianye Hao, Jun Wang, and Kun Shao. Vimo: A generative visual gui world model for app agents. *arXiv preprint arXiv:2504.13936*, 2025a.

Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025b.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, et al. Gui agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22522–22538, 2025.

Songqin Nong, Xiaoxuan Tang, Jingxuan Xu, Sheng Zhou, Jianfeng Chen, Tao Jiang, and Wenhao Xu. Craft-gui: Curriculum-reinforced agent for gui tasks. *arXiv preprint arXiv:2508.11360*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

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

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2322–2331, 2025.

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*, 2025.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, et al. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*, 2025.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025a.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, et al. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*, 2024.

Yiming Wang, Da Yin, Yuedong Cui, Ruichen Zheng, Zhiqian Li, Zongyu Lin, Di Wu, Xueqing Wu, Chenchen Ye, Yu Zhou, et al. Llms as scalable, general-purpose simulators for evolving digital agent training. *arXiv preprint arXiv:2510.14969*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7920–7939, 2025.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: Foundation action model for generalist gui agents. In *International Conference on Learning Representations*, volume 2025, pp. 5090–5108, 2025.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. Mobilerl: Online agentic reinforcement learning for mobile gui agents. *arXiv preprint arXiv:2509.18119*, 2025.

Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baoling Peng, Huan Zhang, Jianfeng Gao, et al. Gui-libra: Training native gui agents to reason and act with action-aware supervision and partially verifiable rl. *arXiv preprint arXiv:2602.22190*, 2026.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22418–22433, 2025.

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

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, et al. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*, 2025.

Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025.

Yuhao Zheng, Li'an Zhong, Yi Wang, Rui Dai, Kaikui Liu, Xiangxiang Chu, Linyuan Lv, Philip Torr, and Kevin Qinghong Lin. Code2world: A gui world model via renderable code generation. *arXiv preprint arXiv:2602.09856*, 2026.

Guanghao Zhou, Panjia Qiu, Cen Chen, Jie Wang, Zheming Yang, Jian Xu, and Minghui Qiu. Reinforced mllm: A survey on rl-based reasoning in multimodal large language models. *arXiv preprint arXiv:2504.21277*, 2025a.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025b.

A APPENDIX

A.1 ADDITIONAL TRAINING DETAILS

Hyperparameters. Table 3 lists the full hyperparameter configuration used in our experiments.

Table 3: WM-R1 training hyperparameters.

Hyperparameter	Value
Base model	Qwen2.5-VL-3B / 7B
World model	Code2World-8B (frozen)
GPUs per node	$1 \times$ NVIDIA H200 (141GB)
Number of nodes	1
World model instances	1
Rollout batch size	2
Group size (G)	8
Max prompt length	64,000
Max response length	8,192
Max pixels	2,116,800
Learning rate	1×10^{-6}
Learning rate schedule	Cosine decay
Warmup ratio	0.05
Gradient clip norm	1.0
PPO epochs	1
Clip ratio (low)	0.2
Clip ratio (high)	0.3
Max trajectory length	15
Total episodes	15
KL penalty λ	0.01
Reward: α (success)	1.0
Reward: β (length)	0.5
Reward: γ (WM call)	0.1
GPU memory utilization	0.6
Temperature	1.0

Distributed training setup. We use Ray for distributed coordination. Each world model instance runs on a separate Ray actor. The actor-rollout worker group uses FSDP for memory-efficient training, with vLLM handling sequence generation during the rollout phase. The vLLM engine is put into sleep mode between generation phases to reduce peak memory usage.

World model configuration. We use the pretrained Code2World-8B model (Zheng et al., 2026), frozen during training. It generates renderable HTML from the current screenshot and action, rendered to a 1080×2400 screenshot using a headless browser.

Data curation. We curate training data by combining three Android GUI datasets: AndroidCode (Zheng et al., 2026), GUI-Odyssey (Lu et al., 2025b), and GUI-R1 (Luo et al., 2025b). All three datasets are publicly available. We filter by task difficulty: tasks with zero-shot success rates above 80% (trivially easy) or below 5% (impossibly hard) are removed. From the filtered pool, we randomly sample 2000 examples, ensuring the training distribution focuses on intermediate-complexity tasks where world-model feedback is most informative.

Reward function details. Our reward function combines three components as described in the main paper:

- **Success reward** R_{success} (+1 if the task is completed as judged by an LLM comparing the final trajectory against the reference solution; 0 otherwise), weighted by $\alpha = 1.0$.

- **Length reward** R_L encourages efficient trajectories by penalizing unnecessarily long ones, following the DAST approach (Shen et al., 2025; Lu et al., 2025c), weighted by $\beta = 0.5$. The length budget L_{budget} is a mixture of the reference trajectory length and the maximum allowed length, with the reward computed based on the normalized deviation from this budget.
- **WM call reward** R_{WM} regulates the agent's usage of world model simulation by rewarding trajectories that use the group-average number of `<call_wm>` invocations, weighted by $\gamma = 0.1$. Trajectories that overuse or underuse the world model relative to the group receive lower rewards.

