BEYOND SUCCESS AND FAILURE: LENGTH-AWARE CONTRASTIVE LEARNING FOR GUI AGENTS

Chengyang Gu†& Hui Xiong*
Artificial Intelligence Thrust, Information Hub
The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
†cgu893@connect.hkust-gz.edu.cn

Le Zhang, Jingbo Zhou, Yu Shi, Siqi Bao, Zheng-Fan Wu & Hua Wu
Baidu Inc. Beijing, China

Yize Chen
Department of Electrical and Computer Engineering
University of Alberta, Edmonton, Alberta, Canada

ABSTRACT

Graphical User Interface (GUI) agents powered by Multimodal Large Language Models (MLLMs) have shown strong potential for automating tasks across diverse digital environments, where reinforcement learning (RL) has become a dominant training paradigm. However, widely used methods such as Group Relative Policy Optimization (GRPO) suffer from reward–gradient misalignment, leading to inefficient and unstable optimization. Recent work addresses this issue by reformulating RL with verifiable rewards (RLVR) as contrastive or classification-based objectives, which improve stability by eliminating problematic gradient behaviors. Despite this progress, existing contrastive RLVR methods rely primarily on outcome-level supervision and fail to capture fine-grained differences in trajectory quality within the same outcome category. In this paper, we propose Length-Aware Contrastive Learning for GUI Agents (LACL-GUI), a contrastive RLVR framework that incorporates trajectory-level quality signals into policy optimization. LACL-GUI introduces structured preferences within both successful and failed trajectories, encouraging concise successful executions and differentiating failure quality based on divergence from successful trajectories, while preserving optimization stability. Experiments on GUI agent benchmarks show that LACL-GUI provides more effective learning signals and consistently improves agent performance over prior methods, highlighting the value of trajectory-level supervision in contrastive RLVR.

1 INTRODUCTION

Graphical User Interface (GUI) agents powered by Multimodal Large Language Models (MLLMs) (Bai et al., 2025; Wang et al., 2025; Zhang et al., 2024) have demonstrated remarkable potential in automating tasks across web, mobile, and desktop platforms. Prior work typically relies on supervised fine-tuning (SFT) (Xu et al., 2024; Qin et al., 2025), which demands substantial high-quality synthetic data and often struggles to generalize to unseen interfaces (Luo et al., 2025; Lu et al., 2026). Unlike SFT, reinforcement learning (RL) enables agents to learn from trial-and-error interaction with environmental feedback, continuously refining policies without exhaustive labeled demonstrations (Zhou et al., 2026; Hu et al., 2026; Yuan et al., 2026). As a result, RL has emerged as a dominant paradigm for training advanced GUI agents (Lu et al., 2025; Hu et al., 2026).

*Correspondence to: xionghui@hkust-gz.edu.cn

Among RL-based approaches, Group Relative Policy Optimization (GRPO) is one of the dominant methods due to its strong empirical performance and computational efficiency (Shao et al., 2024; Guo et al., 2025a). By leveraging group-wise reward normalization, GRPO eliminates the need for a separate critic network while maintaining stable policy optimization. However, recent studies reveal a misalignment between reward signals and actual gradient allocation, where optimization pressure is disproportionately influenced by policy likelihood rather than trajectory quality (Deng et al., 2026; Zhang et al., 2026). (Zhai et al., 2026) theoretically identify two such gradient pathologies in GRPO: Gradient Misassignment in Positives, where high-probability tokens receive disproportionately large updates while low-probability tokens receive weak gradients despite belonging to successful trajectories; and Gradient Domination in Negatives, where gradient magnitudes concentrate excessively on a small number of high-probability negative tokens, allowing them to dominate optimization. Consequently, verifiable rewards are not faithfully reflected in policy updates, limiting both optimization efficiency and training stability.

To address reward-gradient mismatch in GRPO-style optimization, recent work has increasingly reformulated RLVR as contrastive or classification-based objectives rather than scalar reward-weighted optimization (Oord et al., 2018). Representative examples include Rewards as Labels (REAL) (Zhai et al., 2026), CLIPO (Cui et al., 2026), and Harmony (Yu et al., 2026c), which improve optimization stability, robustness, and learning effectiveness through comparison-based objectives, with REAL in particular showing that classification-based optimization induces monotonic and bounded gradient weighting that mitigates the pathologies of GRPO. However, this line of work still reduces every trajectory to a single binary label, leaving the resulting classification-based objectives blind to any structure within positive or negative groups themselves.

