LEARN HOW TO ACT FROM YOUR OWN INTERACTIONS: ON-POLICY SELF-DISTILLATION FOR GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

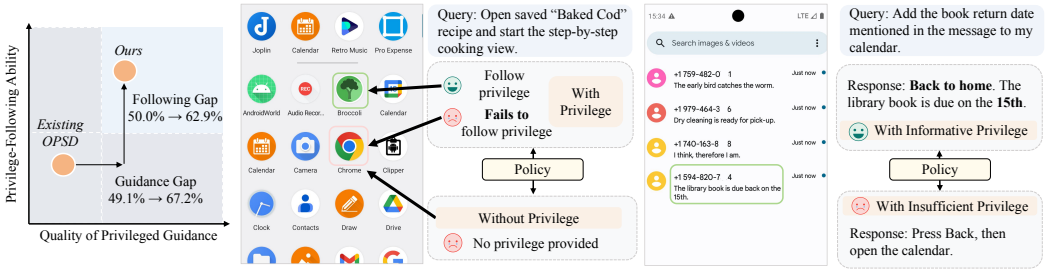
Graphical User Interface (GUI) agents enable the fulfillment of complex user instructions through multi-turn interactions with software environments, requiring step-wise reasoning and long-horizon memory to guide actions and retain task-relevant information, respectively. Recent on-policy self-distillation (OPSD) methods have achieved strong performance on GUI grounding, a foundational subtask for GUI agents, owing to dense token-level supervision from privilege-conditioned self-teachers. However, extending existing OPSD methods to multi-turn GUI agents is hindered by self-teachers' limited privilege-following ability and insufficient privileged guidance. In this paper, we introduce GUI-SD-v2, the next version of GUI-SD, which extends OPSD from GUI grounding to multi-turn GUI interaction and addresses key limitations through a two-stage training framework. Specifically, GUI-SD-v2 first strengthens privilege following by jointly optimizing rollouts with and without privileged guidance from the same GUI states. Furthermore, it selectively distills step-specific reasoning and memory guidance through a privilege-conditioned self-teacher, supporting action decisions and the retention of task-relevant information for subsequent interactions. Extensive experiments on two representative GUI agent benchmarks, AndroidWorld and MobileWorld, show that GUI-SD-v2 compares favorably with existing OPSD baselines while consistently outperforming the evaluated state-of-the-art methods in both Pass@1 and Pass@3 success rates. Code and training data will be publicly released.

1 INTRODUCTION

Autonomous GUI agents have emerged as a promising direction in human–computer interaction, enabling applications ranging from personal assistance to large-scale software testing. In parallel, on-policy self-distillation (OPSD) offers an effective paradigm for post-training such agents, combining dense token-level supervision, informative learning signals for hard cases, and efficient single-rollout training. Within the GUI domain, GUI-SD (Zhang et al., 2026b) presents the first exploration of OPSD for GUI grounding by conditioning the self-teacher on visually enriched privileged information and applying entropy-guided optimization to salient coordinate tokens. Subsequent work further refines GUI-SD in privileged information design and token-level optimization. Specifically, one line of work filters token-level distillation signals according to teacher confidence (Huang et al., 2026), while another introduces diagnostic reflections and next-step GUI screenshots as alternative forms of privileged information (Xuan & Li, 2026; Li et al., 2026a).

Despite recent advances in GUI grounding, existing OPSD methods remain limited in both privilege-following ability and the quality of privileged guidance, as illustrated in Figure 1(a), hindering their extension to long-horizon GUI agents: 1) **Privilege-Following Bottleneck.** Existing OPSD methods (Zhang et al., 2026b; Pei et al., 2026) suffer from limited privilege-following ability, which constrains the self-teacher's capacity to translate privileged guidance into reliable token-level supervision. As shown in Figure 1(b), the policy still incorrectly selects Chrome instead of the Broccoli recipe app even when privileged guidance is provided, indicating that the guidance does not reliably inform its action decisions. 2) Existing OPSD methods rely on coarse ground-truth cues or short self-generated reasoning traces (Li et al., 2026b; Yang et al., 2026), offering limited guidance on

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.



(a) Comparison with OPSD methods. *(b) Privilege-Following Bottleneck.* *(c) Insufficient Privileged Guidance.*

Figure 1: **Motivation for GUI-SD-v2.** (a) GUI-SD-v2 improves privilege following and privileged-guidance quality, with AndroidWorld success rates drawn from Table 2 and Table 3, respectively. (b) Limited privilege following prevents existing OPSD methods from reliably using privileged guidance for action decisions. In this example, Broccoli is a recipe-management app. (c) Coarse privileges offer insufficient reasoning and memory guidance for multi-step interactions.

action reasoning at the current step and retaining task-relevant information as interaction history for subsequent steps. As shown in Figure 1(c), insufficient guidance focuses on navigating to the calendar without retaining the book's due date, which is essential information across steps.

To address these issues, we introduce GUI-SD-v2 (GUI Agent via Self-Distillation), the next version of GUI-SD that extends OPSD from GUI grounding to long-horizon GUI agent tasks. GUI-SD-v2 introduces privilege-following optimization to strengthen the policy's ability to follow privileged guidance, and informative privilege distillation to provide richer supervision across multi-step interactions. Specifically, privilege-following optimization constructs corrective guidance from interaction failures to augment the policy's training context. The policy is then optimized using a shared group of rollouts generated from the same GUI states with and without the guidance, strengthening privilege following for subsequent self-distillation stage. Subsequently, informative privilege distillation conditions the self-teacher on reasoning and memory privileges that guide action decisions and identify task-relevant information to retain as history, transferring these signals to the corresponding policy outputs through selective token-level self-distillation.

Extensive experiments on MobileWorld (Kong et al., 2026) and AndroidWorld (Rawles et al., 2025) demonstrate that GUI-SD-v2 consistently outperforms existing OPSD methods and the evaluated state-of-the-art methods under both Pass@1 and Pass@3. Detailed ablations further show that privilege-following optimization jointly improves privilege following and general GUI capabilities, while informative privilege distillation provides complementary token-level supervision for action reasoning and task-relevant memory throughout multi-step interactions.

