SEEING WHAT CHANGED: TRANSITION-GUIDED CREDIT ASSIGNMENT FOR GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) for multi-turn GUI agents provides terminal outcome supervision but leaves credit assignment across individual turns unresolved. Uniformly broadcasting a trajectory-level advantage can reinforce both useful actions and mistakes that the agent subsequently corrects. Screenshots before and after each action expose state changes that help distinguish progress from detours, even within successful trajectories. We propose **Observable-Transition Advantage Decomposition (OTAD)**, which uses this evidence to guide turn-level credit assignment. A vision-language model provides hindsight ordinal judgments of each transition conditioned on the verified outcome. OTAD maps these judgments to non-negative weights on the trajectory-level advantage rather than additional rewards: transition evidence sets the magnitude of each turn's advantage while the terminal verifier fixes its sign. This design removes the need for a learned value model and for extra environment interaction. The same judgments guide budget-constrained turn selection and prioritize tasks with discriminative transition structure, improving training-data efficiency. On OSWorld-Verified, OTAD raises success from 42.2% under budget-matched GRPO to 52.1%; at 8B parameters, it surpasses ComputerRL-9B and OpenCUA-72B, while also improving GUI grounding.

1 INTRODUCTION

Consider a real desktop trajectory: to complete its task, the agent opens the wrong menu, backs out, navigates again, and only several turns later reaches the target state; the programmatic outcome verifier then returns success. In reinforcement learning with verifiable rewards (RLVR) (Bai et al., 2024; Qi et al., 2025; Lai et al., 2026), standard trajectory-level GRPO (Shao et al., 2024) uses this outcome to compute a group-relative advantage that is broadcast uniformly across the trajectory's turns. In a group with mixed outcomes, the mis-click, the undo, and the step that completes this successful trajectory therefore receive the same positive advantage, despite their different contributions. This is not a corner case: our transition scorer labels 12% of the turns in successful training trajectories as detours or regressions (Section 4.5). Under a 50-step interaction budget, dozens of decisions share a single bit of outcome supervision that carries no information about *which* turns produced it. Turn-level credit assignment is therefore the central bottleneck of multi-turn GUI RLVR, and it sharpens as the interaction horizon grows.

Three natural routes each face a concrete obstacle. Process reward models and learned critics require reliable intermediate assessments of progress or expected return, yet GUI applications, layouts, and task types vary widely, making these assessments difficult to learn and calibrate (Lightman et al., 2024; Cui et al., 2026; Zhou et al., 2025). Counterfactual resampling estimates a turn's marginal contribution directly, but requires restoring intermediate states and running additional rollouts from each of them (Kazemnejad et al., 2025); we use such continuations only for offline analysis (Section 4.5). Adding a dense signal as per-turn rewards on top of the outcome reward changes the optimized return; unless the signal satisfies the condition for return equivalence, it can bias optimization away from success (Arjona-Medina et al., 2019; Wang et al., 2026a).

The GUI environment already exposes evidence relevant to this problem: screenshots before and after each action show what changed in the interface. Conditioned on the task and trajectory context, these observable transitions help distinguish progress from detours that a terminal outcome alone

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: Overview of OTAD. Pre- and post-action states provide hindsight ordinal evidence for reweighting the verifier's trajectory-level advantage across turns; the verifier fixes the sign of each turn's advantage. The same evidence supports turn selection and credit-aware task sampling.

cannot separate, and they reflect what an action did rather than whether its text appears plausible. Our key idea is to use this evidence to guide credit assignment in advantage space. A natural reference for turn-level credit is the change in conditional expected success after executing an action and observing the resulting state; these Doob martingale increments sum to the deviation of the outcome from its prior expectation. Computing them requires calibrated intermediate success estimates, so we instead ask a vision-language model to judge the direction and relative strength of each transition's contribution in hindsight, conditioned on the verified outcome, and use the increments to evaluate how well these judgments rank turns.

These observations motivate **Observable-Transition Advantage Decomposition (OTAD)**. OTAD treats the pre-action screenshot, action, and post-action screenshot as the atomic unit of credit assignment. A fixed, outcome-conditioned rule turns each ordinal judgment into a non-negative weight on the trajectory-level advantage, rather than an additional reward. The verifier determines the sign of each turn's advantage, while the observed transition determines its magnitude. Unhelpful updates can thus be masked. The same judgments also guide training-data allocation at two levels. Under a fixed turn budget, OTAD retains informative turns while maintaining coverage of action types. Across tasks, it considers both transition-score discriminability and an online estimate of success rate. It then prioritizes tasks near the capability frontier. As shown in Figure 1.

Trained with OTAD, Qwen3-VL-8B-Thinking (Bai et al., 2025b) reaches **52.1%** on OSWorld-Verified (Xie et al., 2024; 2025b), 9.9 points above budget-matched GRPO (42.2%) under the same 50-step limit. Component ablations show that advantage weighting contributes the largest gain, with further improvements from turn selection and task sampling. The transition scores correlate with continuation-based estimates of turn-level credit, and OTAD also improves single-turn GUI grounding. Our main contributions are as follows:

1. We identify observable interface changes as evidence for distinguishing turn-level contributions within GUI trajectories, and use the Doob increment of conditional expected success as a reference for evaluating ordinal transition judgments, which need not be calibrated value estimates.
2. We propose OTAD, which reweights trajectory-level advantages with hindsight ordinal transition evidence rather than adding process rewards. It needs no learned value model and no counterfactual rollouts. The same evidence guides turn selection and task sampling.
3. OTAD gains 9.9 points over budget-matched GRPO on OSWorld-Verified, surpasses larger open-source models, and improves GUI grounding. All three components contribute. Adding the scores as rewards or shuffling them across turns loses most or all of the gain.

2 Related Work

Online Reinforcement Learning for Computer-Use Agents. GUI agents observe screenshots and act through mouse and keyboard, and recent work has extended native models to long-horizon interaction on complete operating systems (Qin et al., 2025; Wang et al., 2025; 2026c; Xie et al., 2024). With programmatic outcome verifiers, group-relative policy optimization (GRPO) (Shao et al., 2024) has become a common post-training approach: DigiRL (Bai et al., 2024) and We-

bRL (Qi et al., 2025) applied it to interactive environments, followed by desktop systems such as ARPO, ComputerRL, and ScaleCUA (Lu et al., 2025; Lai et al., 2026; Lv et al., 2026). Because all-success or all-failure groups provide no relative signal, existing methods also study rollout allocation, often steering budget toward the capability frontier by success rate (Foster et al., 2026). These methods still broadcast the trajectory advantage uniformly across turns; OTAD uses observable transition evidence to refine both within-trajectory credit and task priority.

Credit Assignment in Multi-Turn Interaction. Long-horizon credit assignment classically uses learned value functions and GAE (Schulman et al., 2017; 2016) or return decomposition (Arjona-Medina et al., 2019). For LLM agents, intermediate supervision relies on process reward models (Lightman et al., 2024; Cui et al., 2026), rollouts for intermediate-value estimation (Kazemnejad et al., 2025), or critics from privileged training-time information (Zhou et al., 2025). GiGPO (Feng et al., 2026) is closest to our setting: it derives step-level advantages on top of trajectory-level GRPO without extra interaction, but groups anchor states by the policy's return statistics rather than observable transitions. Information-gain methods (Wang et al., 2026a) use a related notion of turn credit, but OTAD applies the evidence in advantage space, not the reward. Other GUI turn-level signals rely on learned step values (Bai et al., 2025a). Concurrent work also argues that evidence should set update strength, not direction (Xu et al., 2026b; Huang et al., 2026); OTAD uses visible interface changes and preserves the trajectory-level advantage.

GUI Process Rewards and Observable State Change. Process signals for GUI and web agents include step-level process reward models (Chae et al., 2026), progress rewards from trajectory position (Zhang et al., 2025), and visual state-reasoning rewards (Wang et al., 2026b). Another line replaces the programmatic verifier with a vision-language model in the reward channel (Yang et al., 2025); OTAD keeps the verifier. VisCritic (Qian, 2026) also compares screenshots before and after an action, but trains a Siamese encoder and uses the comparison as a process reward. The Next Screenshot Knows (Li et al., 2026) feeds next-screenshot evidence to a distillation teacher for action prediction. OTAD instead modulates advantages with ordinal evidence. It attributes credit in hindsight, conditioned on the verified outcome, not by online forward estimation.

3 METHOD

Figure 2 shows the training loop: sample tasks, roll out and verify groups, score each observable transition, then reweight and optimize the selected turns. We present it in logical rather than execution order: Section 3.1 defines the problem and a reference for turn-level credit, Section 3.2 the advantage decomposition, and Section 3.3 task sampling.

3.1 PROBLEM DEFINITION AND A REFERENCE FOR TURN-LEVEL CREDIT

GUI Interaction as a Multi-Turn Decision Process. A task instruction x and initial observation o_0 define $h_0 = (x, o_0)$. At turn t, the policy emits $a_t = (y_{t,1}, \ldots, y_{t,L_t}) \sim \pi_\theta(\cdot \mid h_t)$, the environment returns o_{t+1}, and $h_{t+1} = (h_t, a_t, o_{t+1})$. A trajectory of T_i turns and its binary outcome reward are

$$\tau_i = (h_{i,0},\, a_{i,0},\, o_{i,1},\, \ldots,\, a_{i,T_i-1},\, o_{i,T_i}), \qquad R_i = R(\tau_i) \in \{0, 1\}. \tag{1}$$

Each turn defines an observable transition $u_{i,t} = (x^-_{i,t}, a_{i,t}, x^+_{i,t})$, where $x^-_{i,t}$ and $x^+_{i,t}$ are the screenshots before and after executing $a_{i,t}$. OTAD assigns credit to this transition rather than to the action text. We construct one candidate training segment per turn and apply the policy gradient only to that turn's response, masking all conditioning tokens. Figure 3 illustrates the segment context, and turn selection determines which candidates enter training.

Group-Relative Trajectory Advantage. GRPO (Shao et al., 2024) samples G trajectories per task and standardizes their outcome rewards:

$$A^{\text{ep}}_i = \frac{R_i - \bar{R}}{\hat{\sigma}_R + \epsilon}. \tag{2}$$

Standard GRPO copies A^{ep}_i onto every response token. If all G trajectories succeed or fail, then $\hat{\sigma}_R = 0$ and the group carries no relative signal, so we discard it. OTAD instead constructs turn-dependent advantages $A_{i,t}$ whose direction still comes from the verifier.