Despite these optimization advantages, existing contrastive RLVR methods still derive supervision primarily from outcome-level labels. Consequently, trajectories sharing the same outcome are often treated equivalently, despite potentially large differences in execution quality, efficiency, and proximity to success (Yu et al., 2026a; Zhan et al., 2026; Step, Prompt Intermediate, 2026; Zheng et al., 2026). This coarse-grained supervision limits RLVR methods' capabilities in capturing fine-grained trajectory preferences (Zhang et al., 2026). This limitation is particularly pronounced for GUI agents, where successful trajectories may vary significantly in efficiency, while failed trajectories can differ substantially in how close they come to completing the task. The central challenge is therefore how to incorporate trajectory-level quality indicators into contrastive RLVR, while preserving reliable policy optimization.

To address this challenge, we propose ***L**ength-**A**ware **C**ontrastive **L**earning for **GUI** Agents* **(LACL-GUI)**, a length-aware contrastive RLVR framework for GUI agent training. LACL-GUI work on RLVR-based objectives and introduces structured preferences within both successful and failed trajectory groups: for successful trajectories, we encourage shorter solutions by assigning stronger optimization pressure to concise executions while down-weighting redundant ones; for failed trajectories, we introduce divergence length, which measures how long a failed trajectory remains aligned with a reference successful execution before deviating, treating later-diverging failures as more informative negative samples and applying weaker suppression. These preferences are incorporated through outcome-preserving, zero-mean logit modulations, so that optimization is refined within each outcome category without altering the underlying success–failure decision boundary. As a result, LACL-GUI successfully provides fine-grained efficiency and failure-quality guidance on top of classification-based RLVR, while preserving its optimization stability and gradient safety. We evaluate LACL-GUI on OSWorld benchmark (Xie et al., 2024) using Qwen3-VL-Thinking (4B and 8B) as backbone models. Experiments show LACL-GUI consistently outperforms baselines at both scales, improving task success rate by 2.8% over GRPO and 2.7% over REAL on 8B backbone within 50 training iterations.

2 Related Works

2.1 GUI Agents

GUI Agents aim to execute tasks across web, mobile, and desktop environments by mapping high-level instructions to grounded interface actions. Early approaches rely on structured representations, such as HTML (Deng et al., 2023) and DOM trees (Wen et al., 2024). However, these methods depend heavily on the coverage and quality of such representations, which are often brittle and

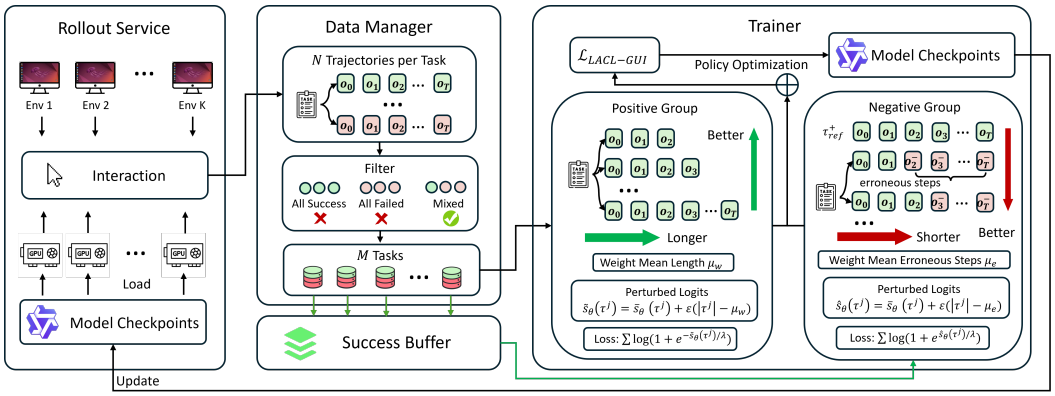

Figure 1: Overview of LACL-GUI framework. A decoupled architecture collects informative trajectory groups and maintains a *Success Buffer* for reference trajectories. The proposed objective introduces length-aware positive preference and divergence-aware negative refinement on top of REAL contrastive objective.

inconsistent across platforms (Wu et al., 2026; Tang et al., 2026). Recent approaches leverage Multimodal Large Language Models (MLLMs) to enable more general and visually grounded interactions. Given natural language instructions and interface observations, MLLM-based GUI agents such as UI-TARS (Qin et al., 2025), OS-Atlas (Wu et al., 2025), and UGround (Gou et al., 2025) parse model responses into executable actions through predefined prompting and action schemas. In this work, we follow the MLLM-based paradigm and adopt the Qwen3-VL series model (Bai et al., 2025) with its action parser as the backbone for policy optimization.

