ASYPO: ASYMMETRIC POST-TRAINING FOR GUI AGENTS WITH SEGMENT-LEVEL REASONING GUIDANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Graphical user interface (GUI) agents require both executable actions and reasoning, but existing outcome-based post-training provides only coarse supervision for the latter. Existing post-training methods mainly include reinforcement learning (RL) and on-policy self-distillation (OPSD). Although OPSD enables process-level supervision through teacher–student token probabilities, applying it independently to long reasoning spans can distort their global structure. We propose an asymmetric post-training framework that combines action-level Group Relative Policy Optimization with relative segment-level reasoning guidance. Our method aggregates teacher guidance within reasoning segments, centers the resulting signals across each reasoning span, and selectively applies them through negative-advantage gating. This provides targeted reasoning supervision while preserving the overall structure of model-generated reasoning. Experiments on OSWorld and OmniACT demonstrate consistent improvements over the evaluated baselines at both 4B and 8B model scales.

1 INTRODUCTION

Graphical user interface agents follow natural-language instructions to operate computers, mobile devices and web applications. They must interpret screens, locate targets, plan multiple steps and execute actions in changing environments (Xie et al., 2024; Lai et al., 2025; Zhang et al., 2026b). Their responses therefore combine reasoning text with executable actions. Although both components support task completion, action outcomes alone cannot precisely evaluate the reasoning that led to them.

Reinforcement learning improves GUI agents through environment feedback. Group Relative Policy Optimization (GRPO) updates sampled responses using relative rewards (Lu et al., 2026b; Li et al., 2026; Wang et al., 2026a; Zeng et al., 2026). Assigning rewards to individual interaction steps can reduce the sparsity of final-outcome feedback, but it still does not identify which parts of a response contributed to the decision. In each step, reasoning and executable action tokens share the same advantage under standard GRPO (Zhang et al., 2026a). Rewards based on action correctness or execution progress therefore provide coarse feedback for the reasoning text and cannot distinguish useful reasoning segments from irrelevant ones.

On-policy self-distillation, or OPSD, provides finer guidance by using a teacher to evaluate responses sampled from the current policy (Zhao et al., 2026). However, because the teacher receives privileged information in the form of the gold action, its token preferences may differ from the reasoning that remains useful when the model must act without access to that action. Directly transferring these preferences may cause the model to focus on local wording rather than the reasoning process needed during execution. This motivates a closer examination of how teacher preferences are distributed across different parts of a reasoning span.

Figure 1 evaluates whether existing training signals can identify useful reasoning. In Panels A and B, we remove one sentence from a reasoning span and measure the resulting change in the probability of the corresponding action. The reward generally separates correct from incorrect responses: correct responses receive higher rewards than incorrect ones. However, after grouping responses by action correctness, the reward changes little with the measured importance of the removed reasoning

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.

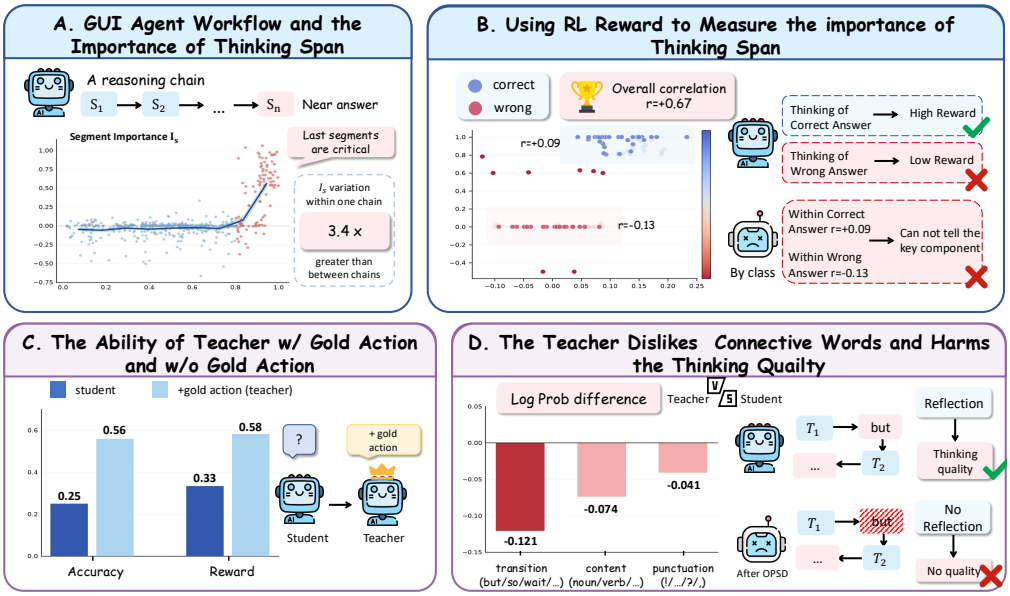


Figure 1: We conducted four diagnostic experiments to identify the limitations of existing training signals and motivate the design of our method.

sentence. Thus, GRPO can distinguish successful from unsuccessful trajectories, but cannot reliably identify which parts of the reasoning are useful within either group. Panels C and D examine the corresponding OPSD signal. Providing the teacher with the gold action improves its accuracy and reward, but also shifts its token preferences: as shown in Panel D, the teacher assigns lower probabilities to connective tokens and favours more direct answers. Because reducing connective words can shorten the reasoning trace and harm reasoning quality (Kaur et al., 2026), the privileged teacher signal is not necessarily suitable for direct token-level optimization of reasoning.

Based on these observations, we propose an ***Asymmetric Post-training*** framework, AsyPO for GUI agents. We aggregate privileged-teacher guidance within contiguous reasoning segments and center the resulting scores across each reasoning span to obtain relative segment weights. Through negative-advantage gating, this guidance is applied only to trajectories whose rewards fall below the group mean. For executable actions, we retain action-level GRPO with a GUI reward that captures action matching and functional progress, without directly distilling actions from the privileged teacher. The resulting framework combines relative segment-level guidance for reasoning with reward optimization for executable actions. Our main contributions are summarized as follows.

- We propose an asymmetric post-training framework for GUI agents that assigns distinct optimization objectives to reasoning spans and executable actions.
- We introduce reasoning-specific OPSD, which uses a teacher with privileged context to generate outcome-conditioned, segment-level shaping signals for on-policy reasoning trajectories without directly distilling the teacher's action distribution.
- We evaluate our framework on OSWorld and OmniACT with two model scales. The framework achieves higher reported overall task success on OSWorld and higher Sequence Scores and Action Scores on OmniACT than the corresponding evaluated baselines.

2 Related Work

GUI Agents Research on GUI agents has focused on benchmark construction, visual grounding and interactive training. SeeClick introduced ScreenSpot for locating interface elements from screenshots and pretrained a model with screenshot and instruction pairs (Cheng et al., 2024). Although this improves local grounding, it does not address long horizon planning. OSWorld evaluates multimodal agents in real operating systems across web applications, desktop software, file opera-

tions and multi-application workflows (Xie et al., 2024; Yuan et al., 2026). OmniACT evaluates agents that generate executable scripts from screenshots and natural language instructions across desktop applications and tasks of different lengths (Kapoor et al., 2024). Recent studies have explored interactive and staged training. DigiRL combines offline reinforcement learning with offline to online fine-tuning for Android control and uses parallel environments with a vision language evaluator (Bai et al., 2024). AndroidWorld evaluates generalization in dynamic Android environments with changing application states and multi application tasks (Rawles et al., 2025). Aguvis separates GUI grounding from planning and reasoning and introduces a multimodal dataset for these capabilities (Xu et al., 2025).