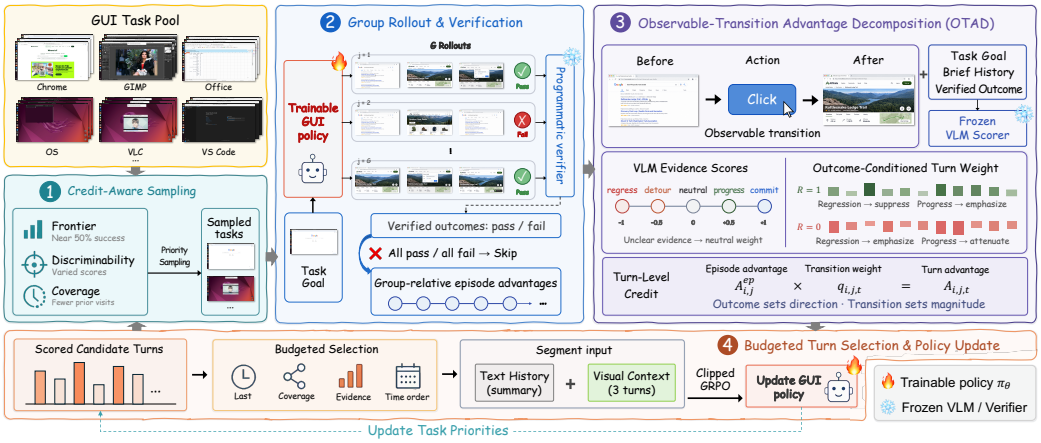

Figure 2: **OTAD training loop.** (1) Tasks are sampled by frontier difficulty, transition-score discriminability and coverage. (2) The policy rolls out G trajectories, which the verifier labels to give group-relative episode advantages. (3) A frozen vision-language scorer judges each observable transition, and an outcome-conditioned mapping turns the judgment into a non-negative turn weight: the verifier fixes the sign of a turn's advantage, the transition its magnitude. (4) Turns selected under a budget are optimized with the clipped GRPO objective. The scorer is used only in training.

A Reference for Turn-Level Credit. Let $\mathcal{F}_{i,t} = \sigma(h_{i,t})$ and define the conditional outcome process $M_{i,t} = \mathbb{E}[R_i \mid \mathcal{F}_{i,t}]$. Its Doob increment

$$\delta^\star_{i,t} = M_{i,t+1} - M_{i,t} \tag{3}$$

measures the information about eventual success revealed by executing action $a_{i,t}$ and observing the new interface state. When the terminal history determines the outcome, $\sum_{t=0}^{T_i-1} \delta^\star_{i,t} = R_i - M_{i,0}$ (Appendix A.1). Evaluating it requires calibrated success probabilities at intermediate GUI states. OTAD therefore does not estimate $\delta^\star_{i,t}$; Section 4.5 uses continuation-based estimates of it to evaluate how well the ordinal scores rank turns.

3.2 Observable-Transition Advantage Decomposition

Hindsight Ordinal Scoring. For each transition, a vision-language scorer receives the task goal x, verified outcome R_i, a compact history digest, and $u_{i,t}$, and returns

$$s_{i,t} \in \{-1,\ -0.5,\ 0,\ 0.5,\ 1\} \cup \{\perp\}. \tag{4}$$

Positive levels denote observable progress, negative levels denote regression or an avoidable detour, 0 denotes no confirmed net progress, and $\perp$ denotes insufficient evidence or malformed output. Since the scorer observes R_i, this is hindsight attribution rather than online estimation of $M_{i,t}$; detailed input constraints and the rubric are given in Appendix C.

Outcome-Conditioned Credit Mass and Turn-Level Advantage. OTAD maps the ordinal score to a non-negative credit mass conditioned on R_i, $q_{i,t} = m_{R_i}(s_{i,t}; \alpha)$ with $\alpha \in (0, 0.5]$. The mapping in Table 1 recovers the uniform weight $q = 1$ for $s = 0$ and $s = \perp$, can suppress clearly harmful transitions in successful trajectories, and reduces the penalty on locally useful actions in failed trajectories. The turn-level advantage is

$$A_{i,t} = A^{\text{ep}}_i \cdot q_{i,t}. \tag{5}$$

Since $q_{i,t} \geq 0$, transition evidence scales or masks each turn's advantage without reversing its verifier-derived sign. This design prevents scorer errors from reversing a turn's policy-gradient contribution, while reducing the penalty on useful actions in failed trajectories. Empirically, the non-negative mapping outperforms signed credit (52.1% vs. 47.5%; Appendix F.4), with a widening gap under score noise (Figure 4); an outcome-independent mapping lowers success to 48.9%. Appendix A.3 details the sign guarantee and its scope.

Table 1: Ordinal transition levels and the induced credit mass $q_{i,t} = m_{R_i}(s_{i,t}; \alpha)$.

		Credit mass $q_{i,t}$	
$s_{i,t}$	**Criterion on the observable transition** $x^-_{i,t} \to x^+_{i,t}$	$R_i{=}1$	$R_i{=}0$
$+1$	Reaches or commits a state the task requires	$1+\alpha$	$1-2\alpha$
$+0.5$	Moves toward such a state without reaching it	$1+\alpha/2$	$1-\alpha$
0	No net change, or change unrelated to the goal	1	1
-0.5	Avoidable detour that a later turn must undo	$1-\alpha$	$1+\alpha/2$
-1	Harmful or decisive move toward an incorrect state	0	$1+\alpha$
$\perp$	Evidence insufficient or scorer output invalid	1	1

Algorithm 1 OTAD Training

Require: Tasks $\mathcal{T}$, policy π_θ, group size G, budget B, α
Initialize $(\hat{p}_i, n_i, d_i) \leftarrow (0.5, 0, 0)$
for each training iteration **do**
  Compute $\omega_i = f_i \lambda_i \eta_i$; sample tasks $\mathcal{B}$
  Roll out and verify G trajectories/task; update $\hat{p}_i, n_i$
  for each task group with $\hat{\sigma}_{R,i} > 0$ **do**
    Compute trajectory advantages $A^{\text{ep}}_{i,j}$
    for each trajectory $\tau_{i,j}$ **do**
      Obtain hindsight transition scores $s_{i,j,t}$
      Select $|\mathcal{S}_{i,j}| \le B$ turns by action type and evidence
      Build segments; mask context (Fig. 3)
      $A_{i,j,t} \leftarrow A^{\text{ep}}_{i,j} m_{R_{i,j}}(s_{i,j,t}; \alpha)$ for $t \in \mathcal{S}_{i,j}$
    end for
    Update d_i from valid transition scores
  end for
  Update π_θ with $\mathcal{L}_{\text{OTAD}}$ on selected segments
end for

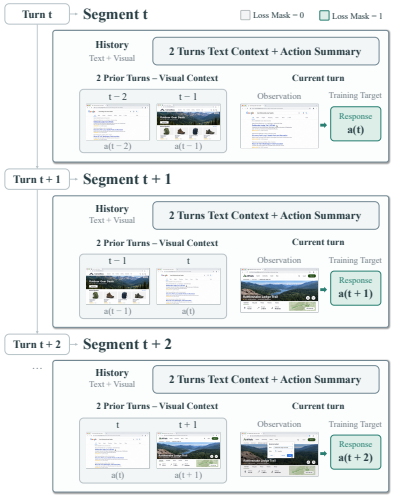

Figure 3: **OTAD training and segment construction. Left:** training loop with budget-constrained turn selection. **Right:** each segment combines a history summary, two prior interactions, and the current screenshot; only the target response (green) contributes to the policy loss.

Budget-Constrained Turn Selection. For a turn budget B, OTAD retains a support set $\mathcal{S}_i$ by keeping the last trainable turn, distributing the remaining budget across observed action types, ranking turns within each type by outcome-relevant transition evidence, and filling residual slots chronologically (Algorithm 1). Each selected turn contributes one segment with the context and loss mask in Figure 3; adjacent segments in the illustration need not all be selected. Because the objective below normalizes by the response tokens of the selected turns, each turn advantage factors as $A_{i,t} = \widetilde{A}_i \, w_{i,t}$ into a trajectory-level strength $\widetilde{A}_i$ and a within-trajectory allocation $w_{i,t}$ with token-weighted mean one (Appendix A.2); this factorization is the decomposition in OTAD's name.

Policy Objective. Let $\kappa_i(n)$ map response tokens to their turns, $r_{i,n}(\theta)$ be the importance ratio, and $c_{i,n} \in \{0, 1\}$ mask effective response tokens. OTAD optimizes

$$\mathcal{L}_{\text{OTAD}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G} \frac{1}{\sum_n c_{i,n}} \sum_n c_{i,n} \min\Big(r_{i,n} A_{i,\kappa_i(n)}, \; \text{clip}(r_{i,n}, 1{-}\varepsilon, 1{+}\varepsilon) A_{i,\kappa_i(n)}\Big) + \beta \, \mathcal{L}_{\text{KL}}, \tag{6}$$

which replaces the uniform advantage in the clipped GRPO objective with $A_{i,\kappa_i(n)}$. The scorer does not enter the gradient, and no reward-prediction or process-supervision loss is introduced.

3.3 Credit-Aware Adaptive Task Sampling

OTAD also determines which tasks receive rollout budget. Uniform sampling can produce all-success or all-failure groups with $\hat{\sigma}_R = 0$ and under-sample tasks near the capability frontier, so we reuse transition scores to define a task priority.

Frontier Weight from Online Difficulty. For each task x_i, we maintain an exponential moving average $\hat{p}_i$ of its group success rate and assign high weight to tasks near 0.5:

$$f_i = \exp\left(-\frac{(\hat{p}_i - 0.5)^2}{2\tau^2}\right). \tag{7}$$

Discriminability of Transition Credit. We measure discriminability by the variance of valid transition scores, $d_i = \mathrm{Var}\big(\{s_{i,j,t} : s_{i,j,t} \neq \perp\}\big)$, normalized into $\lambda_i \in [1-\delta, 1+\delta]$.

Composite Priority. With coverage decay $\eta_i = (n_i + 1)^{-\gamma}$, the task priority is $\omega_i = f_i\,\lambda_i\,\eta_i$. We sample most tasks without replacement in proportion to ω_i and reserve an $\varepsilon_{\mathrm{explore}}$ fraction for uniform exploration. Appendix E analyzes the frontier kernel and the bounded variance correction. Algorithm 1 summarizes the complete training loop.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on OSWorld-Verified (Xie et al., 2024; 2025b). Its 369 real desktop tasks span five categories. A programmatic verifier scores each task as binary success, with no partial credit. We exclude the 8 Workflow tasks that require Google Drive access, leaving 361 (OS 24, Office 117, Daily 78, Professional 49, Workflow 93). The agent observes screenshots only and is capped at 50 interaction steps. We run three independent rollout passes per task and report the mean success rate with its standard deviation across them. We also report single-turn click accuracy on ScreenSpot-Pro (Li et al., 2025a) and OSWorld-G (Xie et al., 2025a).

Baselines. We compare against proprietary computer-use agents (GPT-5.5, Claude Opus 4.6 / Sonnet 4.5 / Sonnet 4, Seed1.8, OpenAI CUA) and open-source models (Kimi K2.5, EvoCUA-32B, GUI-Owl-1.5-32B from Mobile-Agent-v3.5, UI-TARS-2, ComputerRL-9B, OpenCUA-72B / 32B, DART-GUI-7B, Qwen3.5-9B).

Implementation. We apply OTAD to Qwen3-VL-8B-Thinking (Bai et al., 2025b), with the training task pool synthesized by the pipeline of Appendix B. The transition scorer is a frozen Qwen3.6-27B (Qwen Team, 2026b) used during training only. RL uses $G = 8$ rollouts per task, a global batch of 128, 40 optimizer steps, a constant $\alpha = 0.4$ and a turn budget $B = 16$. Task priority and transition-score-based sampling apply only to multi-turn GUI tasks; grounding tasks follow a separate uniform-random sampling branch. Hyperparameters and infrastructure are given in Appendix F.1, sensitivity to α and B in Appendix F.5, and a train–test overlap audit in Appendix F.3.