Our main contributions are summarized as follows:

- Extending OPSD to long-horizon GUI agents reveals two key bottlenecks in existing methods, namely limited privilege following and insufficient guidance for reasoning and interaction memory.
- We propose GUI-SD-v2, the next version of GUI-SD, which integrates privilege-following optimization with informative privilege distillation to strengthen privilege following and provide targeted reasoning and memory supervision across multi-step interactions.
- Extensive experiments on MobileWorld and AndroidWorld demonstrate that GUI-SD-v2 consistently outperforms existing OPSD methods and evaluated state-of-the-art methods under both Pass@1 and Pass@3.

2 Related Work

2.1 On-Policy Self-Distillation for GUI Agents.

OPSD (Li et al., 2026b; Lu et al., 2026b; Zhang et al., 2026a; Sang et al., 2026; Ye et al., 2026) has emerged as an effective post-training paradigm in which the same model serves as both student and teacher, with the teacher conditioned on privileged information to provide dense token-level super-

vision for student-generated on-policy responses. Within the GUI-agent domain, GUI-SD (Zhang et al., 2026b) pioneers the application of OPSD to GUI grounding by conditioning the self-teacher on a visually enriched privileged context and applying entropy-guided optimization to salient coordinate tokens. Recent GUI-oriented OPSD methods (Huang et al., 2026; Li et al., 2026a; Xuan & Li, 2026; Wu et al., 2026; Lian et al., 2026) mainly explore how to calibrate teacher signals and incorporate alternative forms of privileged information. Specifically, quality-aware self-distillation (Huang et al., 2026) combines correctness-aware gating with teacher-confidence scaling to filter unreliable coordinate-token signals and calibrates the remaining supervision. R-OPSD (Xuan & Li, 2026) derives diagnostic reflections from evaluated grounding attempts and conditions the self-teacher on these reflections to enable annotation-free test-time adaptation. However, current work remains centered on local decisions and lacks privileged contexts that capture multi-step planning rationales and task-relevant interaction history.

2.2 Long-Horizon GUI Agent Training.

Long-horizon GUI agents complete complex user tasks through extended interactions with dynamic interfaces, requiring multi-step reasoning, action planning, task-state tracking, and error recovery. To develop these capabilities, recent work (Yang et al., 2025; Gu et al., 2026; Shi et al., 2025; Lai et al., 2026a; Xu et al., 2025; Li et al., 2025; Lu et al., 2025b) has explored end-to-end reinforcement learning directly from agent-environment interactions. Motivated by the effectiveness of GRPO in reasoning tasks, ARPO pioneers GRPO-based outcome-reward training for long-horizon GUI agents, augmenting policy optimization with a replay buffer of successful trajectories (Lu et al., 2025a). WebAgent-R1 further develops outcome-reward training for web agents by integrating asynchronous online exploration with reasoning-oriented warm-up in an end-to-end multi-turn RL framework (Wei et al., 2025). Furthermore, subsequent work improves credit assignment by estimating task progress, evaluating intermediate decisions, and assigning rewards to verifiable milestones. For instance, ProgRM predicts task-completion progress at each interaction step and uses LCS-based self-annotation to construct dense progress rewards for online policy optimization (Zhang et al., 2025). Building on these advances, GUI-SD-v2 further extends credit assignment to the token level through dense OPSD supervision.

3 Method

We present GUI-SD-v2, the next version of GUI-SD that extends OPSD from GUI grounding to long-horizon GUI agent tasks. As shown in Figure 2, GUI-SD-v2 consists of two training stages. (a) **Privilege-Following Optimization** (Figure 2 Top) integrates privilege-context and original rollouts into a shared group for relative-advantage optimization, improving both privilege following and general GUI capabilities. (b) **Informative Privilege Distillation** (Figure 2 Bottom) conditions the self-teacher on reasoning and memory privileges to provide informative token-level supervision throughout multi-turn interactions.

3.1 Task Formulation

GUI-SD-v2 formulates long-horizon GUI interaction as a sequential decision process. At each interaction step t, the policy π_θ generates a structured response y_t conditioned on the user instruction q, the current GUI screenshot I_t, and the interaction history $h_{0:t-1}$ covering all preceding steps:

$$y_t \sim \pi_\theta(\cdot \mid q, I_t, h_{0:t-1}) .$$

As shown in the gray response block of Figure 2(c), the structured response y_t contains four fields: `thought`, `action`, `tool_call`, and `memory`. The `thought` field contains the current reasoning process, the `action` field describes the intended operation, and the `tool_call` field specifies the corresponding GUI action. The `memory` field records task progress, instruction-relevant information, and potential interaction errors at the current step, forming h_t for subsequent interactions. The GUI agent repeats this process until the user instruction is completed or the maximum interaction horizon is reached. Detailed interaction templates are provided in the Appendix D.