On-Policy Self-Distillation On-policy self-distillation provides token level guidance on model generated rollouts and can complement the outcome signals used by GRPO (Agarwal et al., 2024). Recent studies have explored this combination in reasoning and agentic tasks. SDAR uses privileged context to construct a teacher that guides the student on its own reasoning rollouts (Lu et al., 2026a). Purified OPSD separates useful teacher guidance from signals caused by privileged reference answers (Shen et al., 2026). SEED extracts hindsight skills from completed trajectories and distils them into the policy together with outcome based reinforcement learning (Wu et al., 2026). AgentOPSD aggregates probability differences between the teacher and student at each interaction turn and recursively estimates credit from sparse outcomes (Wang et al., 2026b).

Group Relative Policy Optimization GRPO estimates advantages from relative rewards among responses sampled for the same question, avoiding the separate value model required by PPO and reducing training costs (Shao et al., 2024). DeepSeek R1 applies GRPO with verifiable rewards to improve reasoning performance (DeepSeek-AI, 2025), while DAPO introduces several optimization strategies to improve training efficiency and stability (Yu et al., 2025). For interactive agents, GiGPO estimates relative advantages across complete trajectories and among actions sampled from the same environment state, providing finer credit for individual actions (Feng et al., 2025).

However, these methods still use outcome based advantages for responses or actions and do not distinguish the contributions of reasoning and executable actions in GUI interaction. As a result, they cannot provide separate learning signals that reflect the different roles of these two components.

3 PRELIMINARIES

We consider a GUI agent. The agent takes a task instruction q and a visual observation as input and produces a response containing reasoning spans and executable actions. Reasoning spans analyze the current interface and set intermediate subgoals. Executable actions change the environment through clicks, typing, scrolling, and navigation. Let τ_i be the i-th sampled trajectory, and let $y_{i,t}$ be its t-th response token. We use two disjoint masks, $m_{i,t}^{\text{act}}$ and $m_{i,t}^{\text{think}}$, to mark executable-action tokens and reasoning tokens.

GRPO For each task, the rollout policy samples a group of trajectories and assigns a scalar reward R_i to each trajectory. Under GRPO, the group-relative advantage is

$$A_i^{\text{row}} = \frac{R_i - \bar{R}}{\sigma_R + \epsilon}, \qquad \bar{R} = \frac{1}{G}\sum_{j=1}^{G} R_j, \tag{1}$$

where σ_R is the spread of rewards in the group and ϵ is a small constant for numerical stability. All relevant tokens in the same trajectory share this trajectory-level advantage.

OPSD OPSD uses a teacher with privileged context to give process-level guidance on on-policy trajectories. Let π_{T}, π_{old}, and π_θ denote the teacher policy, the rollout policy, and the trainable policy. Let c_i^+ and c_i^- denote the context available to the teacher and to the student. For a sampled token $y_{i,t}$, OPSD compares the teacher and the trainable policy at that position, and moves the trainable policy toward the privileged teacher. Let $\mathcal{S}_i$ be the set of reasoning-token positions in trajectory τ_i. The standard OPSD objective is

$$\mathcal{L}_{\text{OPSD}}^{\text{think}} = \frac{1}{|\mathcal{S}_i|}\sum_{t\in\mathcal{S}_i} D_{\text{KL}}\big(\pi_{\text{T}}(\cdot \mid q, c_i^+, y_{i,<t}) \,\big\|\, \pi_\theta(\cdot \mid q, c_i^-, y_{i,<t})\big). \tag{2}$$

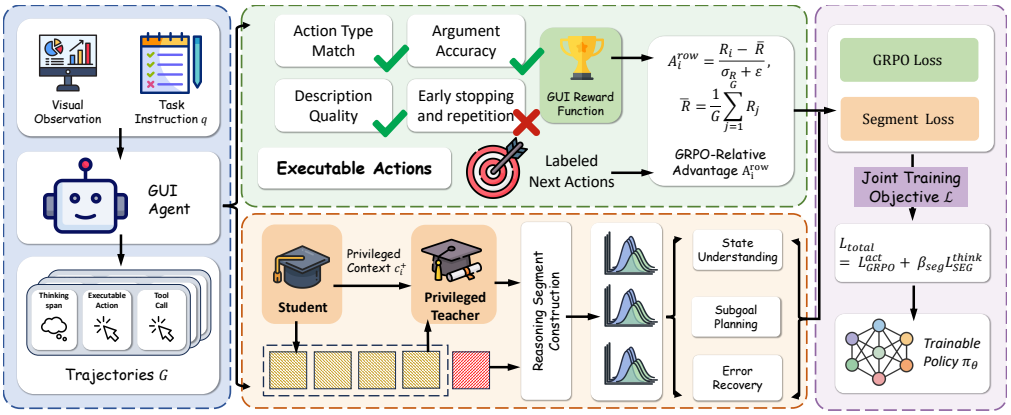


Figure 2: Overview of our framework.

Here, privileged information refers specifically to the gold executable action. Because the teacher observes this action whereas the rollout policy does not, their token preferences may differ.

4 Methodology

We propose an asymmetric post-training framework for GUI agents that optimizes executable actions with Group Relative Policy Optimization and provides local learning signals for reasoning tokens through reasoning-specific segment level OPSD. In this section, we first introduce GRPO and OPSD, and then describe the GUI reward, reasoning segment design and joint training objective.

4.1 Asymmetric post-training

4.1.1 GUI reward and action-level GRPO

The reward function compares the predicted computer-use action with the labelled next action. It combines four signals: action-type matching, argument accuracy, functional action progress, and description quality. Its general form is

$$R_i = w_{\text{type}} I_i^{\text{type}} + w_{\text{arg}} I_i^{\text{type}} r_i^{\text{arg}} + w_{\text{act}} r_i^{\text{act}} + w_{\text{desc}} r_i^{\text{desc}} + P_i^{\text{repeat}} + P_i^{\text{term}}. \tag{3}$$

Here I_i^{type} is 1 if the predicted action type matches the labelled action type, and 0 otherwise. The term r_i^{arg} scores the accuracy of coordinates, text, keyboard input, scroll distance, or other action arguments. For coordinate actions, the argument score is a smooth function of the Euclidean distance:

$$r_i^{\text{arg}} = \frac{1}{1 + (d_i/40)^2}, \tag{4}$$

where d_i is the Euclidean distance in the supplied coordinate units. The coordinate-error scale is 40. For other actions, the system scores text similarity, key-set overlap, scroll direction and magnitude, or numerical error, according to the argument type. If the label does not constrain a parameter, that parameter does not affect the reward.

When the predicted action type matches the label, the reward retains rule-based argument credit and assigns a fixed functional-action score of 1. When the action type differs, the argument term is set to zero and the judge evaluates functional progress and description quality. If the action type differs, a judge assigns credit only when the alternative action advances the task. Repeated invalid actions and premature termination are penalized. This reward distinguishes exact matches, functional alternatives and partial progress, allowing the model to explore diverse GUI paths.