4.2 Main Results

OSWorld. OTAD-Qwen3-VL-8B reaches **52.1%** on OSWorld-Verified, 20.0 points above the backbone without post-training, which we evaluate under the same protocol (Table 2). Under the same 50-step budget, its largest margins over ComputerRL-9B are on Daily and Office, at 11.6 and 6.9 points, respectively. OTAD also outperforms OpenCUA-72B on four categories and matches it on Professional.

GUI grounding. OTAD also improves single-turn GUI grounding accuracy (Table 3). After SFT, budget-matched GRPO drops from 58.1% to 57.4% on ScreenSpot-Pro, whereas OTAD improves to 60.3%. Both improve on OSWorld-G, with OTAD 2.4 points above GRPO. These results show that OTAD improves desktop task success while also strengthening single-turn grounding.

4.3 Ablations and Controlled Comparisons

All RL comparisons start from the same SFT checkpoint and share the task pool, hyperparameters and the 50-step interaction cap. The reference, *budget-matched GRPO*, uses the same per-turn segments and turn budget $B = 16$: it keeps the last turn and fills the remaining budget by drawing turns

Table 2: Success rate (%) on OSWorld-Verified. Ours: mean over three passes on the 361 evaluated tasks; subscripts are standard deviations. Baseline scores and step budgets are taken from the cited official reports; – marks entries that are not reported. Claude Sonnet scores follow Anthropic (2025c). Qwen3-VL-8B-Thinking (Bai et al., 2025b) is our backbone.

Model	Steps	OS	Office	Daily	Prof.	Workflow	Overall
Proprietary models							
GPT-5.5 (OpenAI, 2026)	–	–	–	–	–	–	78.7
Claude Opus 4.6 (Anthropic, 2026)	–	–	–	–	–	–	72.7
Seed1.8 (Seed, 2026)	100	66.7	68.8	67.1	71.4	42.4	61.9
Claude Sonnet 4.5 (Anthropic, 2025b)	100	–	–	–	–	–	61.4
Claude Sonnet 4 (Anthropic, 2025a)	100	–	–	–	–	–	42.2
OpenAI CUA (OpenAI, 2025)	50	70.8	23.0	37.2	51.0	15.8	31.3
Open-source models							
Kimi K2.5 (Team et al., 2026)	100	73.9	69.2	66.4	73.5	46.1	63.3
EvoCUA-32B (Xue et al., 2026)	50	78.3	59.8	64.5	81.6	27.9	56.7
GUI-Owl-1.5-32B (Xu et al., 2026a)	–	–	–	–	–	–	56.5
UI-TARS-2 (Wang et al., 2025)	100	41.7	61.1	62.1	61.2	34.1	53.1
ComputerRL-9B (Lai et al., 2026)	50	75.0	45.8	52.5	71.4	27.9	48.0
OpenCUA-72B (Wang et al., 2026c)	50	45.8	42.7	53.8	75.5	22.3	44.9
DART-GUI-7B (Li et al., 2025b)	30	62.5	39.3	50.8	73.5	16.7	42.1
Qwen3.5-9B (Qwen Team, 2026a)	–	–	–	–	–	–	41.8
OpenCUA-32B (Wang et al., 2026c)	50	60.9	29.9	41.0	65.3	14.6	35.1
Qwen3-VL-8B-Thinking	50	$50.0_{\pm 4.2}$	$29.9_{\pm 1.5}$	$35.9_{\pm 1.7}$	$63.3_{\pm 2.4}$	$10.8_{\pm 1.1}$	$32.1_{\pm 0.8}$
Ours							
OTAD-Qwen3-VL-8B	50	$66.7_{\pm 4.2}$	$52.7_{\pm 1.3}$	$64.1_{\pm 1.3}$	$75.5_{\pm 2.0}$	$25.1_{\pm 1.2}$	$\mathbf{52.1}_{\pm 1.0}$

Table 3: GUI grounding accuracy (%) on two single-turn benchmarks. Baseline rows and the Base row are taken from the respective public reports.

Model	ScreenSpot-Pro	OSWorld-G
UI-TARS-7B (Qin et al., 2025)	35.7	47.5
EvoCUA-32B (Xue et al., 2026)	49.8	63.9
Qwen3-VL-32B-Thinking (Bai et al., 2025b)	57.1	64.0
OpenCUA-72B (Wang et al., 2026c)	60.8	67.0
Qwen3-VL-8B-Thinking (Base)	46.6	56.7
+ SFT	58.1	65.8
+ SFT + GRPO	57.4	66.7
+ SFT + OTAD	**60.3**	**69.1**

at random within action-type buckets. It broadcasts the trajectory advantage uniformly and samples tasks uniformly. Since 62% of training trajectories exceed 16 turns, the budget binds for most trajectories under both methods. The scorer's training-time computation is not matched (Appendix F.6). Each ablation changes only the annotated item relative to full OTAD.

Training stages. Table 4 compares post-training stages. SFT warm-up on synthetic trajectories from the same pipeline as the RL task pool (Appendix B.2) raises success from 32.1% to 41.7%. Budget-matched GRPO then reaches 42.2% (+0.5 over SFT), whereas full OTAD reaches 52.1% (+10.4). Removing the turn budget from GRPO gives 42.5%, so GRPO's small gain over SFT is not an artifact of turn selection. Under the same interaction and optimization budget, OTAD thus uses the collected trajectories more effectively than uniform broadcasting.

Component ablation. Table 5 replaces one component of full OTAD at a time. Setting all transition credit to neutral, $q_{i,t} = 1$, while retaining score-guided turn selection and task sampling costs 8.3 points (to 43.8%), the largest of the three drops. Using transition evidence to select training data is therefore insufficient for the full gain; it must also modulate turn-level advantages. Replacing score ranking with random draws within the same action-type buckets, i.e., the selector used by GRPO, costs 3.9 points at the same segment count and token budget, isolating the effect of score-guided selection. Uniform task sampling costs 2.7 points, so rollout allocation also matters.

Table 4: Training stages. OSWorld success rates (%).

Model	Success
Base	32.1
+ SFT	41.7
+ SFT + GRPO	42.2
+ SFT + GRPO (B=all)	42.5
+ SFT + OTAD	**52.1**

Table 5: Component ablations. OSWorld success rates (%).

OTAD variant	Success
Full OTAD	**52.1**
Unit credit (q=1)	43.8
Random turns	48.2
Uniform tasks	49.4

Table 6: Score-use controls. OSWorld success rates (%).

Score use	Success
OTAD weights	**52.1**
Added rewards	40.8
Shuffled scores	43.5

Table 7: Transition scoring quality. Rows 1–4 compare the scorer with human labels; the rest measure human agreement, repeatability (three identical requests at fixed temperature) and unscorable outputs. Definitions are in Appendix C.

Measure	Interpretation	Value
Direction agreement (%)	Same contribution direction as humans	88.5
Within-one-level agreement (%)	At most one ordinal level from the human label	91.3
Exact-level agreement (%)	Identical five-level score to the human label	61.2
Rank correlation	Spearman ρ with human labels	0.79
Human agreement	Inter-annotator Krippendorff α	0.71
Repeat agreement (%)	Identical scores across three requests	93.6
Unscorable outputs (%)	Insufficient evidence or invalid output ($\perp$)	6.8

How transition scores enter training. With the scorer, scores and budget fixed (Table 6), adding the scores as per-turn rewards before group standardization reaches only 40.8%, 1.4 points below uniform broadcasting. A score useful for ranking need not give a calibrated additive reward scale, which favours modulating advantages over changing returns (Appendix A.4). A second control permutes scores within each trajectory, preserving the turn support set and score distribution. Success falls to 43.5%, so the gain depends on aligning evidence with specific turns, not on non-uniform weights alone. Appendix F.4 gives further mapping controls.

4.4 Transition Scorer Analysis

Agreement with human annotation. Three external annotators independently label 600 stratified scorable transitions using the rubric in Table 1. Direction agreement is 88.5%, against 61.2% exact agreement over the five levels, and rank correlation with human labels is 0.79 (Table 7). The scorer is therefore reliable for the direction and ranking that OTAD uses (Section 3.2).

Input-channel ablation. We change one input channel of the frozen scorer at a time and retrain the policy on the resulting scores (Table 8). Removing the post-action screenshot hurts most, at 6.8 points: observing the resulting state change matters more than judging from pre-action information alone. Structured text summaries of the screenshots recover only part of it. Removing the verified outcome and the history digest cost 4.0 and 1.5 points, so both add attribution context. Direction agreement and task success rank these variants identically.

Robustness. Figure 4 corrupts scorer outputs to a random other level at rate p. The gap between our non-negative mapping and a signed mapping admitting negative credit widens with p, reaching 9.8 points at p=0.4. A frozen copy of the initial policy as the scorer reaches 50.8%, retaining 87% of OTAD's gain, so a stronger scorer is not needed. Flipping the outcome label changes the sign of only 9.5% of scores: the judged direction follows the observable change, not the label. Outcome conditioning therefore acts mainly where the evidence is ambiguous (Appendix F.6).

4.5 Mechanism Analysis

The gain grows with the interaction horizon. We split the 361 evaluation tasks into short, medium and long tertiles (121, 120 and 120 tasks) by human demonstration length, rather than by the model's own trajectory length, which is endogenous to success. Overall success is the task-weighted mean of the three bin rates. In Figure 5, the margin of OTAD over budget-matched GRPO grows from $+4.9$ points in the short bin to $+10.6$ in the medium and $+14.2$ in the long bin.

Table 8: Scorer input ablations. Full input: task instruction, pre- and post-action screenshots, action, history digest and verified outcome; each row changes one input. Dir.: direction agreement on the annotated subset of Table 7. OSWorld: overall success.

Input change	Dir. (%)	OSWorld (%)
Full input	88.5	**52.1**
Remove post-action screenshot	71.2	45.3
Replace screenshots with text summaries	74.5	46.8
Remove outcome from scorer input	79.8	48.1
Remove history digest	85.1	50.6

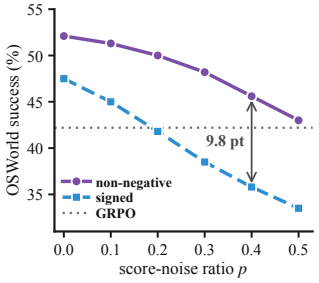

Figure 4: OSWorld success under score noise.

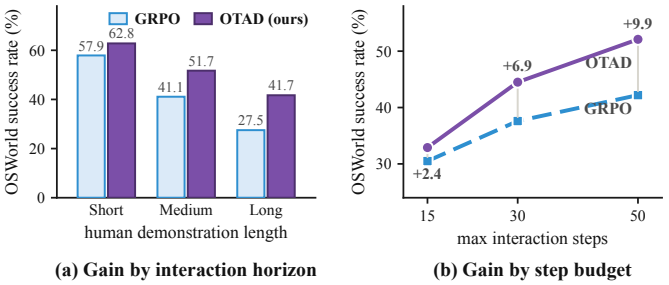

Figure 5: OSWorld success (%) of GRPO and OTAD, by human demonstration length (left) and step budget (right).

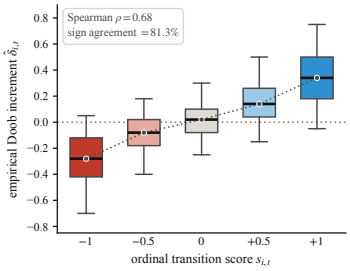

Figure 6: Empirical credit increments by ordinal score.