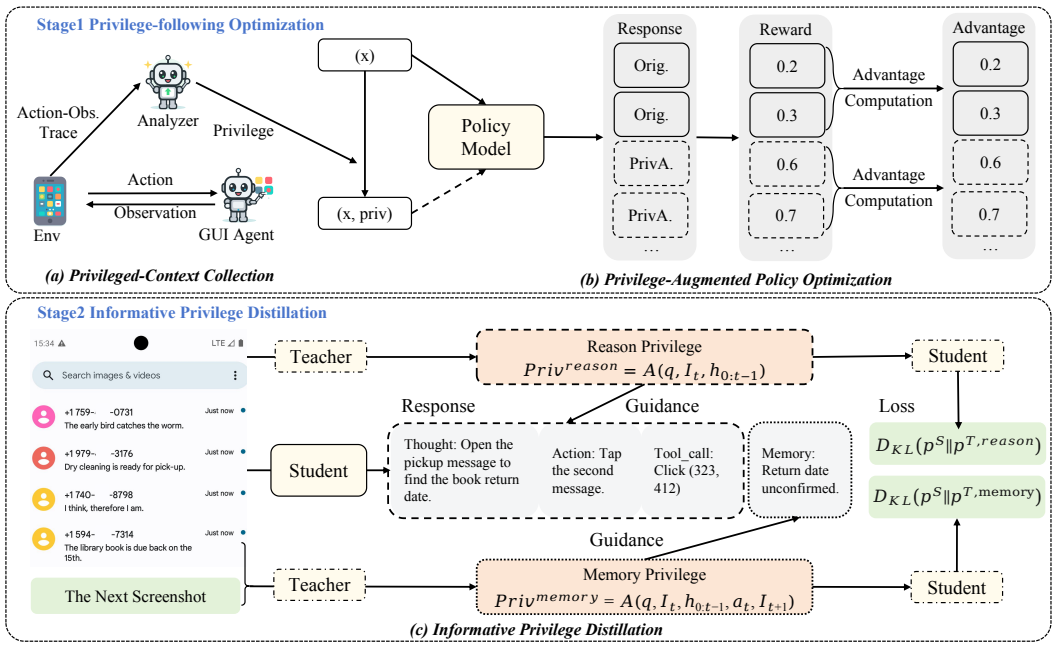


Figure 2: **Overview of GUI-SD-v2. Stage 1 Privilege-Following Optimization.** (a) collects corrective privileges from GUI trajectories and (b) jointly optimizes original-context and privilege-augmented rollouts, strengthening privilege following for subsequent distillation. **Stage 2 Informative Privilege Distillation.** (c) transfers reasoning and memory guidance to decision and memory outputs, respectively, through self-distillation.

3.2 Privilege-Following Optimization

To establish a reliable initialization for subsequent self-distillation, the first stage of GUI-SD-v2 (Figure 2 Top) equips the policy to follow the guidance provided by privileged contexts and generate the corresponding correct actions.

Privileged-Context Collection. As illustrated in Figure 2(a), the base policy generates GUI interaction trajectories for user instructions sampled from an offline pool with observations and actions recorded at each step. The analyzer model identifies an erroneous decision step in the trajectory and generates the corresponding correct action and a corrective experience annotation, which explains how the decision should be revised and serves as the privileged context. To validate these annotations, we apply rejection sampling to draw eight candidate actions from the privilege-conditioned GUI model and retain only samples for which at least one candidate matches the analyzer-provided correct action. Further details are provided in the Appendix B.

Privilege-Augmented Policy Optimization. As illustrated in Figure 2(b), given training samples with privileged annotations, we represent each sample as $(x, priv, a^*)$, where x comprises the user instruction, current GUI screenshot, and interaction history from all preceding steps, $priv$ denotes the privileged information, and a^* denotes the target action. For each sample, the policy constructs a shared group of N rollouts, comprising $N/2$ original-context rollouts conditioned on x and $N/2$ privilege-augmented rollouts conditioned on $(x, priv)$, all evaluated against a^*.

Following prior work (Lu et al., 2026a; Luo et al., 2025), GUI-SD-v2 assigns each rollout a verifiable reward based on response format and action correctness. The format component verifies the response structure defined in Section 3.1, while the accuracy component evaluates the predicted GUI action against a^* using action-specific criteria, including target-region matching for grounding, semantic matching for text input, and exact action-type matching for functional operations. The two components are combined to obtain the final rollout reward.

To jointly improve privileged-context following and general GUI capabilities, we compute advantages by normalizing rewards independently within the privilege-augmented ($c = \text{priv}$) and original-

context ($c = \text{orig}$) rollout subsets:

$$A_i^c = \frac{R_i^c - \mu_c}{\sigma_c}.$$

Here, i indexes a rollout, c identifies its subset, R_i^c is the corresponding reward, and μ_c and σ_c are the reward mean and standard deviation within that subset. These subset-specific advantages are then jointly used to optimize the policy under the standard clipped GRPO objective:

$$\mathcal{L}_{\text{PAPO}}(\theta) = -\frac{1}{N} \sum_{c \in \{\text{priv}, \text{orig}\}} \sum_{i=1}^{N/2} \mathbb{E}_t \left[\min\left(\rho_{i,t}^c A_i^c, \text{clip}\left(\rho_{i,t}^c, 1-\delta, 1+\delta \right) A_i^c \right) \right].$$

where $\rho_{i,t}^c$ denotes the importance ratio between the updated and rollout policies, and δ is the clipping threshold. The normalized advantage determines the contribution of each rollout, while ratio clipping constrains the magnitude of policy updates. Rollouts from both subsets jointly update the same policy, improving privileged-context following and general GUI capabilities.

3.3 Informative Privilege Distillation

As illustrated in Figure 2(c), the second stage provides token-level supervision at each interaction step for reasoning over the user instruction, and for summarizing the current step action as task-relevant interaction history. Specifically, the current policy generates on-policy GUI trajectories, and the teacher simultaneously provides informative privileges at every interaction step of each trajectory. The policy is then trained through self-distillation to internalize the teacher-provided privileges, with its privilege-conditioned predictions supervising those under the original context.

Informative Privilege Construction. For each user instruction, GUI-SD-v2 rolls out the current policy in the Android emulator environment to produce a multi-step GUI trajectory. Learning effectively from such long-horizon trajectories requires step-level supervision for two core capabilities: reasoning over the current interaction context to determine subsequent actions and summarizing each interaction step into task-relevant history for future decisions. Accordingly, the teacher generates complementary reasoning and memory privileges for every interaction step in the trajectory. At interaction step t, the teacher analyzes the current GUI screenshot and accumulated interaction history in light of the user instruction to infer the current task state and reason about the next step. Its resulting reasoning process serves as the reasoning privilege, providing step-specific guidance for subsequent reasoning and action selection. To construct the memory privilege, the teacher additionally observes the executed action and resulting next GUI state. By analyzing this state transition, it identifies task progress, instruction-relevant information, and salient state changes, with the resulting summary serving as the memory privilege.

Informative Privilege-Guided OPSD. Each privilege provides targeted guidance that is incorporated into the policy through self-distillation. Specifically, for each on-policy response, the policy under the original context serves as the student, while the same policy under the corresponding privilege-augmented context serves as the self-teacher. Let p_j^{S} denote the student distribution at token position j, and let $p_j^{\text{T,reason}}$ and $p_j^{\text{T,memory}}$ denote the self-teacher distributions conditioned on the reasoning and memory privileges, respectively.