We apply clipped GRPO only to executable-action tokens:

$$\mathcal{L}_{\text{GRPO}}^{\text{act}} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{t=1}^{T_i} m_{i,t}^{\text{act}} \min\Big[\rho_{i,t} A_i^{\text{row}}, \text{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon) A_i^{\text{row}}\Big], \tag{5}$$

where

$$\rho_{i,t} = \frac{\pi_\theta(y_{i,t} \mid q, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} \mid q, y_{i,<t})}. \tag{6}$$

This objective includes loss terms only at executable action token positions and uses the GUI reward rather than an action distribution provided by the teacher. It excludes direct loss terms at reasoning token positions, although updates to shared model parameters can still affect the reasoning distribution.

The complete argument-level scoring rules, invalid-prediction handling, penalties, judge protocol, and clipping behavior are provided in Appendix B.3.

4.1.2 REASONING-SPECIFIC SEGMENT OPSD

The GUI reward supervises executable actions at the trajectory level but does not identify which parts of the reasoning contribute to the final action. Directly distilling the full reasoning span would instead apply a shared teacher signal to all reasoning tokens. We therefore divide each reasoning span into contiguous, non-overlapping segments and assign each segment a relative teacher signal.

Let $\mathcal{S}_i$ denote the reasoning-token positions in trajectory τ_i, partitioned into segments $s_{i,1}, \ldots, s_{i,K_i}$. For each reasoning token $y_{i,t}$, we compute the log-probability gap between the privileged teacher and the rollout policy as

$$d_{i,t} = \text{stopgrad}\left[\log \pi_{\text{T}}(y_{i,t} \mid q, c_i^+, y_{i,<t}) - \log \pi_{\text{old}}(y_{i,t} \mid q, c_i^-, y_{i,<t})\right]. \tag{7}$$

We average these gaps within each segment and centre the result by the mean gap of the full reasoning span. The segment signal is applied only when the trajectory has a negative group-relative advantage:

$$\widetilde{d}_{i,j} = \frac{1}{|s_{i,j}|} \sum_{t \in s_{i,j}} d_{i,t} - \frac{1}{|\mathcal{S}_i|} \sum_{t \in \mathcal{S}_i} d_{i,t}, \tag{8}$$

$$w_{i,j} = \mathbb{I}[A_i^{\text{row}} < 0]\, \widetilde{d}_{i,j}, \tag{9}$$

$$\mathcal{L}_{\text{SEG}}^{\text{think}} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{K_i} \sum_{t \in s_{i,j}} \widehat{w}_{i,j} \log \pi_\theta(y_{i,t} \mid q, c_i^-, y_{i,<t}). \tag{10}$$

Here, $\widehat{w}_{i,j}$ denotes the segment weight after normalization, clipping, and span-wise re-centering. For trajectories below the group mean, segments with stronger relative teacher preference receive positive weights, whereas those with weaker preference receive negative weights. The signal is disabled for trajectories at or above the group mean. This criterion depends on relative reward rather than a binary task-success label.

We impose a weighted zero-sum constraint on the segment weights so that the guidance is centered within each reasoning span.

$$\sum_{j=1}^{K_i} |s_{i,j}| \widehat{w}_{i,j} = 0. \tag{11}$$

The zero-sum constraint applies to the token weights only; it does not imply that the likelihood of the full reasoning span remains unchanged. The weights are detached from the teacher and rollout scores, so gradients flow only through the sampled reasoning-token log probabilities.

4.1.3 JOINT OBJECTIVE

The final asymmetric training objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{GRPO}}^{\text{act}} + \beta_{\text{seg}} \mathcal{L}_{\text{SEG}}^{\text{think}}. \tag{12}$$

The GRPO loss uses GUI rewards at action token positions, whereas the segment loss uses relative teacher guidance at reasoning token positions. The two terms therefore provide complementary supervision: GUI rewards optimize executable actions, whereas relative teacher signals rank reasoning segments within below-average trajectories.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETTINGS

Datasets and metrics. We evaluated our framework on two computer-use benchmarks. **OS-World** (Xie et al., 2024) places an agent in a real operating-system environment and measures task success across five application domains: OS, Office, Daily, Professional, and Workflow. It contains 369 tasks across five domains. The domain composition and task counts are provided in Table 10; the overall success rate is computed over all 369 tasks rather than by averaging the five domain-level percentages. We report the per-domain success rate and the overall success rate across all tasks. **OmniACT** (Kapoor et al., 2024) evaluates whether the agent produces the correct action sequence for a given screen and instruction; we report the Sequence Score (SS) and the Action Score (AS). All metrics are reported as percentages, and higher is better.

Baselines. We compared AsyPO with the untrained backbone and seven post-training strategies: OPSD, offline GRPO, online GRPO, GiGPO, GRPO+OPSD, AgentOPSD, and SDAR. All trained on the same trajectories and evaluated under the same protocol. *base* denotes the backbone without post-training. *OPSD* applies on-policy self-distillation to the full response. *offline GRPO* and *online GRPO* apply Group Relative Policy Optimization with offline and online rollouts, respectively. *GRPO+OPSD* combines both signals over the whole response. *GiGPO* (Feng et al., 2025) estimates relative credit across complete trajectories and alternative actions sampled from the same state, whereas *AgentOPSD* (Wang et al., 2026b) recursively propagates teacher–student probability differences across interaction turns to assign credit from sparse outcomes. *SDAR* (Lu et al., 2026a) applies on-policy self-distillation as a gated auxiliary objective on top of GRPO.

Implementation details. We instantiated the framework on two backbones, **Qwen3-VL-4B-Thinking** and **Qwen3-VL-8B-Thinking** (Team, 2025). Executable-action tokens are optimized by clipped action-level GRPO, while reasoning tokens receive the reasoning-specific segment OPSD signal; the two objectives are combined through the joint loss $\mathcal{L}_{\text{total}} = \mathcal{L}^{\text{act}}_{\text{GRPO}} + \beta_{\text{seg}}\mathcal{L}^{\text{think}}_{\text{SEG}}$ (Eq. 12). Each reasoning span is divided into contiguous 8-token segments. The segment signal is gated to trajectories whose group-relative advantage is negative and re-centred to zero within each span. We used AdamW with an initial learning rate of 1×10^{-6}, a rollout group size of $G{=}8$, and a training batch size of 64. The post-training experiments used a single node with eight NVIDIA H100 80GB GPUs. The preceding SFT stage used 16 NVIDIA H800 GPUs. Detailed implementation information is provided in Appendix B. For OmniACT, we follow the official evaluation and reporting protocol.

5.2 OVERALL COMPARISON

Table 1 reports the overall comparison on OSWorld and OmniACT for both backbones. For each domain in OSWorld, we calculate the domain-level success rate as the number of completed tasks divided by the total number of tasks in that domain, and report the result rounded to one decimal place. The overall success rate is calculated as the total number of completed tasks divided by the total number of OSWorld tasks. The task counts for each domain are provided in Table 10 in Appendix B.2.

Comparison with SDAR Our method improves over the strongest baseline, SDAR, by $+2.2$ overall and $+1.0/+1.2$ on SS/AS. On Qwen3-VL-4B-Thinking it reached 37.1 overall, a $+3.8$ gain over SDAR, and led on all five OSWorld domains as well as both OmniACT metrics. The gains were largest on the reasoning-heavy domains. On the 4B backbone the framework improved OS by $+8.3$ and Professional by $+8.2$ over SDAR, while consistent gains appeared on OmniACT. This shows the effectiveness of coupling action-level GRPO with the segment-level reasoning signal.