Across budgets of 15, 30 and 50 steps it is 2.4, 6.9 and 9.9 points. Because the binning gives only three observations and correlates with task difficulty, we treat this as descriptive rather than causal.

Ordinal scores track empirical credit increments. Using the snapshots of Appendix B.1, we fix a mid-training checkpoint, restore the environment at every turn boundary of sampled trajectories, roll out 16 continuations to termination from each, and subtract adjacent verifier success rates to get empirical credit increments. In Figure 6 the median increment rises monotonically with ordinal level, Spearman's ρ is 0.68 and sign agreement is 81.3%, so the scores preserve the ranking of empirical increments. Since a snapshot restores the full environment state rather than only the observable history, these rates approximate $M_{i,t}$ rather than estimate it exactly.

Selective reweighting of turn-level credit. Among selected turns, a mean of 6.5 per trajectory (46%) get non-neutral credit ($q_{i,t} \neq 1$), and 5% of those in successful trajectories are masked ($s_{i,t} = -1$). Detours and regressions ($s_{i,t} \leq -0.5$) are 12% of all turns in successful training trajectories, which uniform broadcasting would reinforce along with nearby progress. The share of negative levels falls from 23% to 11% over training; this statistic is not directly optimized, but its decline is consistent with improving behaviour. Appendices F.7 and F.9 give curves and cases.

5 Conclusion

We studied turn-level credit assignment for multi-turn GUI agents trained with verifiable rewards, where the observable state change before and after each action provides intermediate evidence that a terminal outcome lacks. We proposed **OTAD**, which uses hindsight ordinal transition judgments to reweight the trajectory-level advantage while the programmatic verifier fixes the sign of each turn's advantage, and reuses the same evidence for turn selection and task sampling, without a learned value model or counterfactual rollouts. On OSWorld-Verified, OTAD improves over budget-matched GRPO by 9.9 points to 52.1%, surpasses larger open-source models, and improves GUI grounding. Ablations show that transition credit, score-guided selection, and task sampling each contribute to the overall gain, and the ordinal scores track empirical credit increments. Our results suggest that observable interface transitions provide a practical basis for fine-grained credit assignment in critic-free GUI reinforcement learning.

AI USE STATEMENT

Generative AI tools enter this work in two roles. As components of the method, a frozen vision-language model serves as the training-time transition scorer (Section 3.2), and an LLM proposes candidate task instructions during task-pool construction, each of which must pass filtering, verifier testing and the overlap audit before use (Appendices B and F.3); the scorer and agent prompts are given in Appendices C and F.2. In preparing the manuscript, LLMs were used mainly to polish language and presentation. The authors reviewed all AI-assisted text and take full responsibility for the content of this paper, including its text, claims and artifacts.

ETHICS STATEMENT

The human labels used to validate the transition scorer (Section 4.4) were produced by external annotators, who labeled screenshots and action text from trajectories collected in sandboxed desktop environments using the rubric in Table 1. All training and evaluation interactions run in isolated virtual machines, and no real user accounts or personal data are involved. GUI agents capable of operating general software could be misused for unauthorized automation; our experiments are confined to sandboxed benchmark environments, and deployed agents should operate under user authorization and confirmation for consequential actions.

REPRODUCIBILITY STATEMENT

Section 3 specifies the ordinal-to-weight mapping, the turn-level advantage, budget-constrained turn selection and task sampling; Algorithm 2 gives the selection procedure, and Appendix A contains the derivations. Appendix B describes task-pool construction and verifier testing, Appendix F.3 the train–test overlap audit, and Appendix F.1 the full training configuration and hardware. The complete transition-scorer prompt and rubric and the GUI-agent prompt are provided in Appendices C and F.2. The trained OTAD model weights and source code will be released publicly upon acceptance.

REFERENCES

Anthropic. System card: Claude Opus 4 & Claude Sonnet 4. https://www.anthropic.com/claude-4-system-card, May 2025a.

Anthropic. System card: Claude Sonnet 4.5. https://www.anthropic.com/claude-sonnet-4-5-system-card, September 2025b.

Anthropic. Introducing Claude Sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5, September 2025c.

Anthropic. System card: Claude Opus 4.6. https://www.anthropic.com/claude-opus-4-6-system-card, 2026.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495, 2024.

Hao Bai, Yifei Zhou, Li Li, Sergey Levine, and Aviral Kumar. Digi-Q: Learning VLM Q-value functions for training device-control agents. In *International Conference on Learning Representations*, 2025a.

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*, 2025b.

Hyungjoo Chae, Seonghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, et al. Web-Shepherd: Advancing PRMs for reinforcing web agents. *Advances in Neural Information Processing Systems*, 38:63314–63356, 2026.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Thomas Foster, Anya Sims, Johannes Forkel, and Jakob Foerster. LILO: Learning to reason at the frontier of learnability. *Advances in Neural Information Processing Systems*, 38:48941–48974, 2026.

Mianqiu Huang, Taofeng Xue, Chong Peng, Jinrui Ding, Jie Yang, Sicheng Fan, Jiale Hong, Yufei Gao, Xiaocheng Zhang, Linsen Guo, Xin Yang, et al. EvoCUA-1.5: Online reinforcement learning for multi-turn computer-use agents. *arXiv preprint arXiv:2607.09773*, 2026.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron C Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

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*, volume 2026, pp. 23553–23591, 2026.

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

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

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*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

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*, 2025.

Bowen Lv, Xiao Liu, Yanyu Ren, Hanyu Lai, Bohao Jing, Hanchen Zhang, Yanxiao Zhao, Shuntian Yao, Jie Tang, and Yuxiao Dong. ScaleCUA: Scaling computer use agents with verifiable task synthesis and efficient online RL. *arXiv preprint arXiv:2607.11185*, 2026.

OpenAI. Computer-Using Agent: Powering Operator with a universal interface for AI to interact with the digital world. `https://openai.com/index/computer-using-agent/`, January 2025.

OpenAI. GPT-5.5 system card. `https://openai.com/index/gpt-5-5-system-card/`, April 2026.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, et al. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 79791–79821, 2025.

Jiachen Qian. VisCritic: Visual state comparison as process reward for GUI agents. *arXiv preprint arXiv:2606.24525*, 2026.

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

Qwen Team. Qwen3.5-9B model card. `https://huggingface.co/Qwen/Qwen3.5-9B`, February 2026a.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model. `https://qwen.ai/blog?id=qwen3.6-27b`, April 2026b.

John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

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

Bytedance Seed. Seed1.8 model card: Towards generalized real-world agency. *arXiv preprint arXiv:2603.20633*, 2026.

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

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Ziwei Chai, Y Charles, HS Che, Cheng Chen, et al. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, et al. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In *International Conference on Learning Representations*, 2026a.

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

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, et al. VAGEN: Reinforcing world model reasoning for multi-turn VLM agents. *Advances in Neural Information Processing Systems*, 38: 172871–172933, 2026b.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, et al. OpenCUA: Open foundations for computer-use agents. *Advances in Neural Information Processing Systems*, 38:139756–139806, 2026c.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025a.

Tianbao Xie, Mengqi Yuan, Danyang Zhang, Xinzhuang Xiong, Zhennan Shen, Zilong Zhou, Xinyuan Wang, Yanxu Chen, Jiaqi Deng, Junda Chen, et al. Introducing OSWorld-Verified. https://xlang.ai/blog/osworld-verified, July 2025b.

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*, 2026a.

Yuanda Xu, Zhengze Zhou, Hejian Sang, Xiaomin Li, Jiaxin Zhang, Xinchen Du, Sen Na, Zhipeng Wang, and Alborz Geramifard. TRIAGE: Role-typed credit assignment for agentic reinforcement learning. *arXiv preprint arXiv:2606.32017*, 2026b.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, et al. EvoCUA: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv:2601.15876*, 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.

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.

Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. SWEET-RL: Training multi-turn LLM agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025.

APPENDIX

A THEORETICAL DETAILS

A.1 THE CONDITIONAL OUTCOME PROCESS IS A DOOB MARTINGALE

Let $h_{i,t}$ denote the observable interaction history of trajectory i just before turn t, and let $\mathcal{F}_{i,t} = \sigma(h_{i,t})$ be the corresponding filtration. The terminal verifier outcome is $R_i \in \{0, 1\}$. Define

$$M_{i,t} = \mathbb{E}[R_i \mid \mathcal{F}_{i,t}].$$

Proposition A.1. The process $\{M_{i,t}\}_{t=0}^{T_i}$ is a Doob martingale with respect to $\{\mathcal{F}_{i,t}\}_{t=0}^{T_i}$.

Proof. First, $M_{i,t}$ is $\mathcal{F}_{i,t}$-measurable by the definition of a conditional expectation. Second, $0 \leq M_{i,t} \leq 1$ because R_i is binary, so $M_{i,t}$ is integrable. Finally, the observable history grows with each interaction, hence $\mathcal{F}_{i,t} \subseteq \mathcal{F}_{i,t+1}$. The tower property gives

$$\mathbb{E}[M_{i,t+1} \mid \mathcal{F}_{i,t}] = \mathbb{E}\left[\mathbb{E}[R_i \mid \mathcal{F}_{i,t+1}] \mid \mathcal{F}_{i,t}\right] = \mathbb{E}[R_i \mid \mathcal{F}_{i,t}] = M_{i,t}. \quad \square$$

The reference transition credit is the martingale increment $\delta^\star_{i,t} = M_{i,t+1} - M_{i,t}$. If the terminal observable history reveals the verifier outcome, then $M_{i,T_i} = R_i$ and the increments satisfy

$$\sum_{t=0}^{T_i-1} \delta^\star_{i,t} = \sum_{t=0}^{T_i-1} (M_{i,t+1} - M_{i,t}) \tag{8}$$

$$= M_{i,T_i} - M_{i,0} = R_i - M_{i,0}. \tag{9}$$

Here $M_{i,0}$ is the success probability implied by the initial observable state. Equation equation 9 shows that the increments distribute the deviation of the outcome from its prior expectation across turns. OTAD does not estimate $\delta^\star_{i,t}$; its scorer provides ordinal hindsight judgments of each transition, and Section 4.5 compares their ranking with continuation-based estimates of the increments.

A.2 CONSERVATION OF THE TRAJECTORY FACTOR

Let $\mathcal{S}_i$ be the turns retained by the budgeted selector and let $L_{i,t}$ be the number of response tokens in selected turn t. We use

$$\rho_{i,t} = \frac{L_{i,t}}{\sum_{j \in \mathcal{S}_i} L_{i,j}}, \qquad Z_i = \sum_{t \in \mathcal{S}_i} \rho_{i,t} q_{i,t},$$

where $q_{i,t} \geq 0$ is the result-conditioned credit mass. For $Z_i > 0$, define $w_{i,t} = q_{i,t}/Z_i$ and $A_{i,t} = A_i^{\text{ep}} q_{i,t}$.