2.2 Reinforcement Learning

Reinforcement learning (RL) has become a promising paradigm for enhancing the reasoning and decision-making capabilities of LLM and MLLM-based agents (Guo et al., 2025a; Jaech et al., 2024; Team et al., 2025), particularly in multi-turn interactive environments. Early approaches largely rely on reinforcement learning from human feedback (RLHF), including methods such as Direct Preference Optimization (DPO) (Rafailov et al., 2023), which derive reward signals from offline human preference data. However, RLHF suffers from high annotation costs and noisy reward estimation (Kaufmann et al., 2023). Recent work therefore shifts toward Reinforcement Learning with Verifiable Rewards (RLVR), where rewards are directly obtained from task outcomes. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a representative RLVR approach that removes the need for a critic model through group-wise advantage estimation. Subsequent variants, including Dr.GRPO (Liu et al., 2025), DAPO (Yu et al., 2026b), and GSPO (Zheng et al., 2025), further improve training stability and efficiency. Nevertheless, recent studies such as REAL (Zhai et al., 2026) reveal that GRPO-style objectives suffer from gradient misalignment, where policy updates can be dominated by token likelihood rather than trajectory quality. These findings motivate exploration of alternative RLVR objectives beyond conventional reward-weighted operations.

Recent works also explore finer-grained credit assignment for GUI agents beyond sparse outcome rewards. GiGPO (Feng et al., 2026) introduces milestone-based rewards through manually designed environment-specific rules, while other approaches estimate intermediate progress using pretrained representation models (Zheng et al., 2026; Wang et al., 2026b). However, these approaches either require task-specific supervision or additional models, limiting their scalability across diverse GUI environments. In contrast, our approach derives trajectory-level quality signals directly from successful and failed rollouts within a unified contrastive optimization framework.

2.3 Contrastive Learning for RLVR

Contrastive learning is a widely used representation learning paradigm that encourages positive samples to be closer while separating negative samples (Sammani et al., 2023). Since RLVR methods typically rely on binary outcome rewards (Parthasarathi et al., 2025), recent studies show that ob-

jectives such as GRPO can be interpreted from a classification perspective (Mroueh, 2025). This perspective has inspired methods that replace reward-weighted optimization with discriminative or contrastive objectives. REAL (Zhai et al., 2026) formulates RLVR as a binary classification problem using BCE loss, providing a stable optimization alternative by contrasting successful and unsuccessful trajectories. Subsequent works, including CLIPO (Cui et al., 2026) and ConSPO (Zhang et al., 2026), further introduce explicit contrastive objectives based on InfoNCE (Oord et al., 2018). However, existing approaches mainly operate at the trajectory outcome level, where trajectories within the same outcome category receive identical supervision. Our LACL-GUI addresses this limitation by introducing trajectory-level quality preferences into contrastive RLVR.

3 METHODOLOGY

3.1 PROBLEM FORMULATION

We formulate GUI task completion as a sequential decision-making problem under a Partially Observable Markov Decision Process (POMDP). Given a natural language instruction q, an agent interacts with the environment over multiple steps to complete the task. At each timestep t, the agent receives a visual observation $o_t \in \mathcal{O}$, represented as a screenshot of the interface, instead of accessing the full underlying state. To mitigate partial observability, the agent conditions on a finite history window $h_t = \{(o_{t-k}, z_{t-k}, a_{t-k}), \ldots, (o_{t-1}, z_{t-1}, a_{t-1})\}$ of the most recent k interactions, where z_t denotes an intermediate reasoning trace and $a_t \in \mathcal{A}$ denotes a structured executable action (e.g., parameterized tool calls such as clicking at specific coordinates). A policy π_θ, parameterized by a vision-language model, generates a reasoning-action pair:

$$(z_t, a_t) \sim \pi_\theta(\cdot \mid o_t, h_t, q).$$

In practice, z_t serves as auxiliary reasoning tokens, while a_t is the executable action applied to the environment. Executing action a_t leads to the next observation o_{t+1} according to the environment dynamics. An episode produces a trajectory

$$\tau = \{(o_0, z_0, a_0), (o_1, z_1, a_1), \ldots, (o_T, z_T, a_T)\},$$

which terminates either when the task is completed or when a maximum horizon is reached.