The reasoning privilege provides supervision for the complete decision process, covering the `thought`, `action`, and `tool_call` fields. Denoting their token positions by $\mathcal{M}_{\text{reason}}$, the reasoning distillation loss is

$$\mathcal{L}_{\text{RKL}}^{\text{reason}} = \frac{1}{|\mathcal{M}_{\text{reason}}|} \sum_{j \in \mathcal{M}_{\text{reason}}} D_{\text{KL}} \left(p_j^{\text{S}} \,\middle\|\, p_j^{\text{T,reason}} \right).$$

The memory privilege instead supervises only the `memory` field. To isolate this guidance from the current action prediction, the student and self-teacher share the same response prefix through `tool_call`, while the memory privilege is provided only to the self-teacher. The student generates the memory field without observing the next-state screenshot. Denoting the subsequent memory-token positions by $\mathcal{M}_{\text{memory}}$, the memory distillation loss is

$$\mathcal{L}_{\text{RKL}}^{\text{memory}} = \frac{1}{|\mathcal{M}_{\text{memory}}|} \sum_{j \in \mathcal{M}_{\text{memory}}} D_{\text{KL}} \left(p_j^{\text{S}} \,\middle\|\, p_j^{\text{T,memory}} \right).$$

The two signals are combined as $\mathcal{L}_{\text{priv}} = \lambda_{\text{reason}}\mathcal{L}_{\text{RKL}}^{\text{reason}} + \lambda_{\text{memory}}\mathcal{L}_{\text{RKL}}^{\text{memory}}$. This selective distillation enables the policy to reason about the current action and retain task-relevant information for subsequent steps.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Implementation Details. We conduct all experiments using Qwen3-VL-Instruct-8B (Bai et al., 2025) as the base model and perform multi-turn rollouts in Android emulator environments (Rawles et al., 2025) on selected tasks from OpenMobile (Cheng et al., 2026) and MobileForge (Liu et al., 2026). Specifically, the first stage involves 400 tasks, with Kimi-K3 (Team et al., 2026) providing corrective guidance for erroneous decision steps. The second stage involves 1,013 tasks, with Kimi-K3 (Team et al., 2026) providing privileged guidance on action planning and task-relevant memory. Further training details are provided in Appendix B.

Baselines. To ensure a fair comparison, we keep the training tasks fixed across GUI-SD-v2 and all baselines and employ the same Kimi-K3 model to generate the supervision required by each method. Specifically, Kimi-K3 (Team et al., 2026) performs multi-turn rollouts on the selected tasks in the Android emulator, and its resulting GUI trajectories serve as ground-truth supervision. Consistent with prior work (Zhang et al., 2026b), we evaluate GUI-SD-v2 against four baselines: SFT, GRPO, Naive OPSD, and GUI-SD-v1. Both **SFT** and **GRPO** are trained with Kimi-K3 annotations as ground truth: SFT is supervised on complete teacher responses, including reasoning, actions, and task-relevant memory, whereas the action annotations serve as reward references for GRPO. **Naive OPSD** directly conditions the self-teacher on the ground-truth action as textual privileged information. **GUI-SD-v1** represents ground-truth click targets as visual privileges, including bounding boxes and Gaussian masks, while providing ground-truth targets for non-click actions as textual privileged information.

Evaluation Benchmarks. We evaluate GUI-SD-v2 on two representative mobile-agent benchmarks: AndroidWorld (Rawles et al., 2025) and MobileWorld (Kong et al., 2026). **AndroidWorld** provides a reproducible Android emulator environment with verifiable state-based evaluation. It comprises 116 tasks across 20 real-world applications, covering everyday activities such as note-taking, appointment scheduling, and messaging. **MobileWorld** is a more challenging dynamic benchmark comprising 201 tasks across 20 applications, with greater emphasis on long-horizon and cross-application tasks. Following prior work (Cheng et al., 2026; Li et al., 2026a; Liu et al., 2026), we report results on its GUI-only subset. To obtain more reliable performance estimates and mitigate the impact of environmental variability, we evaluate each task multiple times and report both pass@1 and pass@3.

4.2 MAIN RESULT

Comparisons with Baselines. Table 1 reports Pass@1 and Pass@3 results on AndroidWorld (Rawles et al., 2025) and MobileWorld (Kong et al., 2026), two representative benchmarks for GUI Agents. GUI-SD-v2 achieves the highest success rates across both benchmarks and evaluation settings compared with baselines that share the same training-data scale and Kimi-K3 supervision source. Specifically, the *SFT* baseline directly imitates Kimi-K3-generated responses, whereas the *GRPO* baseline assigns rollout rewards based on step-level action correctness, taking Kimi-K3's action annotations as ground truth. Both *SFT* and *GRPO* rely on hard supervision based on Kimi-K3's reference responses and actions, with annotation errors potentially propagating into imitation targets and rollout rewards. Additionally, both *Naive OPSD* and *GUI-SD-v1* (Zhang et al., 2026b) encode ground-truth actions as textual or visual privileges, offering limited guidance on action reasoning and interaction memory. GUI-SD-v2 combines step-specific reasoning and memory privileges with privilege-following optimization to provide richer supervision and strengthen the policy's ability to exploit it throughout multi-step interactions.

Comparisons with SOTA Methods. GUI-SD-v2 also compares favorably with existing GUI-agent methods on AndroidWorld (Rawles et al., 2025) and MobileWorld (Kong et al., 2026), two representative benchmarks. On AndroidWorld, GUI-SD-v2 achieves Pass@1 and Pass@3 success rates

Table 1: GUI Agent performance on two representative benchmarks, including AndroidWorld and MobileWorld. We report Pass@1 and Pass@3 success rates (%). **Bold** indicates the best results.

Method	AndroidWorld		MobileWorld	
	Pass@1	Pass@3	Pass@1	Pass@3
GUI-Owl-7B (Xu et al., 2026)	66.4	–	–	–
DART-GUI-7B (Lu et al., 2026c)	58.0	–	–	–
UI-Venus-7B (Gu et al., 2025)	49.1	–	8.5	–
UI-MOPD-8B (Lian et al., 2026)	–	–	12.0	–
MobileForge-8B (Liu et al., 2026)	50.9	67.2	10.3	–
OpenMobile-8B (Cheng et al., 2026)	64.7	78.0	17.7	24.8
GHD-8B (Li et al., 2026a)	66.5	73.3	–	–
Qwen3-VL-Instruct-8B (Bai et al., 2025)	40.5	55.2	17.9	23.9
+ *SFT*	50.9	56.9	19.7	26.5
+ *GRPO*	52.6	55.2	14.5	21.4
+ *Naive OPSD*	50.0	53.4	13.7	19.7
+ *GUI-SD-v1*	57.8	62.9	18.8	25.6
+ *Ours*	**67.2** (+26.7)	**79.3** (+24.1)	**25.6** (+7.7)	**31.6** (+7.7)

Table 2: Comparison of training strategies for privilege-following optimization on AndroidWorld. PAPO denotes our proposed privilege-augmented policy optimization. Evaluation covers GUI capability after Stage 2, privilege-following accuracy after Stage 1, and GUI capability after Stage 1.