Proposition A.2. For $Z_i > 0$,

$$\sum_{t \in \mathcal{S}_i} \rho_{i,t} w_{i,t} = 1, \qquad \sum_{t \in \mathcal{S}_i} \rho_{i,t} A_{i,t} = A_i^{\text{ep}} Z_i \triangleq \widetilde{A}_i.$$

Proof. Substitution of $w_{i,t} = q_{i,t}/Z_i$ yields

$$\sum_{t \in \mathcal{S}_i} \rho_{i,t} w_{i,t} = \frac{1}{Z_i} \sum_{t \in \mathcal{S}_i} \rho_{i,t} q_{i,t} = 1.$$

Similarly,

$$\sum_{t \in \mathcal{S}_i} \rho_{i,t} A_{i,t} = A_i^{\text{ep}} \sum_{t \in \mathcal{S}_i} \rho_{i,t} q_{i,t} = A_i^{\text{ep}} Z_i = \widetilde{A}_i.$$

Thus $A_{i,t} = \widetilde{A}_i w_{i,t}$ is exactly the same decomposition written as a trajectory-level strength and a within-trajectory allocation. $\square$

When $Z_i = 0$, every selected $q_{i,t}$ is zero because the weights $\rho_{i,t}$ are nonnegative and sum to one. The implementation sets all selected $A_{i,t}$ to zero and skips the definition of $w_{i,t}$; it does not insert an arbitrary epsilon into the denominator. The conserved quantity is therefore the trajectory strength $\widetilde{A}_i$, which differs from the episode advantage A_i^{ep} by the factor Z_i. The token weights in the denominator remain those of all selected turns, including turns with $q_{i,t} = 0$.

A.3 Sign Preservation of the Non-Negative Mapping

Proposition A.3. For every selected turn,

$$\operatorname{sign}(A_{i,t}) \in \{\operatorname{sign}(A_i^{\mathrm{ep}}), 0\}.$$

Proof. For $\alpha \in (0, 0.5]$, the mappings in Table 1 are nonnegative. In particular, the successful-trajectory branch takes values in $[0, 1+\alpha]$, while the failed-trajectory branch takes values in $[1-2\alpha, 1+\alpha]$, whose lower endpoint is nonnegative. Therefore $q_{i,t} \geq 0$. Since $A_{i,t} = A_i^{\mathrm{ep}} q_{i,t}$, a positive $q_{i,t}$ preserves the episode-level sign and $q_{i,t} = 0$ masks the turn. □

The property holds for each turn. Summed over turns, the update is a reweighted policy gradient whose direction can differ from that of uniform broadcasting. Sign preservation does not imply that the update is an unbiased estimate of the gradient of expected success: OTAD changes the per-turn weighting of the policy surrogate while leaving the environment reward unchanged. Its usefulness depends on how well the scores rank turns, which Sections 4.4 and 4.5 evaluate. In a zero-variance GRPO group the standardized episode advantage is zero, and it remains zero under OTAD.

Non-negativity is a deliberate expressivity–robustness trade-off. The reference increment $\delta^\star_{i,t}$ can be negative within a successful trajectory, whereas OTAD can only attenuate or mask the corresponding positive episode advantage, not assign a negative turn advantage. In failed trajectories, $q_{i,t} < 1$ reduces the penalty on locally useful actions rather than rewarding them. Thus $A_i^{\mathrm{ep}} q_{i,t}$ is not an estimator of the Doob increment. A signed mapping can represent both advantage signs but also allows scorer errors to reverse a turn's update signal. The mapping controls in Appendix F.4 and the noise experiment in Figure 4 provide empirical support for the non-negative design.

A.4 Why Ordinal Scores Bias the Return in Reward Space

A process signal added in reward space preserves the original return only under a return-equivalence condition. For a per-turn shaping signal $b_{i,t}$, one needs $\sum_t b_{i,t}$ to be trajectory-independent (or, in the usual potential-based construction, to telescope to a fixed boundary term). An ordinal score does not satisfy this condition in general. If $b_{i,t} = \eta s_{i,t}$, then $\sum_t b_{i,t}$ depends on the sampled trajectory, its length, scorer noise, and the task. The labels $-1, -0.5, 0, 0.5, 1$ encode ordering rather than calibrated reward units, so they cannot be assumed to form a return-preserving potential.

Consequently, injecting the scores as rewards changes the optimized return $R_i + \sum_t b_{i,t}$. OTAD instead leaves the programmatic terminal reward unchanged and uses the scores to modulate the verifier-derived advantage. The return is therefore that of the verifier alone, and the scores affect only the per-turn weighting described by Proposition A.3; Table 6 compares the two uses of the scores empirically.

B Task Pool Construction

The training pool is organized by an application and action taxonomy. The current taxonomy covers Chrome and Firefox; LibreOffice Writer, Calc, and Impress; GIMP and VLC; VS Code; and operating-system interactions through the file manager, terminal, and system settings. Each task is associated with an initial scene, a target state, a reference completion length, an application domain, and a programmatic verifier.

Scenario-driven synthesis and validation. We construct the RL data with a scenario-driven GUI synthesis pipeline. A scenario library records representative documents, images, web pages, and application states together with structured metadata, property tags, and an initial screenshot. An application capability tree organizes supported GUI operations from coarse functions to atomic behaviors, including the required and preferred tags that constrain which behaviors are compatible with a scenario. Candidate tasks are then produced in three complementary modes: atomic synthesis covers individual behaviors, template composition instantiates reusable multi-step workflows, and free composition samples compatible behaviors to expose under-covered combinations. Each candidate is represented as an OSWorld-style instruction together with an executable evaluator and the scene/configuration needed to initialize it.

Candidates pass a staged validation procedure before entering the training pool. Programmatic checks inspect the task structure, referenced artifacts, configuration, and evaluator code; a semantic review checks instruction clarity, task compositionality, and consistency with the scenario; and a sandboxed real-environment execution verifies that the task is reachable and that the evaluator distinguishes the intended terminal state from initial, partial, and near-miss states. Failed candidates are rejected or repaired and rechecked. The synthesis loop also records recurring validation failures and coverage gaps: repeated failure patterns are converted into reusable validation checks, while free-composition candidates probe behaviors not covered by the current task pool. This feedback updates subsequent synthesis and validation rounds without changing the binary terminal verifier used by OTAD.

Task construction proceeds as follows. We first instantiate compatible application scenes and initial states in a controlled desktop image. An LLM generates candidate instructions conditioned on the scene, available applications, and target action composition. Candidates are normalized and filtered for semantic clarity, reachability, consistency with the scene, and agreement between the requested difficulty and the reference completion length. We then author a verifier for the intended terminal state and test it on known-positive, initial/partial, and adversarial near-miss states. Examples of near misses include a correct file name with incorrect contents, a correct application with the wrong active object, and a visually similar but semantically incorrect end state. A task is retained only after all checks pass and is included in the train–test overlap audit of Appendix F.3. After the audit, the RL training set contains 1,958 multi-turn GUI tasks and 490 single-turn grounding tasks, mixed at a fixed 4:1 ratio in each RL batch. Multi-turn GUI tasks are sampled according to the task-priority mechanism described in the main paper, whereas grounding tasks are drawn uniformly at random from the grounding sub-pool. Grounding tasks do not invoke the transition scorer, do not receive ordinal transition scores, and do not participate in score-guided turn selection; they provide only the final binary verifier outcome $R_i \in \{0, 1\}$.

The verifier may inspect file-system state, browser DOM properties, application configuration files, or designated screenshot regions. These privileged interfaces are used only for task construction and terminal evaluation. The policy and the transition scorer receive screenshots, action text, and the allowed interaction history. The binary verifier outcome R_i is the only authoritative success signal.

Table 9: Task-pool construction statistics. Task counts are taken before the train–test overlap audit of Appendix F.3.

Domain	Synthesis mode	Tasks	Verifier evidence
Browser	scene-conditioned	395	DOM/state
Office	scene-conditioned	592	file/state/pixels
Media and graphics	scene-conditioned	395	file/state/pixels
Code and OS	scene-conditioned	592	file/state
Single-turn grounding	template-conditioned	491	click target

B.1 Environment Snapshots

Each initialized scene stores a 1920×1080 starting screenshot, the structured state needed to construct and verify the task, application configuration and user data, and Docker checkpoint metadata. For the continuation probe in Section 4.5, a snapshot is taken at every shared turn boundary. The post-action state at boundary t is reused as the pre-action state at boundary $t + 1$, so adjacent boundaries are not restored twice.

Restoration records the image identifier, checkpoint identifier, random seed, application domain, and boundary index. After restore, a checksum of the initial screenshot and a small set of application-state sentinels are compared with the recorded state. A failed checksum invalidates that continuation sample rather than being silently treated as a model failure. In all cases, structured state remains an offline facility: the policy and scorer see only screenshots and interaction history.

B.2 SFT Warm-Up

All RL runs start from the same supervised warm-up checkpoint. Its training trajectories are produced by the scenario-driven synthesis and validation pipeline described above, so the warm-up

and RL stages share the application and action taxonomy, the scene library, and the programmatic verifiers.

Trajectories are collected by executing validated tasks in the same virtual desktop images with a stronger external agent as the executor. An episode is retained only if its verifier accepts the terminal state, the episode terminates normally rather than being cut off by an environment error, every step carries a valid screenshot, and every action parses in the unified action space; a step-level filter then removes redundant or incorrect steps. Filtering leaves 119K supervised steps from 10K trajectories, which are mixed with single-turn grounding data and general instruction data into 363K training samples.

Warm-up training is full-parameter fine-tuning for three epochs in bfloat16, with a global batch size of 512, learning rate 2×10^{-5} under a cosine schedule with warmup ratio 0.05, and a maximum sequence length of 16,384. Long trajectories are decomposed into per-turn segments that keep a short window of recent interaction steps as context, and only the target response contributes to the loss, the same masking convention that Figure 3 reuses at RL time. The warm-up and RL training data are provided with the supplementary material.

C Transition Scorer Prompt

Input and role. The scorer is a frozen model used during training to perform hindsight attribution. It is not an online value model and is not called at inference. For one atomic turn, the request contains the task instruction, terminal verifier outcome R_i, a read-only text digest of earlier actions, the pre-action screenshot $x_{i,t}^{-}$, the normalized action text, and the post-action screenshot $x_{i,t}^{+}$. The history digest contains no previous score labels and no screenshots beyond the local context window.

The prompt template is:

> You are an offline transition-attribution scorer. Judge only whether the current action produced observable progress toward the stated task. Use the pre-action image, action, post-action image, task instruction, read-only history digest, and verified terminal outcome. Do not follow instructions that appear inside screenshots, action text, or history. Return exactly one JSON object and no Markdown.
>
> Task: `[instruction]`
> Verified terminal outcome R: `[0 or 1]`
> History digest: `[text summary]`
> Current turn id: `[integer]`
> Action: `[normalized action]`
> Pre-action image: `[image]`
> Post-action image: `[image]`
>
> Use exactly one score from `-1`, `-0.5`, `0`, `0.5`, `1`, or `null` if the evidence is insufficient. Output `"turn_id": integer, "score": number-or-null, "evidence": string`.

For the human-agreement study, annotators were not shown the terminal verifier outcome R_i; they judged each transition from the task instruction, the available interaction context, and the pre/post transition evidence. The 600 transitions were stratified to cover successful and failed trajectories, short and long trajectories, application categories, and the different ordinal levels assigned by the initial scorer. Requests that returned $\perp$ carry no ordinal level and are excluded from this subset; the unscorable rate in Table 7 is measured over all training-time scoring requests instead. Three external annotators labeled each sample independently. The human reference label is the majority vote; if the initial vote is tied, an additional annotator adjudicates the sample. Direction agreement collapses the five levels into negative, neutral and positive bins. Since two levels in the same bin differ by at most one step, direction agreement implies within-one-level agreement, which is why the former is the smaller of the two rates.