Comparison with trajectory-level baselines. We group *GiGPO*, *online GRPO*, and *AgentOPSD* as trajectory-level baselines because they estimate credit from trajectory outcomes. Among these methods, *online GRPO* achieves the strongest overall performance. Our method improves over it by $+2.4$ points for Qwen3-VL-8B-Thinking and $+4.1$ points for Qwen3-VL-4B-Thinking, with gains

Table 1: Overall performance on OSWorld and OmniACT. Best per column in **bold**; our method is highlighted, with Δ improvement over SDAR (deeper colour indicates a larger gain). All values are percentages.

Method	OSWorld						OmniACT	
	OS	Office	Daily	Professional	Workflow	overall	SS	AS
Qwen3-VL-4B-Thinking								
base	50.0	29.9	30.8	61.2	11.9	30.6	42.9	30.5
OPSD	45.8	18.8	29.5	55.1	10.9	25.5	41.2	32.2
offline GRPO	**54.2**	31.6	42.3	59.2	8.9	32.8	44.3	34.1
online GRPO	33.3	30.8	46.2	59.2	12.9	33.1	-	-
GiGPO	41.7	29.9	44.9	57.1	13.9	33.1	-	-
GRPO+OPSD	45.8	26.5	46.2	59.2	11.9	32.2	44.1	33.9
AgentOPSD	41.7	29.9	44.9	61.2	10.9	32.8	-	-
SDAR	45.8	31.6	46.2	57.1	10.9	33.3	45.9	35.6
AsyPO	**54.2**	**34.2**	**47.4**	**65.3**	**14.9**	**37.1**	**46.8**	**37.5**
Δ *vs SDAR*	**+8.3**	**+2.6**	**+1.3**	**+8.2**	**+4.0**	**+3.8**	**+0.9**	**+1.9**
Qwen3-VL-8B-Thinking								
base	50.0	29.9	32.1	63.3	11.9	31.2	43.8	33.5
OPSD	50.0	21.4	33.3	61.2	5.0	26.6	41.6	28.2
offline GRPO	54.2	33.3	44.9	61.2	8.9	34.1	45.6	35.0
online GRPO	50.0	32.5	47.4	63.3	13.9	35.8	-	-
GiGPO	33.3	30.8	46.2	59.2	12.9	33.1	-	-
GRPO+OPSD	50.0	28.2	47.4	63.3	12.9	34.1	45.4	34.8
AgentOPSD	45.8	31.6	46.2	63.3	11.9	34.4	-	-
SDAR	54.2	34.2	47.4	63.3	11.9	36.0	47.9	37.3
AsyPO	**58.3**	**35.0**	**50.0**	**65.3**	**14.9**	**38.2**	**48.9**	**38.5**
Δ *vs SDAR*	**+4.2**	**+0.9**	**+2.6**	**+2.0**	**+3.0**	**+2.2**	**+1.0**	**+1.2**

across all five OSWorld domains. This demonstrates that the proposed segment-level reasoning guidance provides additional benefits beyond trajectory-level credit assignment.

Comparison with Symmetric Baselines The framework clearly outperforms the two baselines that treat reasoning and action tokens symmetrically. *OPSD* distils the whole response, while *offline GRPO* assigns a single score to the entire response, so neither separates reasoning from action. AsyPO improved overall OSWorld success over OPSD by more than $+11$ on both backbones. This margin supports the central design choice: reasoning and action tokens are better served by different objectives than by a single shared signal.

Overall, the three comparisons show that our gain comes from coupling action-level GRPO with a segment-level reasoning signal, since it beats the strongest baseline SDAR, the symmetric single-signal objectives, and outcome-only online GRPO across both backbones and benchmarks.

5.3 Training After SFT

To assess the contribution of the asymmetric post SFT reinforcement learning stage, we decomposed it into an action-level GRPO branch and a reasoning level segment OPSD branch, and evaluated the resulting variants on OSWorld. All variants start from the same Qwen3-VL-8B-Thinking checkpoint after three epochs of SFT, so their scores are higher than those in Table 1. *SFT* evaluates this checkpoint without reinforcement learning. *w/ OPSD* retains reasoning distillation but removes action-level GRPO. *w/ Segment* removes the GRPO reward while retaining segment level averaging. *w/ GRPO* retains action-level GRPO without the segment signal. *Ours* uses the full asymmetric objective. As shown in Fig. 3, *Ours* achieves the highest overall score, whereas *w/ OPSD* performs worst among the reinforcement learning variants. These results support combining action level optimization with segment level reasoning guidance after SFT.

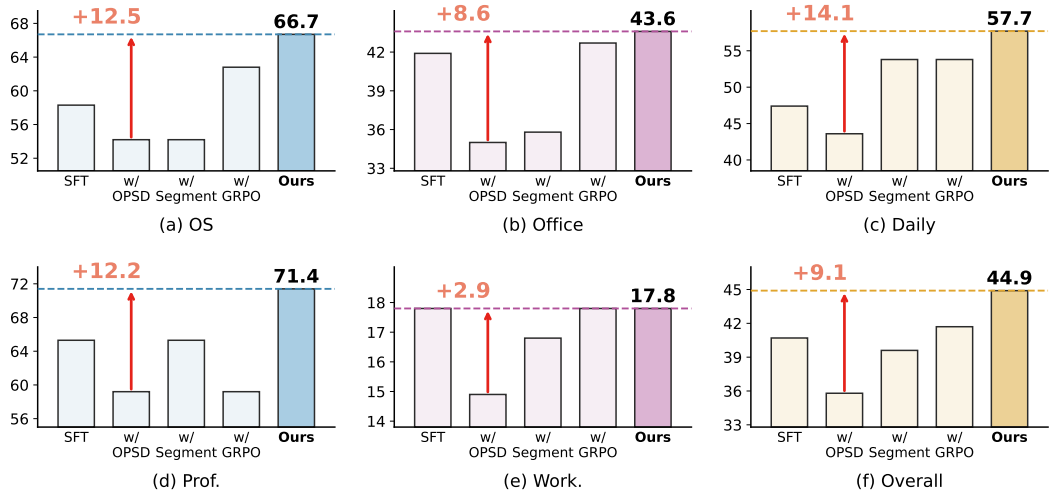


Figure 3: Comparison of training strategies after supervised fine-tuning on OSWorld. The panels report success rates as percentages across five application domains and overall for the SFT checkpoint and the subsequent training variants. All variants start from the same SFT checkpoint.

5.4 Ablation Study

We conducted all ablations on the more challenging OSWorld benchmark with the larger Qwen3-VL-8B-Thinking backbone, which exercises the complete reasoning–action pipeline and provides a stable and representative setting for isolating each component.

Reward component analysis. We progressively added rule-based rewards, judge scores, and penalty terms to action-only GRPO to evaluate the contribution of each GUI reward component, without using segment level reasoning guidance. In Table 2, + *rule*, + *judge*, and + *penalty* denote the corresponding cumulative settings, with + *penalty* representing action-only GRPO with the complete GUI reward. Its overall success rate of 35.50% matches the action-only GRPO result in Table 3. This cumulative ablation shows how progressively adding reward components improves action-only GRPO under the tested configuration. We also report *Ours* as a reference, which adds segment level reasoning guidance and improves the overall success rate to 38.21%.