Method	GUI Capability (Stage 2)	Privilege Following (Stage 1)	GUI Capability (Stage 1)
Baseline	55.2	50.0	40.5
SFT	61.2	55.2	44.8
GRPO	60.3	57.8	44.8
PAPO	**67.2**	**62.9**	**49.1**

of 67.2% and 79.3%, respectively, outperforming GHD (Li et al., 2026a), which conditions the self-teacher on the next screenshot and applies distillation when the self-teacher corrects an erroneous student prediction. On MobileWorld, GUI-SD-v2 achieves 25.6% Pass@1 and 31.6% Pass@3, surpassing OpenMobile (Cheng et al., 2026) under both evaluation settings. Notably, GUI-SD-v2 obtains these gains by transferring Kimi-K3-generated reasoning and memory guidance into the GUI policy through OPSD-based training. This design provides targeted feedback on the policy's own interaction trajectories, supporting more reliable execution of long-horizon GUI tasks.

4.3 Ablation Studies

Effectiveness of Privilege-Following Optimization. Privilege-Following Optimization, the first stage of GUI-SD-v2, is designed to jointly improve the policy's privilege-following ability and general GUI capabilities. Table 2 reports final AndroidWorld success rate after Stage 2 to assess whether this initialization benefits subsequent distillation, alongside task success rates after Stage 1 measured with and without privileged information provided at each rollout step to assess privilege following and general GUI capabilities, respectively.

Specifically, in row 1, the policy is trained directly in Stage 2 without privilege-following optimization, yielding the weakest results across all three evaluations and suggesting that limited privilege following constrains the effectiveness of informative privilege distillation. Row 2 adopts privilege-conditioned *SFT* with annotated CoT responses as supervised targets, achieving 61.2% AndroidWorld success after Stage 2. Row 3 instead optimizes privilege-conditioned rollouts through *GRPO*, assigning rewards based on action correctness and achieving 60.3% after Stage 2. Row 4 applies

Table 3: Ablation of the memory and reasoning guidance components in informative privilege distillation. In our evaluation, the 116 AndroidWorld tasks are grouped into information retrieval (27 tasks) and task completion (89 tasks), with the former placing greater demands on retaining key information across interactions.

Memory	Reason	AndroidWorld	Information Retrieval	Task Completion
✗	✗	49.1	44.4 (12/27)	50.6 (45/89)
✗	✓	61.2	51.9 (14/27)	64.0 (57/89)
✓	✗	56.0	55.6 (15/27)	56.2 (50/89)
✓	✓	**67.2**	**59.3** (16/27)	**69.7** (62/89)

Table 4: Effect of privileged-guidance granularity on AndroidWorld. We report Pass@1 and Pass@3 success rates (%).

Granularity	AndroidWorld	
	Pass@1	**Pass@3**
Trajectory-level	51.7	60.3
Step-level	**67.2**	**79.3**

our proposed PAPO, jointly optimizing privilege-augmented and original-context rollouts within a shared group while normalizing their advantages separately. This design achieves 67.2% final AndroidWorld success after Stage 2, together with 62.9% privilege-following accuracy and 49.1% general GUI performance after Stage 1, delivering the best results across all three evaluations. We attribute these gains to PAPO, which prepares the shared policy for both roles in Stage 2 by improving the self-teacher's use of privileged guidance and strengthening the student's general GUI capabilities.

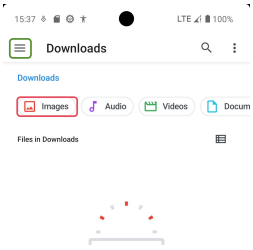
Effectiveness of Informative Privilege Distillation. Table 3 evaluates informative privilege distillation by ablating its reasoning and memory guidance, reporting overall AndroidWorld success alongside performance on information-retrieval tasks requiring the retention of key information across interactions and task-completion tasks demanding multi-step reasoning. The first row reports a baseline without informative privilege distillation, in which neither reasoning nor memory guidance is provided, yielding the lowest success rates across all three evaluations. Additionally, introducing privileged reasoning guidance improves information-retrieval and task-completion success rates from 44.4% to 51.9% and from 50.6% to 64.0%, respectively, suggesting that better GUI-state interpretation and action planning benefit both task types. Incorporating privileged memory guidance increases information-retrieval success from 44.4% to 55.6%, suggesting that explicit memory supervision helps the policy remember task-relevant information across interactions. Combining both privileges yields the best results across all three evaluations, supporting the complementary roles of reasoning and memory guidance in long-horizon GUI interactions.

Effect of Supervision Granularity. To investigate the effect of privileged-guidance granularity, we compare trajectory-level and step-level variants, both incorporating reasoning and memory guidance. The trajectory-level variant provides guidance for the entire trajectory and applies the same guidance at every interaction step, whereas our step-level variant provides guidance tailored to each step. As shown in Table 4, step-level supervision improves AndroidWorld Pass@1 from 51.7% to 67.2%, suggesting that matching reasoning and memory guidance to individual steps provides more effective supervision for long-horizon interactions.

Effect of Privilege Guidance from Different Teachers. We compare four external models as sources of reasoning and memory privileges for GUI-SD-v2. As shown in Table 5, guidance from Qwen3.5-Plus (Qwen Team, 2026), MiniMax-M3 (Lai et al., 2026b), and Gemini3.8-Flash (Team et al., 2025) yields AndroidWorld success rates of 57.8%, 59.5%, and 63.7%, respectively. GUI-SD-v2 performs best with Kimi-K3-generated guidance, reaching 67.2%. These results highlight the influence of the privilege source on self-distillation effectiveness and support our choice of Kimi-K3 for privilege generation.

Table 5: Effect of privilege guidance from different teachers on AndroidWorld.