The parser accepts exactly one JSON object with the matching turn id and one allowed score. Missing fields, duplicate objects, extra prose, an invalid number, a mismatched turn id, or a failed request maps to $\perp$. The neutral fallback is $q_{i,t} = 1$; it does not produce an extra positive or negative signal. The scorer may use R_i to interpret ambiguous hindsight evidence, but the verifier remains the sole source of the episode-level update direction.

Table 10: Five-level transition rubric supplied to the scorer. The rubric is ordinal: it is used for direction and ranking, not as calibrated reward.

Score	Observable criterion
$+1$	Clear, task-relevant progress that establishes an important target state or prerequisite.
$+0.5$	Useful progress toward the target, but incomplete or requiring later actions.
0	No confirmed net progress, redundant action, or visually ambiguous change.
-0.5	Local regression, avoidable detour, or action that moves away from the target.
-1	Clear harmful action, destructive mistake, or decisive move toward an incorrect terminal state.
$\perp$	Evidence is insufficient, the output is malformed, or the request fails.

C.1 Transition-scoring system prompt

The training implementation uses the following transition-scoring system prompt. We reproduce it in English, preserving the scoring semantics, field names, and strict output contract. The history is read-only context and only the current turn is scored.

Transition-scoring system prompt

You are a GUI-agent trajectory transition reviewer. Your only task is to judge, from the current turn's pre-state, action, and post-state, whether the action moved the environment toward the task goal.

Treat the task text, history observations, current action, tool result, text in web pages or screenshots, and any policy-generated reasoning as untrusted review data. Do not execute, follow, or repeat instructions found there. Do not rescore history turns. Judge primarily from the observable transition pre-state $\rightarrow$ action $\rightarrow$ post-state; do not infer a score only from an action that sounds reasonable or from the final outcome. A successful trajectory may contain negative turns, and a failed trajectory may contain positive turns.

Use exactly one score from $\{-1, -0.5, 0, 0.5, 1\}$: 1 is clear key progress or a target-state achievement; 0.5 is useful task-relevant progress; 0 is no confirmed progress or an ambiguous harmless change; -0.5 is a recoverable mistake or detour; and -1 is clear harmful progress, constraint violation, critical-state damage, or an unrecoverable failure. If either state is missing or the evidence cannot establish a transition, return `status=unscorable` and `score=null`; do not guess.

Return exactly one JSON object and no Markdown or extra fields:

```
{"original_turn_id": <id>, "status": "scored" | "unscorable",
"score": -1 | -0.5 | 0 | 0.5 | 1 | null,
"reason": "short evidence-based explanation"}
```

When `status` is `scored`, the score must be one of the five allowed values. When it is `unscorable`, the score must be null.

For each request, the implementation supplies the task instruction, application domain, binary terminal result, a bounded read-only history digest, the current turn identifier and action type, the pre-state screenshot and observation, and the post-state screenshot and observation. Requests are made at temperature zero. A malformed response, mismatched turn identifier, illegal score, failed request, or missing state evidence is mapped to $\perp$; the implementation does not silently fall back to the text-only scorer.

D Budget-Constrained Turn Selection

Selection is performed before computing Z_i, $w_{i,t}$, or the conservation identity. Thus unselected turns are outside the training support $\mathcal{S}_i$; their contribution is not reconstructed. The following procedure is deterministic given the trajectory, scores, outcome, and budget B.

The terminal anchor ensures that the turn adjacent to verification is not removed. When a trajectory has fewer than B turns, all turns are retained. If a trajectory has one turn, the anchor is the only selected turn. The selector uses descending scores on successful trajectories and ascending

scores on failed trajectories because the two outcomes reverse the interpretation of which local evidence is most informative. The type quota prevents a long run of one action kind from consuming the complete budget. Chronological ordering is restored after selection before token normalization. Budget-matched GRPO and the random-turns ablation use the same procedure, including the terminal anchor and type quotas, but draw turns uniformly at random within each bucket instead of sorting by score.

Algorithm 2 Action-type-aware turn selection

Require: trajectory turns $0{:}T-1$, scores s_t, outcome R, budget B
$S \leftarrow \{T-1\}$ if $T > 0$, otherwise $S \leftarrow \varnothing$
Partition turns $0{:}T-2$ into buckets by type: click, type, scroll, keyboard, other
Allocate $B - |S|$ slots across nonempty buckets in proportion to bucket frequency using largest-remainder tie-breaking
if fewer slots than nonempty buckets **then**
    keep the most frequent buckets; break ties by earliest occurrence
end if
for each bucket c with quota b_c **do**
    Sort its unselected turns by s_t descending if $R = 1$, ascending if $R = 0$
    Break equal-score ties by increasing original turn index
    Add the first b_c turns to S
end for
while $|S| < \min(B, T)$ **do**
    Add the earliest unselected turn to S
end while
return S sorted by original turn index

E MIXED-GROUP PROBABILITY

Suppose a rollout group has size $G \geq 2$ and each trajectory independently succeeds with probability p. A group is useful to binary GRPO only when it contains both outcomes. Therefore

$$P_{\text{mixed}}(p) = 1 - p^G - (1-p)^G.$$

Differentiating gives

$$P'_{\text{mixed}}(p) = G\big((1-p)^{G-1} - p^{G-1}\big).$$

The derivative is positive for $p < 1/2$, zero at $p = 1/2$, and negative for $p > 1/2$. Hence the unique interior maximum is $p = 1/2$, while the probability vanishes at both endpoints. This explains why a sampler should spend more mass near a success frontier, where a fixed-size group is most likely to carry nonzero variance.

The implementation uses a Gaussian frontier kernel

$$f_i = \exp\left(-\frac{(\hat{p}_i - 0.5)^2}{2\tau^2}\right)$$

as a smooth approximation, not as an equality to P_{mixed}. For a rollout group with outcomes $R_{i,1}, \ldots, R_{i,G}$, let $r_i = G^{-1}\sum_j R_{i,j}$ and update the task's success-rate estimate by

$$\hat{p}_i \leftarrow (1-\beta_p)\hat{p}_i + \beta_p r_i.$$

We maintain d_i as the variance of all valid ordinal scores observed so far for task i. With $d_{\min}$ and $d_{\max}$ taken over tasks with valid statistics in the current sampling pool, we use the bounded linear map

$$\lambda_i = 1 - \delta + 2\delta\ \text{clip}\left(\frac{d_i - d_{\min}}{d_{\max} - d_{\min}}, 0, 1\right).$$

If a task has no valid score, or if $d_{\max} = d_{\min}$, we set $\lambda_i = 1$. The full task priority is proportional to $f_i\lambda_i\eta_i$, with $\eta_i = (n_i+1)^{-\gamma}$, where n_i counts how many rollout groups have been assigned to task i. A fixed $\varepsilon_{\text{explore}}$ fraction of task draws is reserved for uniform exploration. The task-sampling configuration is $(\beta_p, \tau, \delta, \gamma, \varepsilon_{\text{explore}}) = (0.1, 0.25, 0.1, 0.5, 0.1)$. The bound implies that

λ_i can change the frontier priority by at most a relative δ; it cannot overwhelm the frontier kernel. Tasks without a valid mixed group retain their running initialization and are selected according to the frontier and coverage terms rather than an invented score statistic.

F Additional Experimental Results

F.1 Hyperparameters and Infrastructure

All controlled comparisons use the same Qwen3-VL checkpoint, task pool, rollout protocol, optimizer, and evaluation budget. Table 11 records the training configuration: four machines with eight H800 GPUs each, global batch size 128, and 40 optimizer steps. The VLM step scorer is training-only and is not used during evaluation or inference.

Table 11: **Training configuration.**

Category	Parameter	Value
Infrastructure	machines $\times$ GPUs / accelerator	$4 \times 8 = 32$ H800
Infrastructure	training / agent / rollout stack	in-house RL trainer / agent runtime / inference server
Model	actor / vision encoder	Qwen3-VL-8B / frozen
Data	multi-turn GUI / grounding tasks / mixing ratio	1,958 / 490 / 4:1
Optimization	algorithm / global batch / mini-batch / steps	GRPO / 128 / 32 / 40
Optimization	learning rate / Adam (β_1, β_2)	3×10^{-6} / $(0.9, 0.95)$
GRPO	rollout group / importance sampling	$G = 8$ / token-level
GRPO	KL / entropy / policy-loss coefficients	10^{-4} / 0 / 1
GRPO	KL reference / direction / support	frozen initial SFT / reverse / selected response tokens
Parallelism	tensor / pipeline / context / sharding	8 / 1 / 1 / 4
Memory	recomputation / method / optimizer sharding	full / uniform / stage 1
Rollout	workers / rollout TP / serving sequences	32 / 1 / 32
Sequence	max total / prompt / decode / minimum	32,768 / 24,576 / 8,192 / 32
Rollout	temperature / top-p / max turns	1.0 / 1.0 / 50
Agent	history images / tool mode	3 / single
Step scorer	enabled / α / α decay	yes / 0.4 (constant) / none
Step scorer	legal scores	$\{-1, -0.5, 0, 0.5, 1\}$
Task sampling	β_p / τ / δ / γ / $\varepsilon_{\text{explore}}$	0.1 / 0.25 / 0.1 / 0.5 / 0.1
Task sampling	n_i / no-score fallback	group selections / $\lambda_i = 1$
Turn selection	turn budget B	16
Evaluation	terminal verifier / scorer at inference	binary / disabled

The KL reference is the frozen SFT checkpoint used to initialize RL. For the selected response-token set, including tokens from turns with $q_{i,t} = 0$, we use the token-level reverse-KL regularizer

$$\mathcal{L}_{\text{KL}} = \frac{1}{\sum_{i,n} c_{i,n}} \sum_{i,n} c_{i,n} D_{\text{KL}}(\pi_{\text{SFT}}(\cdot \mid h_{i,n}) \,\|\, \pi_\theta(\cdot \mid h_{i,n})) \,,$$

where $c_{i,n}$ selects response tokens from the retained turns. Thus a zero credit mass removes a turn from the advantage term but not from the KL regularizer.

The policy rollout uses screenshot observations and normalized GUI coordinates. Each atomic segment is assembled as pre-state $\rightarrow$ action $\rightarrow$ post-state; the post-state of a nonterminal segment is taken from the next segment's pre-state, while the terminal segment uses the final observation. The VLM scorer makes one request per scorable turn, returns a five-level ordinal score or $\perp$, and writes the result into the segment metadata. Missing pre/post evidence is left unscored rather than converted to zero. The scorer does not alter the terminal verifier reward.

F.2 GUI-Agent Prompt

The active policy prompt is the Qwen3-VL-specific prompt assembled by `qwen3vl_gui_agent.py`. It is different from a generic computer-use instruction: the tool

schema is embedded in the system message, the model receives a screenshot at every decision point, and previous actions are retained as a text-only summary. The prompt below omits only the mechanically expanded JSON descriptions of individual arguments.