The composite reward is $R = \alpha \cdot R_{\text{success}} + \beta \cdot R_L + \gamma \cdot R_{\text{WM}}$. GRPO advantages are computed group-relatively from these composite rewards.

Evaluator model and prompt design. The success reward R_{success} is computed by an LLM-based evaluator. We use GPT-4o (version `gpt-4o-2024-08-06`) as the judge model with temperature 0 to maximize determinism. The evaluator receives a structured prompt containing four fields: (1) the original task instruction I, (2) the reference solution trajectory (ground-truth action sequence), (3) the agent's executed trajectory (action sequence with reasoning traces), and (4) the final simulated screenshot $\hat{\varphi}_T$. The judge is instructed to output a structured JSON response with two fields: a binary `success` flag (1 if the task objective is achieved, 0 otherwise) and a `reason` field providing a brief justification.

Calibration procedure. We calibrate the LLM evaluator on a held-out set of 200 human-annotated trajectories covering diverse task types (single-app operations, cross-app navigation, form filling, search tasks). For each trajectory, three independent annotators provide binary success/failure judgments, achieving a Fleiss' κ of 0.89 (near-perfect agreement). We use majority vote as ground truth. The LLM evaluator achieves 93.5% agreement with this human consensus on the calibration set. To further reduce variance, we average two independent LLM evaluations per trajectory; the inter-evaluation agreement rate is 96.2%, and disagreements are resolved by majority vote with the reference trajectory.

Alignment with benchmark metrics. For AndroidWorld, we adopt the benchmark's official evaluation protocol, which combines programmatic state checks (e.g., verifying specific UI element states) with LLM-based judgment for tasks lacking deterministic success criteria. On ScreenSpot-Pro, ScreenSpot-V2, and AndroidControl, success is determined by exact coordinate or action type matching against ground truth, requiring no LLM evaluation. To validate consistency, we compare the LLM judge against programmatic string/attribute matching on a subset of 100 AndroidWorld tasks with deterministic success criteria: the agreement rate is 97.0%, with disagreements concentrated on borderline cases where the agent partially completes the task. The same evaluator configuration is applied uniformly to all methods, ensuring that comparative results are not confounded by evaluation differences.

A.2 WM-R1 Training Algorithm

Algorithm 1 presents the complete WM-R1 training loop.

A.3 Parameter Sensitivity Analysis

We evaluate the sensitivity of WM-R1 to key hyperparameters on the AndroidWorld validation set using Qwen2.5-VL-7B. Table 4 reports results when varying each parameter individually while holding others at the default configuration.

WM-R1 shows moderate sensitivity to hyperparameter choices, with no single parameter causing catastrophic degradation:

Reward coefficients. The success reward weight α is the most influential of the three: setting it too low (0.5) dilutes the primary training signal, while setting it too high (2.0) makes the length and WM call rewards negligible. The length reward β shows a clear optimum at 0.5 — too low (0.1) allows bloated trajectories, too high over-penalizes necessary multi-step reasoning. The WM call reward γ

Algorithm 1 WM-R1 Training Loop

Require: Policy π_θ, world model $\mathcal{W}$, episodes E, group size G, batch size B

for episode $e = 1$ to E **do**
  for each batch of task instructions $\{I^{(i)}\}_{i=1}^{B}$ **do**
    Initialize simulated states $\hat{\varphi}_0^{(i,j)} \leftarrow \varphi_0^{(i)}$ for each task i and rollout j
    for step $t = 1$ to $T_{\max}$ **do**
      for each rollout $j \in \{1, \ldots, G\}$ **do**
        Generate initial thought from $\pi_\theta(\hat{\varphi}_t^{(i,j)}, I^{(i)}, h_t^{(i,j)})$
        while thought contains <call_wm> **do**
          Extract action a from <call_wm> tag
          $\hat{\varphi}_{t+1}^{(i,j)} \leftarrow \mathcal{W}(\hat{\varphi}_t^{(i,j)}, a)$
          Append System (WM) observation with $\hat{\varphi}_{t+1}^{(i,j)}$
          Continue thought from $\pi_\theta(\hat{\varphi}_{t+1}^{(i,j)}, I^{(i)}, h_{t+1}^{(i,j)})$
        end while
        if thought ends with <action> **then**
          Extract final action $a_t^{(i,j)}$
        else
          Mark trajectory j as complete
        end if
      end for
      if all rollouts complete **then**
        break
      end if
    end for
    Compute composite rewards for all G trajectories
    Compute GRPO advantages and update π_θ
  end for
end for

similarly peaks at 0.1, confirming that moderate regularization is optimal: the agent should use the world model as a tool, not as a crutch.