Teacher	AndroidWorld
Qwen3.5-Plus (Qwen Team, 2026)	57.8
MiniMax-M3 (Lai et al., 2026b)	59.5
Gemini3.8-Flash (Team et al., 2025)	63.7
Kimi-K3 (Team et al., 2026)	**67.2**

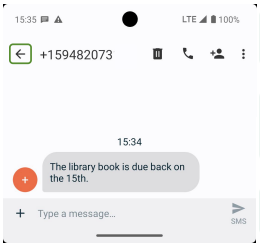


Query: In the Recent View of the files app, switch the filter to 'Images' and verify if any files are displayed.

Teacher–Student Token Probability on Student Response: Divergence at Decision Tokens

Response	…	I	need	to	switch	the	filter	to	Images	…
Teacher p	…	0.62	0.58	0.64	0.43	0.61	0.56	0.50	0.41	…
Student p	…	0.60	0.50	0.50	0.68	0.50	0.64	0.45	0.70	…
Gap Δp	…	0.02	0.08	0.14	−0.25	0.11	−0.08	0.05	−0.29	…

(a) Token Probabilities at Decision Tokens



Query: Add a calendar event to remind me to return the library book on the date mentioned in this message.

Response (with Memory Guidance): Thought: …, ToolCall: …, Memory: Tap the back arrow in the top-left corner, then open Calendar. Create a "Return library book" event for the 15th and confirm the month and year before saving.

Response (without Memory Guidance): Thought:…, ToolCall: …, Memory: Tap the back arrow in the top-left corner.

(b) Response Differences With and Without Memory Guidance

Figure 3: Qualitative analysis of informative privilege distillation. (a) The self-teacher assigns lower probabilities to decision-critical tokens in an erroneous student response, providing a corrective distillation signal. (b) Memory guidance encourages the policy to retain the task-relevant event details, whereas the unguided response retains only the navigation operation.

4.4 Qualitative Analysis

Token Probabilities at Decision Tokens. As shown in Figure 3(a), the self-teacher provides informative probability signals when its response is incorrect. When scoring the student's response, the teacher assigns lower probabilities to decision-critical tokens such as "switch" (0.43 versus 0.68) and "Images" (0.41 versus 0.70). These differences provide a distillation signal that discourages the erroneous action intent, illustrating the value of soft supervision beyond the correctness of the teacher's complete response.

Response Differences With and Without Memory Guidance. As shown in Figure 3(b), memory guidance encourages the policy to retain the book return date required by the instruction, whereas the response without memory guidance retains only the navigation operation. This comparison illustrates how memory supervision helps retain task-relevant information from the current step for subsequent decisions and task completion.

5 Conclusion and Limitations

We present GUI-SD-v2, a two-stage framework that extends on-policy self-distillation from GUI grounding to long-horizon GUI agents. By strengthening privilege following and selectively distilling step-specific reasoning and memory guidance, GUI-SD-v2 provides targeted token-level supervision across multi-turn interactions. Experiments on AndroidWorld and MobileWorld demonstrate consistent improvements over existing OPSD baselines and the evaluated state-of-the-art methods in both Pass@1 and Pass@3. While our evaluation focuses on Android environments, exploring additional GUI platforms and a broader range of model scales is a promising direction for future work.

ETHICS STATEMENT

This work studies GUI-agent training using existing datasets and Android emulator environments. The main ethical considerations are shared with GUI-agent research more broadly, including user privacy, authorized access, and responsible automation. Deployment and reuse of the resulting models and data should respect applicable licenses and privacy requirements, operate within user-granted permissions, and maintain human oversight for consequential actions.

AI USE STATEMENT

We use Kimi-K3 to generate training supervision, including reference trajectories and responses, corrective action annotations, and step-specific reasoning and memory guidance, as described in the paper. Other generative models are compared as alternative sources of privileged guidance. AI tools also assist with manuscript drafting and editing. The authors take full responsibility for the research methodology, experimental results, and final manuscript.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kanzhi Cheng, Zehao Li, Zheng Ma, Nuo Chen, Jialin Cao, Qiushi Sun, Zichen Ding, Fangzhi Xu, Hang Yan, Jiajun Chen, et al. Openmobile: Building open mobile agents with task and trajectory synthesis. *arXiv preprint arXiv:2604.15093*, 2026.

Jihao Gu, Qihang Ai, Yingyao Wang, Pi Bu, Jingxuan Xing, Yue Cao, Zekun Zhu, Wei Jiang, Ziming Wang, Yingxiu Zhao, et al. Mobile-r1: Towards interactive capability for vlm-based mobile agent via systematic training. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30802–30820, 2026.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. Ui-venus technical report: Building high-performance ui agents with rft. *arXiv preprint arXiv:2508.10833*, 2025.

Jingyuan Huang, Zuming Huang, Yucheng Shi, Tianze Yang, Xiaoming Zhai, Wei Chu, and Ninghao Liu. Trust the right teacher: Quality-aware self-distillation for gui grounding. *arXiv preprint arXiv:2606.18101*, 2026.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6142–6167, 2026.

Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. Computerrl: Scaling end-to-end online reinforcement learning for computer use agents. In *International Conference on Learning Representations*, 2026a.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, et al. Minimax sparse attention. *arXiv preprint arXiv:2606.13392*, 2026b.

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*, 2025.

Weiwei Li, Junzhuo Liu, Tong Chu, Hengfu Yu, and Wen Li. The next screenshot knows: Gated hindsight distillation for mobile gui agents. *arXiv preprint arXiv:2608.06065*, 2026a.

Yijiang Li, Bingyang Wang, Yijun Liang, Yunjie Tian, Di Fu, and Nuno Vasconcelos. On-policy self-distillation without any supervision. *arXiv preprint arXiv:2608.06296*, 2026b.

Niu Lian, Tongbo Chen, Zhehao Yu, Chengzhen Duan, Fazhan Liu, Hui Liu, Pei Fu, Jian Luan, Heng Qu, Shu-Tao Xia, and Jinpeng Wang. Ui-mopd: Multi-platform on-policy distillation for unified gui agents, 2026. URL https://arxiv.org/abs/2607.04425.

Guangyi Liu, Pengxiang Zhao, Gao Wu, Yiwen Yin, Mading Li, Liang Liu, Congxiao Liu, Zhang Qi, Mengyan Wang, Liang Guo, et al. Mobileforge: Annotation-free adaptation for mobile gui agents with hierarchical feedback-guided policy optimization. *arXiv preprint arXiv:2606.19930*, 2026.