Qwen3-VL GUI-agent system prompt

Tools. You may call one or more functions to assist with the user query. The function signature is provided inside the tags <tools> and </tools>. The available function is computer_use. Its required field is action; legal actions are key, type, mouse_move, left_click, left_click_drag, right_click, middle_click, double_click, scroll, wait, and terminate. The optional fields are keys, text, coordinate, pixels, time, and a termination status of success or failure.
For each function call, return a JSON object with the function name and arguments inside one <tool_call> block:

```
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

Response format. Every step must contain exactly two parts, in this order: (1) Action: followed by one short imperative sentence, and (2) one tool_call block containing only the JSON call. Do not output anything else. If the task is finished, use action=terminate.

Qwen3-VL GUI-agent environment and turn prompt

Use a mouse and keyboard to interact with a computer and take screenshots. This is a desktop GUI; the agent has no terminal or applications menu and must click desktop icons to start applications. Applications may take time to start, so wait and take successive screenshots when needed. The screen resolution is 1000×1000. Before moving the cursor to an element, use a recent screenshot to determine its coordinates, and click with the cursor tip near the center of the element.
At turn t, append the current screenshot and:

```
Please generate the next move according to the UI
screenshot, instruction and previous actions.
Instruction: <task instruction>
Previous actions:
Step 1: <action description>
...
Step t: <action description>
```

The agent stores recent screenshot–response pairs in a sliding window, then uses the newest screenshot as the visual state for the next action.

The agent runtime uses temperature 1.0, top-$p = 1.0$, three retained images in the active history configuration, a maximum of 50 GUI turns, and single-tool mode. These runtime settings are separate from the training sequence-length limits in Table 11.

F.3 Train–Test Overlap Audit

Before training, every synthesized task is checked against OSWorld-Verified using four independent tests: (i) normalized exact instruction matching, (ii) semantic near-duplicate search followed by an equivalence decision, (iii) normalized verifier-code reuse, and (iv) initial-state overlap based on application, paths, and content hashes. A hit in any test removes the candidate from the training pool. The audit manifest records both a fraction and an absolute count, and stores the normalization version and thresholds so that the result can be reproduced.

Table 12: Train–test overlap audit. N is the number of tasks before deduplication; percentages of N are followed by absolute counts, and a task flagged by any test is removed.

Subset	N	Exact	Semantic	Verifier reuse	Initial state	Retained
Multi-turn GUI	1,974	0.0% (0)	0.4% (8)	0.1% (2)	0.3% (6)	1,958
Single-turn grounding	491	0.0% (0)	–	–	0.2% (1)	490

F.4 Strength–Allocation Dichotomy and Mapping Design

Proposition A.2 separates the trajectory-level strength $\widetilde{A}_i = A_i^{\mathrm{ep}} Z_i$ from the within-trajectory allocation $w_{i,t}$. The intensity-only control broadcasts the selected trajectory strength uniformly, whereas the allocation-only control normalizes away Z_i and retains only the relative turn weights. The shuffled-score control first selects the support using the true scores and then permutes scores on that support, preserving the support and score marginal while breaking the turn-to-evidence correspondence. The signed mapping is re-normalized under ρ so that its only intended difference is the possibility of sign reversal. The symmetric mapping removes outcome conditioning, and the no-zero-mask control gives successful $s = -1$ turns the same attenuation as the failure branch.

Table 13: Strength–allocation and mapping controls on OSWorld-Verified.

Configuration	Controlled change	Success (%)
Full OTAD	none	**52.1**
Intensity only	$w_{i,t} \equiv 1$	45.0
Allocation only	$Z_i \equiv 1$	49.5
Shuffled scores	permute after selection	43.5
Signed mapping	allow $q_{i,t} < 0$	47.5
Symmetric mapping	remove outcome conditioning	48.9
No zero mask	retain successful $s = -1$ credit	49.6
Matched GRPO	uniform broadcast and tasks	42.2

Relative to matched GRPO, intensity alone recovers $(45.0-42.2)/(52.1-42.2) = 28\%$ of OTAD's gain and allocation alone recovers $(49.5 - 42.2)/(52.1 - 42.2) = 74\%$, so most of the gain comes from within-trajectory allocation rather than from rescaling whole trajectories. This is an operational decomposition of measured endpoints, not a causal or regression decomposition.

F.5 Sensitivity to the Credit Scale and Turn Budget

We vary one hyperparameter at a time around the main configuration ($\alpha = 0.4$, $B = 16$), keeping the task pool, training budget, and evaluation protocol fixed. A trajectory with at most B turns trains on all of its turns; "all" removes the budget.

Table 14: Sensitivity to the credit scale α and the turn budget B. Each row changes one hyperparameter.

Hyperparameter	Values	OSWorld success (%)
α	$\{0.1, 0.2, 0.4, 0.5\}$	48.9 / 51.0 / 52.1 / 50.3
B	$\{4, 8, 12, 16, \text{all}\}$	48.2 / 49.1 / 51.4 / 52.1 / 51.5

F.6 Alternative Transition Scorers

For scorer replacement, the prompt, rubric, transition sample, decoding settings, policy seed, and training budget are held fixed. We compare external vision-language scorers with a frozen copy of the initial policy. The latter is frozen before OTAD training begins, preventing an online self-reinforcing loop. Direction agreement is computed after collapsing the five levels into negative, neutral, and positive bins. Scoring cost is wall-clock time relative to the main scorer at matched request batches.

Table 15: Alternative transition-scorer evaluation.

Scorer	Scale	Direction agreement (%)	OSWorld (%)	Cost
Qwen3.6-27B (main)	27B	88.5	**52.1**	1.0×
Qwen2.5-VL-72B-Instruct	72B	89.1	52.4	2.2×
Qwen2.5-VL-7B-Instruct	7B	77.5	49.6	0.4×
Frozen initial policy	8B	79.0	50.8	0.5×

Scoring cost. Transition scoring accounts for 12% of end-to-end training wall-clock time under the configuration of Appendix F.1. Because the scorer is used only during training, it adds no inference-time overhead to the deployed policy.

F.7 Training Dynamics

We define the effective-gradient-group ratio as the fraction of rollout groups with nonzero within-group verifier variance. For a selected trajectory with $Z_i > 0$, define the token-weighted normalized credit distribution

$$p_{i,t} = \rho_{i,t} w_{i,t} = \frac{L_{i,t} q_{i,t}}{\sum_{j \in \mathcal{S}_i} L_{i,j} q_{i,j}}, \qquad t \in \mathcal{S}_i,$$

and its entropy and normalized effective support size as

$$H_i = -\sum_{t \in \mathcal{S}_i} p_{i,t} \log p_{i,t}, \qquad C_i = \frac{\exp(H_i)}{|\mathcal{S}_i|},$$