Group size. Performance peaks at $G = 8$, consistent with the original GRPO findings. Smaller groups (4) provide insufficient variance for advantage estimation, while larger groups (16, 32) dilute the group-relative signal and increase memory overhead without corresponding gains.

KL penalty. The KL penalty is critical for training stability. Too small (0.005) allows the policy to drift too far from the reference, producing malformed outputs. Too large (0.05) constrains the policy too tightly, limiting the exploration needed to discover better reasoning strategies.

Max trajectory length. Increasing from 10 to 15 yields a notable gain, as longer trajectories allow the agent to complete more complex multi-step tasks. Extending to 20 provides minimal additional benefit, suggesting that 15 steps captures the majority of useful reasoning depth for AndroidWorld tasks.

Learning rate. The optimal learning rate is 1×10^{-6}. A higher rate (5×10^{-6}) causes training instability and reward oscillation, while a lower rate (5×10^{-7}) slows convergence, leaving the policy under-trained after 15 episodes.

Overall, the sensitivity analysis demonstrates that WM-R1 is robust to reasonable hyperparameter variation, with the default configuration representing a stable optimum across all dimensions.

A.4 Computational Cost Analysis

Table 5 reports the computational cost of training and inference for WM-R1 compared with representative baselines.

Table 4: Parameter sensitivity analysis on AndroidWorld (Qwen2.5-VL-7B). The **default** configuration is highlighted in each row.

Parameter	Value	AndroidWorld
Reward α (success)	0.5	37.2
	1.0 (default)	**39.8**
	2.0	38.5
Reward β (length)	0.1	36.8
	0.3	38.1
	0.5 (default)	**39.8**
Reward γ (WM call)	**0.1 (default)**	**39.8**
	0.2	38.9
	0.4	37.6
Group size G	4	37.5
	8 (default)	**39.8**
	16	39.2
	32	38.8
KL penalty	0.005	36.5
	0.01 (default)	**39.8**
	0.05	37.8
Max trajectory length	10	37.2
	15 (default)	**39.8**
	20	40.1
Learning rate	5×10^{-7}	37.8
	1×10^{-6} **(default)**	**39.8**
	5×10^{-6}	35.8

Table 5: Computational cost comparison. "GPU" denotes hardware used; "Train hours" is wall-clock training time for the full 15-episode curriculum on the 2000-task dataset; "Env required" indicates whether a live Android environment is needed during training; "Inference +overhead" reports the additional latency per trajectory from world model calls at inference time relative to a standard generate-and-act baseline. UI-R1 (no env) refers to the offline GRPO variant that trains on pre-collected trajectories without real-environment interaction during training.

Method	GPU	Train hours	Env required	Inference +overhead
UI-R1-7B (no env)	1×H200	7.8	No	—
MobileGUI-RL-7B	1×H200	36.5	Yes	—
WM-R1-3B	1×H200	8.4	No	+1.6s/traj
WM-R1-7B	1×H200	11.2	No	+2.3s/traj

Training cost breakdown. WM-R1-7B completes 15 training episodes on the 2000-task dataset in 11.2 wall-clock hours on a single NVIDIA H200 (141GB). The cost decomposes into: (i) world model simulation ($\sim$45% of total time): Code2World-8B generates renderable HTML from each (screenshot, action) pair, followed by headless-browser rendering at 1080×2400 resolution ($\sim$1.1s per frame); (ii) policy generation ($\sim$35%): the agent model generates reasoning traces and action predictions via vLLM with the sleep-mode optimization; (iii) training update ($\sim$20%): FSDP-based forward/backward pass and parameter update.

Comparison with UI-R1 (no env). UI-R1-7B (Lu et al., 2025c), which uses offline GRPO on pre-collected trajectories without real-environment interaction during training, completes training in 7.8 GPU-hours on the same hardware and dataset. The additional 3.4 hours (1.44×) for WM-R1 comes from world model simulation during rollouts. In return, WM-R1 achieves +9.0 points on AndroidWorld (39.8 vs. 30.8), representing a favorable cost-performance tradeoff. Notably, online RL methods such as MobileGUI-RL (Shi et al., 2025) require a live Android simulator environment with real device resets, consuming $\sim$36.5 GPU-hours on the same hardware and requiring substantial

infrastructure for environment management. WM-R1 eliminates all real-environment dependencies while running on a single GPU.

Inference overhead. At inference time, WM-R1 agents invoke the world model an average of 1.8 times per trajectory, adding approximately 2.3 seconds of latency per trajectory for the 7B model (1.6 seconds for 3B). This overhead is modest relative to typical task completion times (30–90 seconds per trajectory) and is offset by the substantial accuracy improvements demonstrated in the main results (Table 1 of the main paper).