Table 2: Comparison of GUI reward and training configurations on OSWorld. All values are success rates expressed as percentages, with higher values indicating better performance. The best result in each column is shown in bold.

Config	OS	Office	Daily	Prof.	Work.	Overall
base	50.0	29.9	32.1	63.3	11.9	31.2
+ rule	54.2	30.8	37.2	63.3	11.9	32.8
+ judge	54.2	32.5	39.7	63.3	11.9	33.9
+ penalty	54.2	**35.9**	39.7	**67.3**	11.9	35.5
full (Ours)	**58.3**	35.0	**50.0**	65.3	**14.9**	**38.2**

Optimization signal analysis. Table 3 compares GRPO and OPSD when applied to different token ranges. For GRPO, optimizing only action tokens achieves an overall success rate of 35.5%, which is higher than the 34.7% obtained by optimizing the full response. This suggests that restricting reward driven updates to action tokens is more effective. For OPSD, optimizing only reasoning tokens achieves an overall success rate of 18.4%, lower than the 26.6% obtained by optimizing the full response. These results suggest that OPSD requires a more suitable optimization design for the thinking tokens and that reasoning distillation alone cannot replace effective action optimization. Together, these findings provide empirical support for combining action-level GRPO with segment averaged optimization of the reasoning tokens.

Zero-sum recentering. We average teacher rollout log probability gaps within each reasoning segment and subtract their token weighted mean across the full reasoning span. As shown in Table 3, Under the same action reward and training budget, the ablation results support both steps: *GRPO*

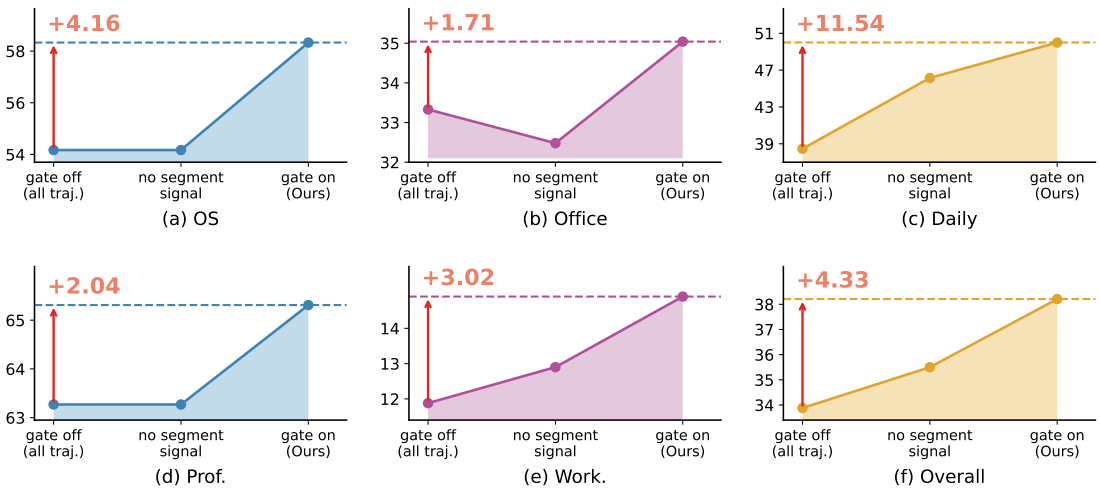


Figure 4: Ablation of segment guidance and negative-advantage gating.

Table 3: Ablation study of training signals, segment averaging and segment centering.

Method	OS	Office	Daily	Prof.	Work.	overall
Thinking-only OPSD	25.0	12.8	28.2	40.8	5.0	18.4
Full-response OPSD	50.0	21.4	33.3	61.2	5.0	26.6
Action-only GRPO	54.2	35.9	39.7	67.3	11.9	35.5
Full-response GRPO	50.0	**38.5**	32.1	**69.4**	11.9	34.7
GRPO w/ OPSD	50.0	28.2	47.4	63.3	12.9	34.1
GRPO w/ Segment mean	**58.3**	36.8	41.0	65.3	12.9	36.3
AsyPO (Ours)	**58.3**	35.0	**50.0**	65.3	**14.9**	**38.2**

w/ OPSD, without segment averaging or centering, achieves 34.1%, *GRPO w/ Segment mean*, with only within segment averaging, reaches 36.3%, and the *AsyPO* with span level centering reaches 38.2%. It also outperforms action-only GRPO at 35.5%, indicating that the processed reasoning signal provides additional benefits for action optimization.

Negative advantage gate. The segment signal is applied only to trajectories with negative group relative advantage through $\mathbb{I}[A_i^{\text{row}} < 0]$ in Equation 9. This selects rollouts whose dense rewards fall below the group mean rather than necessarily indicating task failure. As shown in Fig 4, applying the signal to all trajectories reduced the overall score to 34.4, while removing it entirely increased the score to 35.5. The gated model achieved 38.2, supporting the use of below average trajectories for reasoning shaping.

6 Conclusion

We introduced an asymmetric post-training framework for GUI agents that applies action-level GRPO to executable actions and segment level OPSD to reasoning. The reasoning signal assigns relative weights to segments within trajectories whose rewards fall below the group mean, using negative-advantage gating and centering within each reasoning span. Experiments on two computer use benchmarks and two model scales showed higher reported scores than the evaluated baselines. The ablations suggest benefits from segment guidance and its associated design choices under the tested configurations. These results support the combined framework as a promising approach to training GUI agents with mixed reasoning and action outputs.

AI USE STATEMENT

For manuscript preparation, we used generative AI tools to improve the grammar and clarity of author-written text. Separately, during model training, we used a large language model accessed via an API as a reward evaluator. Given the task instruction, interface screenshot, reference action, and model-predicted action, the evaluator assessed how effectively the predicted action advanced the task and the quality of its accompanying description. AI use was limited to these purposes. All AI-assisted text was reviewed by the authors to ensure that it introduced no new factual, methodological, or numerical content. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work involves no human subjects, personally identifiable information, or new data collection, and uses only the publicly available OSWorld and OmniACT benchmarks under their licenses. All evaluation is confined to their sandboxed environments; we do not deploy the trained agents on real systems. We note that GUI agents able to operate real interfaces carry a dual-use risk, and recommend human oversight and action confirmation before any real-world deployment. We are aware of no conflicts of interest.

REPRODUCIBILITY STATEMENT

The training and evaluation configurations are provided in the main text and Appendix B. These include data splits, model initialization, reward definitions, teacher and judge prompts, segmentation rules, random seeds, evaluation protocol, optimization settings, evaluation task counts, decoding budgets, and run counts. The eight-seed stability results use the same base models, data, optimizer settings, rollout budget, and fixed training budget for all compared methods. Code, prompts, and evaluation scripts will be released upon acceptance.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations, ICLR*. OpenReview.net, 2024.

Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2024.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, 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), ACL*, pp. 9313–9332. Association for Computational Linguistics, 2024.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems , NeurIPS*, 2025.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *Computer Vision - ECCV 2024 - 18th European Conference*, volume 15126, pp. 161–178, 2024.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *CoRR*, abs/2607.05184, 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. *CoRR*, abs/2508.14040, 2025.