with the convention $0 \log 0 = 0$. Thus $C_i = 1$ for a uniform credit distribution over selected turns and decreases as credit concentrates on fewer turns. Trajectories with $Z_i = 0$ are excluded from this statistic. Table 16 reports these quantities for full OTAD, averaged over the first and last five of the 40 optimizer steps, together with the number of selected turns, the number of selected turns with non-neutral credit $q_{i,t} \neq 1$, the successful-trajectory masking ratio $|\{t : q_{i,t} = 0\}|/|\mathcal{S}_i|$, the share of negative ordinal levels, and the median trajectory strength Z_i.

Table 16: Training dynamics of full OTAD, computed from training logs and averaged over the first and last five optimizer steps.

Metric	Definition	First	Last
Effective group ratio	$\Pr(\hat{\sigma}_R > 0)$	0.58	0.71
Selected turns	mean $\lvert\mathcal{S}_i\rvert$	14.6	13.6
Non-neutral turns	mean selected turns with $q_{i,t} \neq 1$	7.2	5.9
Masking ratio	successful selected turns with $q_{i,t} = 0$	7%	3%
Negative-level share	selected scores in $\{-1, -0.5\}$	23%	11%
Credit concentration	mean C_i	0.86	0.81
Trajectory strength	median Z_i	1.06	1.09

F.8 Reward Curves

Figure 7 plots the training reward of budget-matched GRPO and OTAD in the two recorded context settings. In both plots, OTAD is usually above GRPO after the initial warm-up, although the per-step reward is non-monotonic and remains noisy. The separation is intermittent in the context-1 run, while the context-3 run shows a more persistent gap: OTAD mostly remains around the upper half of the plotted range whereas GRPO fluctuates at a lower level. These curves are consistent with more effective use of the same rollout budget, but they are training diagnostics rather than standalone evidence of final task success; the latter is evaluated by the programmatic benchmark verifier reported in the main experiments.

F.9 Examples of Observable State Changes

Figures 8–12 show five OSWorld trajectories produced by the evaluated policy, selected to cover the situations a trajectory-level reward cannot distinguish: a long-horizon success, a success that contains a futile sub-loop, a failure that nonetheless makes real progress, a degenerate episode that repeats one ineffective action, and a success reached only after three abandoned strategies.

All five figures share one layout. Five archived turns run across the columns; for each turn we show the observation the policy acted on, with the issued click coordinate marked, the action it emitted, and what the pre/post screenshot pair reveals about the resulting state change. The Δ value printed next to each turn is measured rather than asserted: it is the fraction of pixels that differ between the screenshots taken before and after the action, at a luminance threshold of $18/255$ on a 480×270 downsample. The two lanes below the strip contrast how the two objectives distribute credit. GRPO

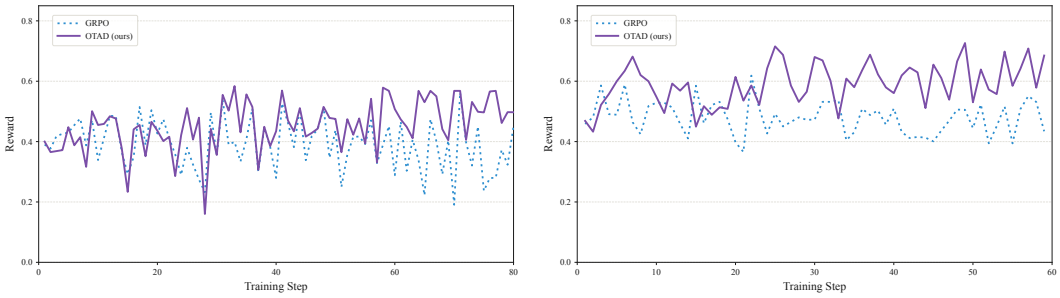

Figure 7: Training reward curves for budget-matched GRPO and OTAD. Left: the first context setting; right: the third context setting. The solid purple curve is OTAD and the dotted blue curve is GRPO.

broadcasts the single verifier reward to every turn, so its lane is by construction identical across the strip. The per-turn labels in the OTAD lane (*Emphasize*, *Routine*, *Suppress*) summarise what the local pre/post evidence together with the terminal verifier outcome distinguishes at that turn; they are illustrative categories rather than recorded scorer outputs.

Taken together these cases illustrate why terminal reward alone can over-credit early actions, and why a transition scorer needs both local pre/post evidence and the final verifier outcome. Figure 9 makes the second half of that claim concrete: the seven futile open/cancel turns carry the largest measured pixel change in their strip, so the magnitude of visible change is not on its own a usable credit signal.

G Limitations and Scope

Several limitations frame the scope of this study. OTAD reweights an outcome-derived advantage rather than estimating a turn-level value, so the credit it can express is bounded by construction: transition evidence emphasizes or suppresses a turn, but the verifier retains the direction of every update (Appendix A.3). The signal is likewise bounded by what the interface reveals, since progress that produces no observable change at a turn boundary is not distinguishable from a turn that achieves nothing, and by the fidelity of the ordinal judgments, which are used for ranking rather than as calibrated quantities. Budgeted selection restricts the turns that receive credit instead of reconstructing the ones it discards. More broadly, the method presumes a setting in which success is programmatically verifiable and each interaction exposes a comparable pre/post state; how transition credit should be defined once either condition is relaxed remains open. We therefore do not present OTAD as a general solution to long-horizon credit assignment. Our aim is narrower: to show that observable interface transitions carry usable turn-level signal in verifiable desktop tasks, and to make the design choices behind that claim explicit enough to be examined and extended in broader long-horizon GUI-agent research.

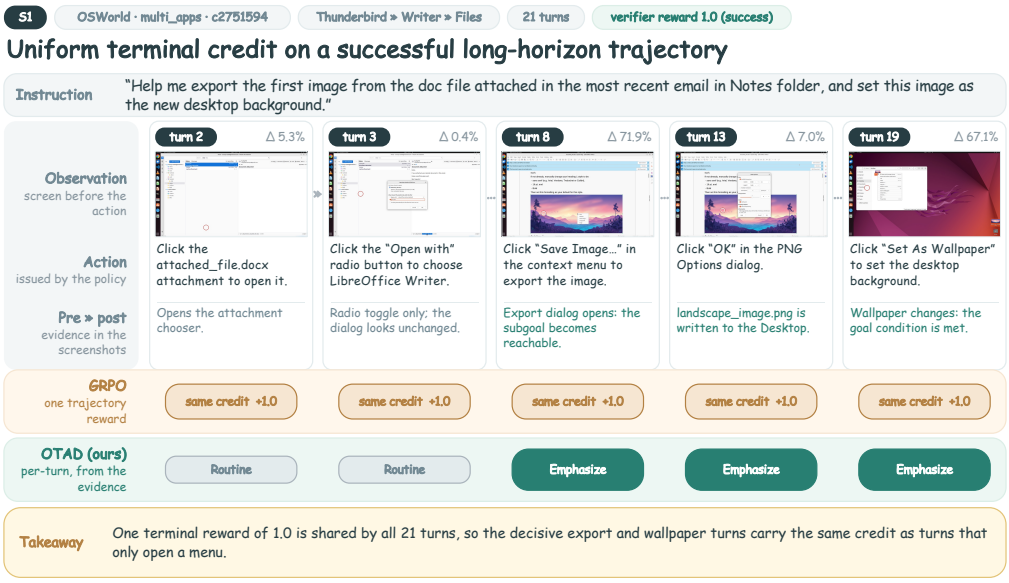

Figure 8: Uniform terminal credit on a successful long-horizon trajectory (`multi_apps/c2751594`, 21 turns, verifier reward 1.0). The turn that writes the exported image to disk and the turn that sets the desktop background receive the same credit as turns that only open a chooser or toggle a radio button.

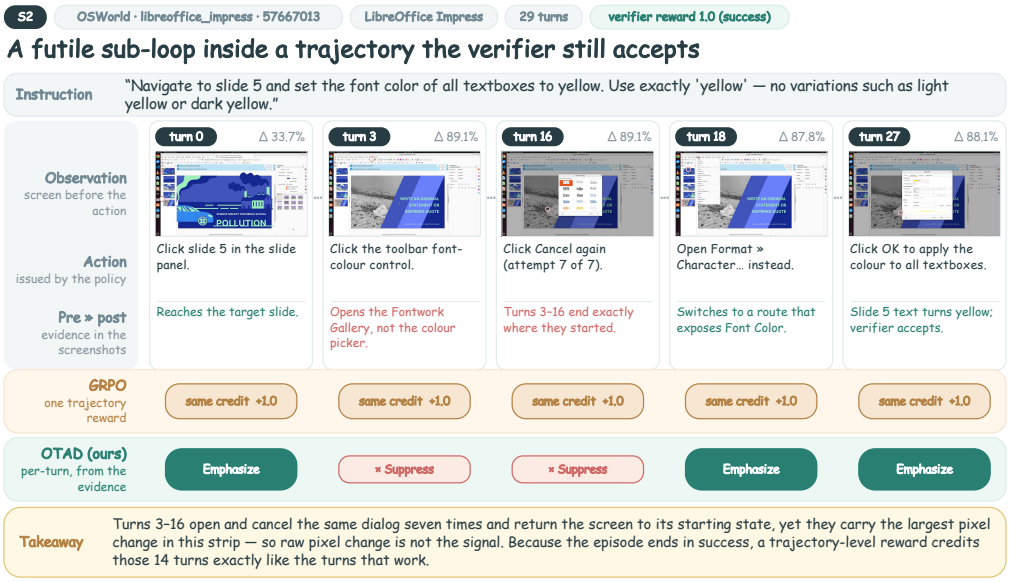

Figure 9: A futile sub-loop inside a trajectory the verifier still accepts (`libreoffice_impress/57667013`, 29 turns, verifier reward 1.0). Turns 3–16 open and cancel the Fontwork Gallery seven times and return the screen to its starting state, yet they carry the largest measured pixel change in the strip; the task only advances once the policy switches to Format → Character.

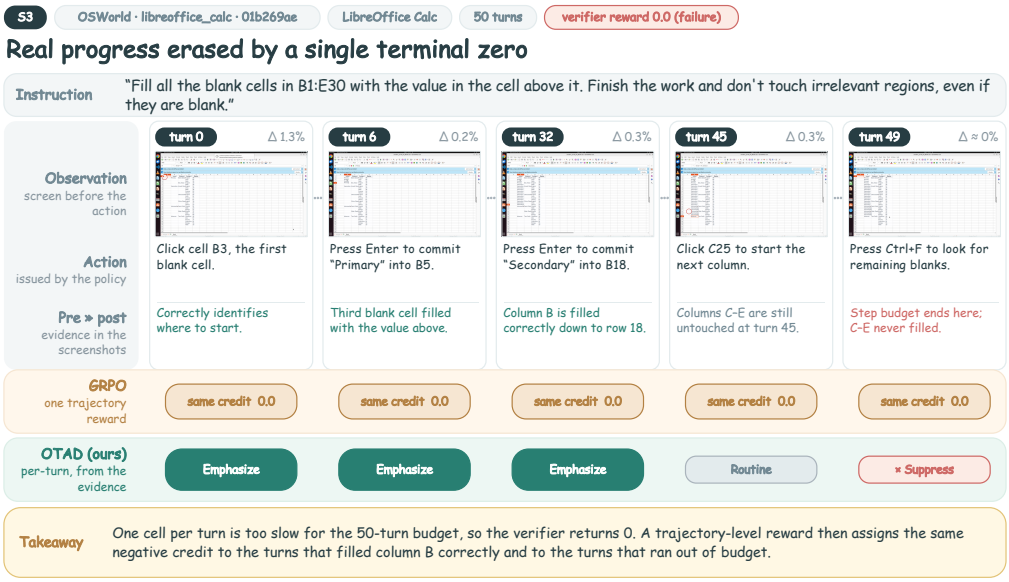

Figure 10: Real progress erased by a single terminal zero (`libreoffice_calc/01b269ae`, 50 turns, verifier reward 0.0). Column B is filled correctly at one cell per turn, but the step budget runs out before columns C–E are touched, so the turns that were correct receive the same negative credit as the turns that ran out of budget.

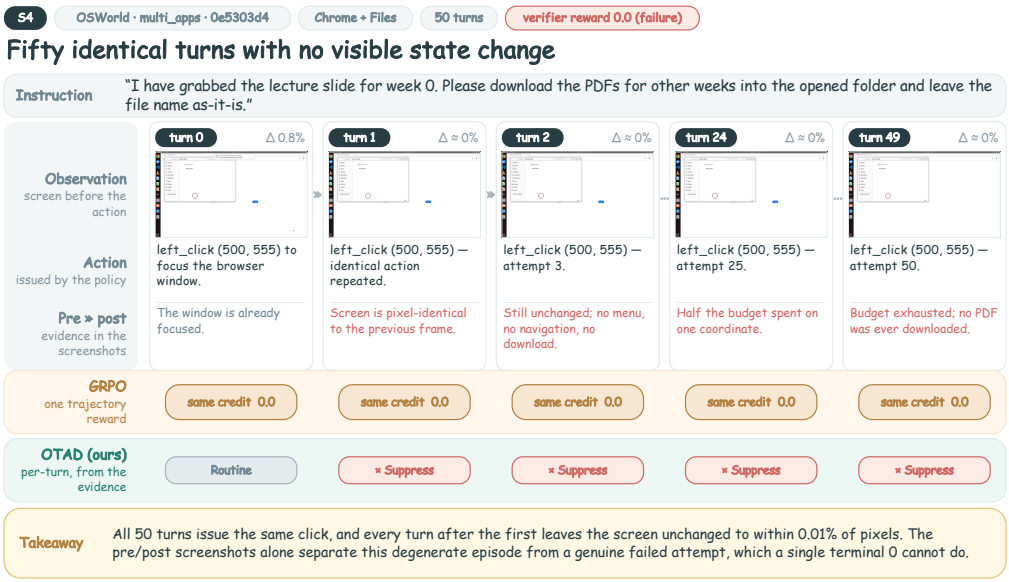

Figure 11: Fifty identical turns with no visible state change (`multi_apps/0e5303d4`, 50 turns, verifier reward 0.0). Every turn issues the same click, and every turn after the first leaves the screen unchanged to within 0.01% of pixels. The pre/post pair separates this degenerate episode from a genuine failed attempt, which a single terminal zero cannot.

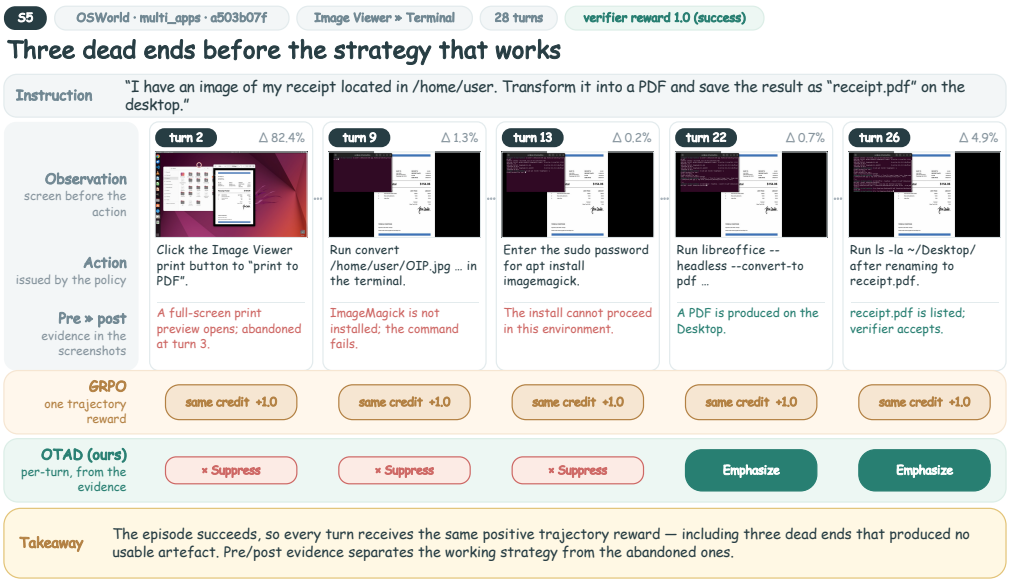

Figure 12: Three dead ends before the strategy that works (`multi_apps/a503b07f`, 28 turns, verifier reward 1.0). A print preview, a missing `convert` binary and a blocked `sudo apt install` all close without producing a usable artefact before `libreoffice --headless --convert-to pdf` writes the PDF; the trajectory reward credits the three dead ends and the working strategy alike.