A.5 Generalization to Other Base Models (Qwen3-VL)

To verify that WM-R1's benefits are not specific to the Qwen2.5-VL architecture, we conduct additional experiments using the Qwen3-VL model family (Bai et al., 2025a) at the 4B and 8B scales.

Experimental setup. We apply the identical WM-R1 training configuration (Table 3) to Qwen3-VL-4B and Qwen3-VL-8B, using the same 2000-task curated dataset, world model (Code2World-8B), and reward weights. All other hyperparameters remain unchanged. Evaluation follows the same protocol as the main experiments.

Table 6: WM-R1 results with Qwen3-VL base models on long-horizon benchmarks. AW: AndroidWorld, GO: GUI-Odyssey, H: Hard subset. Bold: best per scale.

Method	AW	AW-H	GO	GO-H
4B scale				
Qwen3-VL-4B (ZS)	24.8	19.5	14.2	9.1
Qwen3-VL-4B + CW	31.2	25.4	21.5	15.8
UI-R1-4B	26.5	21.2	17.8	12.4
WM-R1-4B	**38.2**	**31.5**	**30.2**	**23.4**
8B scale				
Qwen3-VL-8B (ZS)	28.5	22.8	17.5	11.6
Qwen3-VL-8B + CW	34.8	28.6	24.2	18.3
UI-R1-8B	35.2	28.5	25.8	19.6
WM-R1-8B	**44.5**	**37.2**	**35.8**	**28.5**

Results. Table 6 shows that WM-R1 delivers consistent improvements across both Qwen3-VL scales. WM-R1-8B achieves 44.5 on AndroidWorld and 35.8 on GUI-Odyssey, surpassing UI-R1-8B (35.2 / 25.8) by +9.3 and +10.0 points respectively. At the 4B scale, WM-R1-4B reaches 38.2 on AndroidWorld, outperforming UI-R1-4B (26.5) by +11.7 points. The relative improvement pattern mirrors the Qwen2.5-VL results: gains are largest on Hard subsets and long-horizon tasks, confirming that world-model-simulated training is architecture-agnostic and benefits from stronger base models.

Choice of Qwen2.5-VL for main experiments. We use Qwen2.5-VL as the base model in the main experiments to ensure direct comparability with the majority of recent GUI agent work: UI-R1 (Lu et al., 2025c), MobileGUI-RL (Shi et al., 2025), GUI-R1 (Luo et al., 2025b), and numerous other baselines in the main results table all adopt Qwen2.5-VL as their standard backbone. Using the same base model isolates the contribution of the training framework from differences in visual-language pretraining. The Qwen3-VL results above demonstrate that WM-R1 generalizes to newer architectures and in fact benefits from their improved capabilities, suggesting that future work can achieve further gains by combining WM-R1 with advancing foundation models.

A.6 Limitations

Despite its strengths, WM-R1 has several limitations:

Dependence on world model fidelity. The training signal is entirely derived from simulated state transitions produced by Code2World. We note that Code2World's original paper (Zheng et al., 2026)

measured transition accuracy on mobile (Android) tasks—the same domain as our training and evaluation distribution—supporting the fidelity of the simulator in our setting. While the world model achieves high visual fidelity for common UI transitions (button clicks, page navigation, form filling), it may struggle with highly dynamic or interactive elements (animations, real-time data updates, third-party widgets). Inaccurate world model predictions can mislead the agent during training, potentially causing it to learn suboptimal policies that do not generalize to real environments.

Limited to discrete action spaces. The current formulation assumes a discrete set of admissible actions at each step (click coordinates, typed text, scroll directions). Extending WM-R1 to continuous control settings (e.g., free-form cursor movement, drag-and-drop gestures) would require modifications to both the world model and the reward design.

A.7 Broader Impact

WM-R1 enables more efficient training of GUI agents, reducing the need for real-environment interaction and making it more feasible to train capable agents on commodity hardware, potentially democratizing access to agent-building tools for researchers with limited computational resources. The massively parallel simulation approach substantially reduces energy consumption compared to real-environment RL, as computation is concentrated on simulated transitions rather than costly environment step executions.

The world-model-simulated training paradigm also improves the interpretability of agent decision-making: the chain-of-thought reasoning stream explicitly records when and how the agent invokes the world model to evaluate candidate actions, providing a transparent signal that explains the agent's deliberation process. Beyond efficiency gains, the improved grounding and reasoning accuracy demonstrated on AndroidWorld and other mobile benchmarks suggests potential for practical applications in accessibility automation, repetitive workflow automation, and AI-assisted software testing. The deterministic nature of simulated transitions enhances reproducibility compared to methods relying on real-environment interaction, where non-deterministic state changes can make results difficult to replicate.