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.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, et al. Ui-s1: Advancing gui automation via semi-online reinforcement learning. *arXiv preprint arXiv:2509.11543*, 2025b.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, et al. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17608–17616, 2026a.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026b.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, et al. Experience-driven multi-turn reinforcement learning for gui agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9474–9496, 2026c.

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*, 2025.

Rongcan Pei, Zhepei Wei, Shuyao Xu, Xinyu Zhu, Wei-Lin Chen, and Yu Meng. Negative self-distillation: Learning to reason by avoiding flaws. *arXiv preprint arXiv:2609.11699*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2025. URL https://arxiv.org/abs/2405.14573.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. Crisp: Compressed reasoning via iterative self-policy distillation. *arXiv preprint arXiv:2603.05433*, 2026.

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.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, et al. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, et al. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

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. 7909–7928, 2025.

Yubin Wu, Zicheng Cai, Liping Ning, Hua Wang, Zhi Chen, Yaohua Tang, and Hao Chen. Litegui: Distilling compact gui agents with reinforcement learning. *arXiv preprint arXiv:2605.07505*, 2026.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-agent-v3.5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026.

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, 2025. URL https://arxiv.org/abs/2509.18119.

Shiyu Xuan and Zechao Li. Test-time self-evolving gui visual grounding via reflection-guided on-policy self-distillation. *arXiv preprint arXiv:2608.11191*, 2026.

Chenyu Yang, Shiqian Su, Shi Liu, Xuan Dong, Yue Yu, Weijie Su, Xuehui Wang, Zhaoyang Liu, Jinguo Zhu, Hao Li, et al. Zerogui: Automating online gui learning at zero human cost. *arXiv preprint arXiv:2505.23762*, 2025.

Yongkang Yang, Zhezheng Hao, Hong Zhang, Yi Liu, Xiankun Lin, Wence Ji, Fanjunduo Wei, Jiarui Yu, Qiang Lin, Xiaoyun Liang, et al. Matching supervision to the student's learning capacity: A unified framework for on-policy self-distillation. *arXiv preprint arXiv:2608.08176*, 2026.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Danyang Zhang, Situo Zhang, Ziyue Yang, Zichen Zhu, Zihan Zhao, Ruisheng Cao, Lu Chen, and Kai Yu. Progrm: Build better gui agents with progress rewards. *arXiv preprint arXiv:2505.18121*, 2025.

Xinsen Zhang, Zhenkai Ding, Tianjun Pan, Run Yang, Chun Kang, Xue Xiong, and Jingnan Gu. Opsdl: On-policy self-distillation for long-context language models. *arXiv preprint arXiv:2604.17535*, 2026a.

Yan Zhang, Daiqing Wu, Huawen Shen, Can Ma, and Yu Zhou. Learn where to click from yourself: On-policy self-distillation for gui grounding. *arXiv preprint arXiv:2605.00642*, 2026b.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Hong Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift:a scalable lightweight infrastructure for fine-tuning, 2025. URL https://arxiv.org/abs/2408.05517.

APPENDIX

The appendix includes the following aspects:

- Appendix A: Evaluation Benchmarks.
- Appendix B: Training Details.
- Appendix C: Evaluation Setting.
- Appendix D: GUI-SD-v2 Template.

A EVALUATION BENCHMARKS

AndroidWorld. AndroidWorld (Rawles et al., 2025) is an interactive benchmark for evaluating autonomous agents in a fully functional Android environment. It comprises 116 parameterized task templates across 20 applications, covering everyday activities such as messaging, calendar management, note-taking, file manipulation, and system configuration. The benchmark includes tasks that modify application or device states and information-retrieval tasks that require answering questions about existing data. Each task is dynamically instantiated with randomized parameters and dedicated initialization, validation, and teardown procedures, supporting reproducible evaluation under varying task conditions. Task outcomes are assessed programmatically by inspecting application databases, files, and system settings, or by checking the agent's answer against the expected response. This outcome-based evaluation accommodates different valid interaction sequences rather than requiring agents to reproduce a predefined action trajectory.

MobileWorld. MobileWorld (Kong et al., 2026) comprises 201 tasks across 20 applications, emphasizing long-horizon workflows and cross-application interactions in domains such as communication, e-commerce, productivity, and information retrieval. Across the complete benchmark, 62.2% of tasks involve multiple applications, requiring agents to coordinate actions and maintain relevant information across application boundaries. The environment supports reproducible evaluation through snapshot-based initialization and self-hosted application backends. Task success is determined through textual answer verification, backend database checks, local storage inspection, or application-specific callbacks. In this work, we evaluate on the GUI-only subset, which includes both task-completion and information-retrieval tasks, focusing on multi-step GUI interaction without requiring user clarification or MCP tool use.

B TRAINING DETAILS

B.1 HYPERPARAMETERS

We implement training using the ms-swift framework (Zhao et al., 2025) with Qwen3-VL-Instruct-8B (Bai et al., 2025) as the base model. All model parameters, including the vision encoder and vision-language connector, are optimized using AdamW. Table 6 summarizes the hyperparameters for both stages.

B.2 REWARD SPECIFICATION

Stage 1 evaluates generated responses using a format reward and an action-specific accuracy reward.

Format Reward. The format reward R_{format} checks whether the response follows the required structure containing `thought`, `action`, `tool_call`, and `memory`.

Table 6: Training hyperparameters for the two stages of GUI-SD-v2.