Meng Li, Lei Li, Xiting Wang, Yi Yuan, Zheng Wei, Jiang Bian, and Zang Li. SOAR: supervision from observation for agentic reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pp. 35175–35197. Association for Computational Linguistics, 2026.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning, 2026a.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, and Yueting Zhuang. 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), ACL*, pp. 9474–9496. Association for Computational Linguistics, 2026b.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P. Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

Zhanming Shen, Jintao Tong, Shaotian Yan, Chen Shen, Hao Chen, Wentao Ye, Xiaomeng Hu, Rui Miao, Haobo Wang, Junbo Zhao, Gang Chen, and Jieping Ye. Purified OPSD: on-policy self-distillation without losing how to think. *CoRR*, abs/2607.02234, 2026.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys, Rotterdam, The Netherlands*, pp. 1279–1297. ACM, 2025.

Qwen Team. Qwen3 technical report, 2025.

Junzhe Wang, Zhiheng Xi, Yajie Yang, Hao Luo, Shihan Dou, Tao Gui, and Qi Zhang. Enhancing llm-based search agents via contribution weighted group relative policy optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pp. 31704–31718. Association for Computational Linguistics, 2026a.

Zi-Han Wang, Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Jie Wu, Zhengzhou Cai, Yueqing Sun, Ziang Ye, Linji Hao, Qi Gu, Xunliang Cai, Yongliang Shen, and Yujiu Yang. Agentopsd: Recursive self-distillation for agentic reinforcement learning, 2026b.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems*, 2024.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous GUI interaction. In *Forty-second International Conference on Machine Learning, ICML 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems , NeurIPS*, 2025.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, Dunjie Lu, Haikong Lu, Qi Zhen, Xinyuan Wang, Jiaqi Deng, Yuhao Yang, Cheng Chen, Boyuan Zheng, Alex Su, Xiao Yu, Hao Zou, Saaket Agashe, Xing Han Lu, Manpreet Kaur, Zhengyang Qi, Vincent Sunn Chen, Frederic Sala, Dayiheng Liu, Junyang Lin, Zhou Yu, Yu Su, Siva Reddy, Xin Eric Wang, Peng Qi, Tianbao Xie, and Tao Yu. Osworld2.0: Benchmarking computer use agents on long-horizon real-world tasks. *CoRR*, abs/2606.29537, 2026.

Xiangjian Zeng, Wenjing Li, Qingqiang Wu, and Liang Zhang. Fdc-ground: Improving GRPO for GUI grounding via exponential rewards and fact-aligned pruning. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI*, pp. 28122–28130. AAAI Press, 2026.

Li Zhang, Longxi Gao, and Mengwei Xu. Does chain-of-thought reasoning help mobile GUI agents? an empirical study. In *Findings of the Association for Computational Linguistics, ACL*, pp. 7981–7996, 2026a.

Ziyun Zhang, Zezhou Wang, Xiaoyi Zhang, Zongyu Guo, Jiahao Li, Bin Li, and Yan Lu. Infiniteweb: Scalable web environment synthesis for GUI agent training. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics, ACL*, pp. 28465–28492, 2026b.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *CoRR*, abs/2601.18734, 2026.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, et al. Swift: a scalable lightweight infrastructure for fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 29733–29735, 2025.

A Supplementary Experiments

A.1 Reward Weight

As shown in Table 4, both rule-based and model judged rewards have limited effectiveness when used alone. The judge-only configuration performs worst in this ablation, possibly because its supervision is relatively sparse, while the rule-based reward performs better but lacks semantic evaluation. Combining them with weights of 0.6 and 0.4 achieves the best or tied-best performance in all five domains and overall (38.21%), demonstrating the necessity of our hybrid reward design.

A.2 Segment loss weight

The segment loss coefficient in Equation 12 balances reasoning guidance with action optimization. Setting $\beta_{\text{seg}} = 0$ recovers vanilla GRPO. As shown in Table 5, removing the segment loss or increasing its weight to 1 reduced overall success. The best score of 38.21 was obtained at $\beta_{\text{seg}} = 0.05$. This result shows that the reasoning signal is most effective when it guides rather than dominates the action objective.

Table 4: Ablation on the reward blend weight (rule vs. judge). All entries are task success rates in percentages. Overall denotes the success rate over the full OSWorld evaluation set.

Config (rule/judge)	OS	Office	Daily	Prof.	Work.	Overall
judge only (0.0/1.0)	50.0	29.9	42.3	59.2	12.9	33.1
judge-heavy (0.4/0.6)	54.2	31.6	43.6	61.2	13.9	34.7
rule only (1.0/0.0)	54.2	33.3	44.9	63.3	14.9	36.0
Ours (0.6/0.4)	**58.3**	**35.0**	**50.0**	**65.3**	**14.9**	**38.2**

Table 5: Ablation on the segment loss weight β_{seg} (OSWorld, % ↑).

β_{seg}	OS	Office	Daily	Prof.	Work.	overall
0	54.2	35.9	39.7	67.3	11.9	35.5
0.01	**58.3**	**37.6**	41.0	**69.39**	12.9	37.1
0.05 (Ours)	**58.3**	35.0	**50.0**	65.3	**14.9**	**38.2**
0.1	**58.3**	**37.6**	42.3	**69.4**	12.9	37.4
0.5	54.2	35.0	39.7	65.3	12.9	35.2
1	54.2	32.5	37.2	63.3	12.9	33.6

A.3 Reasoning segment granularity

We varied the segment length to study its effect on credit assignment. As shown in Table 6, treating each token in the thinking span as a separate segment produced the lowest score of 34.15. Performance improved as the number of tokens per segment increased and reached 38.21 at 8 tokens per segment, before declining at 16/32 tokens per segment. These results show that moderate segmentation provides useful local credit, whereas too few segments lose local information and too many weaken the signal.

In this experiment, the single token setting treats each reasoning token as a separate segment, rather than merging the entire reasoning span into one segment. In this setting, the mean within each segment reduces to the difference in log probability between the teacher and the rollout policy for that token, from which we still subtract the mean difference over the entire reasoning span. Centering therefore removes only the mean offset shared by all tokens and does not generally reduce all guidance weights to zero, so this setting retains relative guidance at the token level. In contrast, when the entire reasoning span forms a single segment, the segment mean equals the span mean, yielding zero weights after centering. This setting is therefore equivalent to disabling segment guidance for reasoning. To avoid confusion, we label these settings as "one token per segment (token-level centering)" and "one segment per span ($K = 1$)", respectively.

A.4 Stability Evaluation

To evaluate training stability, we independently trained each method with eight random seeds and report the mean performance across the resulting final checkpoints. We started from the same Qwen3-VL-4B-Thinking and Qwen3-VL-8B-Thinking base model and trained each method with eight independent random seeds. All experiments used a fixed 20 optimizer updates and a batch size of 64, while keeping the training data, number of rollouts, optimizer settings, sampling configuration and total training budget identical. After each training run, we evaluated the final checkpoint obtained after the 20th optimizer update directly, without selecting the best model from multiple intermediate checkpoints based on test-set performance. Therefore, the Avg@8 results reported in Table 7 represent the average performance across eight independent training runs, rather than repeated evaluations of a single checkpoint. Because OSWorld evaluation is computationally expensive, we compare only our method with SDAR. We selected SDAR because it is the strongest baseline after our method, while the other baselines show substantially larger performance gaps. These results demonstrate that the advantages of our method are robust.

Table 6: Ablation on the number of tokens per reasoning segment.

Segmentation setting	OS	Office	Daily	Prof.	Work.	overall
1 token/segment	54.2	33.3	38.5	65.3	11.9	34.2
2 tokens/segment	**58.3**	36.8	41.0	**69.4**	12.9	36.9
4 tokens/segment	**58.3**	**37.6**	42.3	**69.4**	**14.9**	37.9
8 tokens/segment (Ours)	**58.3**	35.0	**50.0**	65.3	**14.9**	**38.2**
16 tokens/segment	**58.3**	35.0	41.0	67.3	12.9	36.0
32 tokens/segment	54.2	34.2	38.5	65.3	11.9	34.4
1 segment/span	54.2	35.9	39.7	67.3	11.9	35.5

Table 7: Mean performance over eight independently trained runs with different random seeds for Asymmetric and SDAR. Each run uses the final checkpoint after the fixed training budget.

Method	**OSWorld**						**OmniACT**	
	OS	Office	Daily	Professional	Workflow	overall	SS	AS
Qwen3-VL-4B-Thinking								
SDAR	$45.8_{\pm 3.2}$	$31.6_{\pm 0.7}$	$46.2_{\pm 1.0}$	$57.1_{\pm 1.5}$	$10.9_{\pm 0.8}$	$33.3_{\pm 0.7}$	$45.7_{\pm 0.2}$	$35.5_{\pm 0.2}$
AsyPO	$\mathbf{58.9}_{\pm 2.7}$	$\mathbf{34.6}_{\pm 0.4}$	$\mathbf{48.7}_{\pm 0.7}$	$\mathbf{62.7}_{\pm 0.9}$	$\mathbf{13.0}_{\pm 0.4}$	$\mathbf{37.0}_{\pm 0.1}$	$\mathbf{46.7}_{\pm 0.2}$	$\mathbf{36.5}_{\pm 0.2}$
Qwen3-VL-8B-Thinking								
SDAR	$54.2_{\pm 3.2}$	$34.2_{\pm 0.7}$	$47.4_{\pm 1.0}$	$63.3_{\pm 1.6}$	$11.9_{\pm 0.8}$	$36.1_{\pm 0.7}$	$47.8_{\pm 0.2}$	$37.2_{\pm 0.2}$
AsyPO	$\mathbf{58.9}_{\pm 2.7}$	$\mathbf{35.4}_{\pm 0.4}$	$\mathbf{49.6}_{\pm 0.9}$	$\mathbf{64.5}_{\pm 1.0}$	$\mathbf{15.0}_{\pm 0.4}$	$\mathbf{38.2}_{\pm 0.1}$	$\mathbf{48.8}_{\pm 0.2}$	$\mathbf{38.4}_{\pm 0.2}$

B Supplementary Implementation Details

B.1 Usage of Judge Model

We use DeepSeek-V4-Flash as the judge model, and the prompt is shown in Fig 5. The reasoning-specific segment guidance requires one batched teacher log-probability evaluation for each training update. With a batch size of 64 and eight rollouts per task, each update contains up to 512 rollouts, while the default 20-step training round processes approximately 20,480 rollouts. The reward function uses the DeepSeek-V4-Flash judge through an API, with at most one logical judge call per rollout; failed requests are retried up to three total attempts. In practice, the observed first-attempt failure rate was approximately 5%, and all requests recovered successfully after retries. The judge increases the training time from approximately 8 to 10 minutes per optimizer update, corresponding to a 25% overhead over vanilla GRPO. A complete run using eight GPUs for approximately six hours consumes about 48 GPU-hours.

B.2 Implementation Details

This section describes the training and evaluation protocols, including data composition, optimization settings, computational resources, and the OSWorld evaluation procedure. The main SFT and GRPO hyperparameters are summarized in Tables 8 and 9.

The SFT corpus was internally generated and constructed independently of all evaluation tasks. No OSWorld or OmniACT test instruction, screenshot, trajectory, or task identifier was used during SFT or post-training. For OmniACT, training and evaluation strictly follow the official train/test split, and the test split is used only for final evaluation.

OSWorld evaluation. We evaluate the agents using the official OSWorld interaction loop and evaluator. As shown in Table 10, the evaluation set contains 369 tasks distributed across five domains. Each task allows at most 50 interaction steps, and all experiments use a screen resolution of 1920×1080. During evaluation, model inference uses a temperature of 0.0 and a maximum generation length of 32,768 tokens.

Supervised fine-tuning. We conduct supervised fine-tuning with MS-SWIFT (Zhao et al., 2025) on 16 NVIDIA H800 GPUs. The training mixture consists of 362,979 samples, which are shuffled

API Reward Prompt

You grade one step of a GUI agent. The agent's action does NOT string-match the ground-truth action, so your job is to decide how wrong it really is.
You get the current screenshot, the task instruction, the actions taken so far, the GROUND-TRUTH next action, and the agent's predicted action plus its one-line natural-language Action description.
Return TWO scores.
action -- does the predicted action move the task forward? 1 = functionally equivalent to the ground truth. Same effect on the UI, or a different but equally valid way to reach the same next state (e.g. pressing Enter instead of clicking the confirmed default button, double_click vs left_click on an item that opens either way). 0.5 = PARTIALLY CORRECT. Not equivalent, but it still advances the task: it targets the right UI element with the wrong gesture, or performs a reasonable intermediate step (scrolling to bring the target into view, opening the right menu by a different route) that leaves the task recoverable and closer to done. 0 = does not advance the task: wrong target, no effect, or it takes the task somewhere the ground-truth path cannot recover from cheaply.
Judge the EFFECT on the task, not the wording. Do not give 0.5 merely because the action is a plausible-looking GUI action; it must genuinely make progress.
desc -- is the one-line description concise and specific? 1 = right verb + one specific identifiable target, single clear phrase. 0.5 = intent readable but vague, verbose, or the target is described loosely. 0 = it hedges or enumerates several possibilities to cover all cases, or it does not describe the action the agent actually took. Enumerating alternatives is reward hacking and must score 0.
Return STRICT JSON only: {action: 0|0.5|1, desc: 0|0.5|1, reason: <brief>}

Figure 5: Prompt for the judge model.

before training. We train each model for three epochs with bfloat16 precision and use the checkpoint from the final epoch for subsequent evaluation. The global batch size is 512 for Qwen3-VL-8B-Thinking and 2,048 for Qwen3-VL-4B-Thinking. We use a learning rate of 2×10^{-5}, a cosine learning-rate schedule, and a warmup ratio of 0.05. The maximum sequence length is 16,384 tokens. To reduce memory consumption, we enable gradient checkpointing and ZeRO-3, and use sequence parallelism with a parallel size of 4. The maximum number of image pixels is set to 2,073,600.

Reinforcement learning. We implement GRPO with verl (Sheng et al., 2025). For each task, the rollout policy samples a group of G responses, and each response receives a trajectory-level reward R_i from the executable evaluator. We normalize rewards within each group to obtain the relative advantage:

$$A_i^{\text{row}} = \frac{R_i - \bar{R}}{\hat{\sigma}_R + \epsilon}. \tag{13}$$

Here, $\bar{R}$ and $\hat{\sigma}_R$ are the within-group mean and standard deviation of the trajectory rewards, respectively.