Hyperparameter	Value
Stage 1: Privilege-Following Optimization	
Learning rate	5×10^{-6}
Learning-rate schedule	Cosine decay to zero
Warmup ratio	0.05
Training epochs	1
Per-device micro-batch size	2
Gradient accumulation steps	16
Rollouts per GUI state	8
Rollouts with / without privilege	4 / 4
Sampling temperature	1.0
Top-p	1.0
Max. response length (tokens)	16,384
Stage 2: Informative Privilege Distillation	
Initialization	Stage 1 checkpoint
Learning rate	1×10^{-6}
Learning-rate schedule	Cosine decay to zero
Warmup ratio	0.1
Training epochs	2
Gradient accumulation steps	16
On-policy trajectories per task	1
Sampling temperature	1.0
Top-p	1.0
Max. response length (tokens)	16,384
Max. trajectory length (steps)	30

Accuracy Reward. The accuracy reward R_{accuracy} is selected according to the action category, using R_{click} for click actions, R_{text} for text-input actions, and R_{func} for functional actions:

$$R_{\text{click}} = \begin{cases} 1, & \|o_{\text{point}} - p_{\text{gt}}\|_2 \le 50, \\ 0, & \text{otherwise}, \end{cases}$$

$$R_{\text{text}} = \begin{cases} 1, & \text{F1}(o_{\text{text}}, T_{\text{gt}}) > 0.5, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

$$R_{\text{func}} = \begin{cases} 1, & o_{\text{act}} = A_{\text{gt}}, \\ 0, & \text{otherwise}. \end{cases}$$

Here, o_{point} and p_{gt} denote the predicted and reference click coordinates, with Euclidean distance measured in pixels. $\text{F1}(\cdot)$ measures the agreement between the predicted text o_{text} and reference text T_{gt}. For functional actions, o_{act} and A_{gt} denote the predicted and reference action types.

Response Reward. The final reward combines format compliance and action accuracy:

$$R = 0.2R_{\text{format}} + 0.8R_{\text{accuracy}}. \tag{2}$$

B.3 Privileged-Context Collection

GUI-SD-v2 collects trajectories by rolling out the base policy on 400 tasks from MobileForge (Liu et al., 2026), recording the observations and actions at each interaction step. Kimi-K3 (Team et al., 2026) analyzes these trajectories to identify erroneous decisions and generate the corresponding reference actions and corrective experience annotations. Each annotation explains how the erroneous decision should be revised and serves as the privileged context for that step.

GUI-SD-v2 validates the annotations through rejection sampling before Stage 1 training. For each annotated step, the GUI policy receives the original context augmented with the corrective annotation and generates eight candidate responses. The sample is retained only if at least one candidate predicts an action matching the reference. The retained samples provide the GUI states, privileged contexts, and reference actions used for privilege-following optimization.

C Evaluation Settings

Evaluation Metrics. GUI-SD-v2 is evaluated on each benchmark over three runs, with every task evaluated once per run and task seeds held fixed across runs. Pass@1 is the mean success rate across the three complete benchmark runs, while Pass@3 is the percentage of tasks successfully completed in at least one of the three attempts. Task success is determined by each benchmark's evaluator. Attempts affected by environment initialization failures are excluded from scoring and rerun, ensuring three valid attempts per task.

Baseline Comparison. All baselines use training data from both stages of GUI-SD-v2, sharing the same training-task pools and supervision source, Kimi-K3 (Team et al., 2026). The supervision format is adapted to each method: SFT learns from teacher-generated responses, while GRPO uses teacher-provided actions to compute step-level rewards. Naive OPSD and GUI-SD-v1 use reference actions as textual or visual privileges for self-distillation.

D GUI-SD-v2 Template

The tool-definition template in Figure 4 specifies the available GUI actions, their required arguments, and the coordinate conventions for spatial operations. The response template in Figure 5 organizes each response into four fields: thought explains the next operation, action describes the intended action, tool call specifies the function invocation and its arguments, and memory captures task-relevant information to retain across interactions.

Mobile Interaction Prompt: Tool Definition

You are a mobile GUI assistant. Use the user instruction, current screenshot, and interaction history to select the next operation and retain information needed later.

 # Tools

Coordinates are normalized to 0–999 on each axis, with the origin at the top-left corner. Target the center of the intended element and allow the interface to update before choosing the next operation.

```
<tools>
{
  "type": "function",
  "function": {
    "name": "mobile_use",
    "description": "Operate a mobile touchscreen and receive
  screenshots.",
    "parameters": {
      "type": "object",
      "properties": {
        "action": {
          "type": "string",
          "enum": ["click", "long_press", "swipe", "type",
                   "answer", "system_button", "wait", "terminate"]
        },
        "coordinate": {
          "type": "array",
          "description": "[x,y]: target for click/long_press; start
  for swipe."
        },
        "coordinate2": {
          "type": "array",
          "description": "[x,y]: ending point for swipe."
        },
        "text": {
          "type": "string",
          "description": "Input content for type; response content for
   answer."
        },
        "time": {
          "type": "number",
          "description": "Duration in seconds for long_press or wait."
        },
        "button": {
          "type": "string",
          "enum": ["Back", "Home", "Menu", "Enter"],
          "description": "System button selected by system_button."
        },
        "status": {
          "type": "string",
          "enum": ["success", "failure"],
          "description": "Task outcome reported by terminate."
        }
      },
      "required": ["action"]
    }
  }
}
</tools>
```

Include the arguments associated with the selected action. For `swipe`, provide both coordinate fields. For `system_button`, use `Back` for the previous screen, `Home` for the home screen, `Menu` for recent apps, or `Enter` for the Enter key.

Figure 4: Tool definition for GUI-SD-v2.

Mobile Interaction Prompt: Response Format

 # Response Format
Return exactly four parts in the following order.

1. Thought. Within `<thought>...</thought>`, briefly explain the next operation based on the current observation, user instruction, and interaction history.

2. Action. Within `<action>...</action>`, describe the intended operation in one short sentence.

3. Tool call. Within `<tool_call>...</tool_call>`, provide exactly one valid JSON object. Set `name` to `"mobile_use"` and include the selected action and its associated parameters in `arguments`. The call must implement the operation described in `action`.

4. Memory. Within `<memory>...</memory>`, retain task-relevant facts from the current screenshot and interaction history, preserving relevant names, dates, numbers, and other details needed later. Summarize confirmed progress and observed errors. Record the intended action and any outcomes that require verification in subsequent steps. Update unresolved outcomes only when supported by new observations.

 # Rules
Use `answer` with `text` to deliver requested information. Use `terminate` with `status` to end the task. Output no content outside the four required parts.

 # Example
The following illustrates a waiting step when the current screen displays a loading indicator.

```
<thought>
The loading indicator suggests that the interface needs more time.
</thought>
<action>
Wait one second for the interface to update.
</action>
<tool_call>
{"name": "mobile_use", "arguments": {"action": "wait", "time": 1}}
</tool_call>
<memory>
The current screen shows a loading indicator; completion is unverified
  .
</memory>
```

Figure 5: Response instructions specifying a structured output with thought, action, tool call, and memory fields.