Table 8: SFT training configuration.

Hyperparameter	Value
Framework	MS-SWIFT
Hardware	16×H800
Training epochs	3
Learning rate	2×10^{-5}
LR schedule	Cosine
Warmup ratio	0.05
Per-device batch size	1
Gradient accumulation	64
Maximum sequence length	16,384
Maximum image pixels	2,073,600
Precision	bfloat16
Sequence parallel size	4
Memory optimization	ZeRO-3
Gradient checkpointing	Enabled
Freeze vision encoder	Yes
Freeze vision–language aligner	Yes

Table 9: Post-training configuration for GRPO and segment-level reasoning guidance.

Hyperparameter	Value
Learning rate	1×10^{-6}
Task batch size	64
Rollouts per task	8
Entropy coefficient	0
Clip ratio	$0.2/0.2$
Clip-ratio cap	3.0
Rollout temperature	1.0
Maximum sequence length	14,336
Maximum prompt length	12,288
Maximum response length	2,048
Maximum interaction turns	50
Segment loss coefficient	$\beta_{\text{seg}} = 0.05$
Segment length	8 tokens
Negative-advantage gating	Enabled
Teacher guidance	Segment-level OPSD
GUI reward judge	DeepSeek-V4-Flash
Training hardware	8×H100 80GB
Training time	Approximately 6 hours
GPU hours	48

The action-level objective applies the group-relative trajectory advantage only at executable-action positions:

$$\mathcal{L}_{\text{GRPO}}^{\text{act}} = -\frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{T_i} m_{i,t}^{\text{act}} \min\left[\rho_{i,t}A_i^{\text{row}}, \text{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)A_i^{\text{row}}\right]. \tag{14}$$

The likelihood ratio is defined as

$$\rho_{i,t} = \frac{\pi_\theta(y_{i,t} \mid q, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} \mid q, y_{i,<t})}. \tag{15}$$

The binary mask $m_{i,t}^{\text{act}}$ excludes non-action tokens from the direct GRPO loss.

Before reinforcement learning, we use the SFT checkpoint to generate eight rollouts for each candidate task and evaluate their outcomes. We remove groups in which all rollouts are successful or

Table 10: Domain composition of the OSWorld evaluation set.

Domain	Applications	# Tasks
OS	os	24
Office	libreoffice_calc, libreoffice_impress, libreoffice_writer	$47 + 47 + 23 = 117$
Daily	chrome, vlc, thunderbird	$46 + 17 + 15 = 78$
Professional	gimp, vs_code	$26 + 23 = 49$
Workflow	multi_apps	101
Overall	All domains	369

all are unsuccessful, because such groups do not provide useful within-group reward variation for GRPO. This filtering procedure retains 6,628 tasks for reinforcement-learning training.

We use a learning rate of 1×10^{-6} and a task batch size of 64. Each task is sampled with eight rollouts at temperature 1.0. The clipping range is set to $0.2/0.2$, with the likelihood-ratio cap set to 3.0. The maximum sequence length is 14,336 tokens, including up to 12,288 prompt tokens and 2,048 response tokens. Training proceeds for approximately 20 optimizer steps. We used a random seed of 123 for the training experiments.

B.3 Complete GUI reward definition

The implementation uses

$$(w_{\text{type}}, w_{\text{arg}}, w_{\text{act}}, w_{\text{desc}}) = (0.25, 0.25, 0.35, 0.15).$$

For type-correct actions, $r_i^{\text{act}} = 1$ and the argument score is computed by action-specific rules. For type-mismatched actions, $r_i^{\text{arg}} = 0$, and the judge returns functional-action and description scores.

$$r_i^{\text{arg}} = \begin{cases} \dfrac{1}{1 + (d_i/40)^2}, & a_i \in \mathcal{A}_{\text{coord}}, \\ \text{sim}_{\text{text}}(\hat{z}_i, z_i), & a_i \in \{\text{type}, \text{answer}\}, \\ \dfrac{|K_i \cap K_i^\star|}{\max(|K_i|, |K_i^\star|)}, & a_i = \text{key}, \\ \text{scroll score}, & a_i \in \{\text{scroll}, \text{hscroll}\}, \\ \text{normalized numerical score}, & a_i = \text{wait}, \\ \mathbb{I}(z_i = z_i^\star), & a_i = \text{terminate}, \\ 1, & \text{if the reference does not constrain the argument.} \end{cases} \tag{16}$$

The coordinate-action set is

$$\mathcal{A}_{\text{coord}} = \{\texttt{left_click}, \texttt{right_click}, \texttt{middle_click}, \texttt{double_click}, \texttt{triple_click}, \texttt{mouse_move}, \texttt{left_click_drag}\}.$$

Coordinate distance is Euclidean, with scale 40. Text and answer arguments use the lowercase Python `SequenceMatcher` ratio.

The premature-termination penalty is -0.5 when the predicted action is `terminate` but the labelled action is not. The repeat penalty is -0.3 under the configured repetition condition, using a coordinate tolerance of 10 and a labelled-coordinate margin of 30.

Unusable labels and invalid or missing `computer_use` calls receive zero reward. An unclosed `<tool_call>` is marked truncated. Judge fields are clamped to $[0, 1]$; malformed JSON or missing fields are treated as judge failures. The API timeout is 90 s, and `API_RETRIES=2` permits up to three HTTP attempts. Screenshot retrieval uses a 20-s timeout and falls back to text-only input on failure. There is no additional global final reward clipping.

B.4 SEGMENT OPSD IMPLEMENTATION

Let

$$\mathcal{T}_i = \{t : m_{i,t}^{\text{think}} = 1\}$$

denote the valid reasoning-token positions in trajectory τ_i. Padding and action-token positions are excluded. The valid positions are partitioned into contiguous, non-overlapping segments $s_{i,1}, \ldots, s_{i,K_i}$ with a configured length of 8 tokens. Empty reasoning spans contribute zero segment loss.

For each valid reasoning token, we compute the detached teacher-rollout gap

$$d_{i,t} = \text{stopgrad}\left[\log \pi_{\text{T}}(y_{i,t} \mid q, c_i^+, y_{i,<t}) - \log \pi_{\text{old}}(y_{i,t} \mid q, c_i^-, y_{i,<t})\right]. \tag{17}$$

$$\bar{d}_{i,j} = \frac{1}{|s_{i,j}|} \sum_{t \in s_{i,j}} d_{i,t}, \tag{18}$$

$$\widetilde{d}_{i,j} = \bar{d}_{i,j} - \frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} d_{i,t}, \tag{19}$$

$$w_{i,j} = \mathbb{I}[A_i^{\text{row}} < 0]\widetilde{d}_{i,j}. \tag{20}$$

With `SEG_W_NORMALIZE=True`, the valid gated weights are normalized before being clipped to $[-3, 3]$ under `SEG_W_CLIP=3.0`. The clipped weights are then re-centred within each span using the token-weighted mean:

$$\sum_{j=1}^{K_i} |s_{i,j}| \widehat{w}_{i,j} = 0. \tag{21}$$

The launch script exposes the segment bounds and weighting controls, but the underlying trainer source is not included in the repository. Therefore, the exact normalization statistic, numerical stabilizer, and final remainder-packing behavior are implementation-specific. All processed weights are detached, so gradients flow only through the student reasoning-token log-probabilities.