We consider a sparse outcome-based reward setting, where supervision is provided only at the trajectory level:$R(\tau) \in \{0, 1\}$, indicating task success or failure. All steps within a trajectory share the same outcome-level supervision, without any intermediate rewards for individual actions. The objective is to learn a policy π_θ that maximizes the expected success rate $\theta^* = \arg\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$.

3.2 ARCHITECTURE

Following DART (Li et al., 2025), we adopt an asynchronous training architecture for proposed LACL-GUI framework. The system consists of four modules: *Rollout Service*, *Data Manager*, *Success Buffer*, and *Trainer*. The *Rollout Service* collects trajectories from multiple GUI environments in parallel, which are filtered and organized by the *Data Manager*. Successful trajectories are stored in the *Success Buffer* for reuse and divergence estimation, while filtered trajectory groups are sent to the *Trainer* for policy optimization with the LACL-GUI objective. Updated parameters are periodically synchronized back to the rollout workers, forming an asynchronous training loop.

Rollout Service Interacting with real-world environments to solve GUI tasks using vision-language models is time-consuming. To address this, we employ asynchronous rollout-wise sampling following DART (Li et al., 2025), where each environment independently starts a new trajectory after completion without waiting for other workers. This design improves environment utilization and enables efficient trajectory collection. For each task instruction, we collect N trajectories under the same initial condition, forming trajectory groups for contrastive optimization.

Data Manager The *Data Manager* filters sampled trajectories to construct informative training groups. Since LACL-GUI relies on contrastive optimization between successful and failed trajectories, groups where all N trajectories share the same outcome (i.e., all success or all failure) provide no discriminative supervision and are discarded. For each training iteration, the *Data Manager* collects a batch of M tasks whose trajectory groups contain both successful and failed samples, and

then forwards them to the *Trainer*. Following DART, policy updates are performed after all trajectories in the batch are collected, enabling meaningful within-task comparisons among sampled trajectories.

Success Buffer Sparse rewards in GUI tasks often lead to insufficient successful trajectories during training. To improve sample efficiency, we maintain a *Success Buffer* that stores successful trajectories for each task. The buffer serves two purposes: first, it provides additional positive samples when current rollouts contain limited successes; second, it provides reference successful trajectories for divergence-aware failure analysis. Specifically, for each failed trajectory ($R(\tau) = 0$), we estimate its quality by comparing it with a successful reference trajectory from the same task. The reference is selected as the shortest successful trajectory from the union of the *Success Buffer* and current-batch successful trajectories, while only current-batch successful trajectories are considered when the buffer is unavailable.

Trainer The *Trainer* is decoupled from rollout workers and asynchronously optimizes the policy using filtered trajectory groups to eliminate blocking between data collection and policy optimization. After each update, the latest parameters are synchronized back to the rollout service, allowing trajectory collection and policy optimization to proceed concurrently.

Overall, this architecture provides an efficient infrastructure for collecting structured trajectory groups and maintaining high-quality successful references, which are essential for the trajectory-level contrastive optimization in LACL-GUI. We next introduce our proposed objective in detail.

3.3 LACL-GUI Objective

3.3.1 Revisiting REAL Objective

We first revisit the REAL (Zhai et al., 2026) objective, which formulates RLVR from a classification perspective. Given an instruction q, a set of N trajectories $\{\tau^1, \ldots, \tau^N\}$ are sampled from the same policy. Each trajectory τ^j is assigned a binary reward $r^j \in \{0, 1\}$ by a verifier indicating task success. Based on these labels, trajectories are partitioned into two disjoint sets:

$$\mathcal{T}^+ = \{\tau^j | r^j = 1\}, \quad \mathcal{T}^- = \{\tau^j | r^j = 0\}.$$

For each trajectory τ^j in both $\mathcal{T}^+$ and $\mathcal{T}^-$, REAL defines a length-normalized relative log probability score over its actions as *logits*:

$$\bar{s}^j = \frac{1}{|\tau^j|} \sum_{t=1}^{|\tau^j|} (\log \frac{\pi_\theta(a_t|o_t, h_t, q)}{\pi_{old}(a_t|o_t, h_t, q)}).$$

where $|\tau^j|$ denotes trajectory length, and a_t, o_t, and h_t represent the action, observation, and history at step t, respectively. The score measures the relative change of trajectory probability under the updated policy compared to the old policy. Specifically, $\bar{s}^j > 0$ indicates that the trajectory is promoted, whereas $\bar{s}^j < 0$ indicates suppression. The binary reward determines the optimization direction by separating desirable and undesirable trajectories, while the logits reflect the magnitude of optimization updates.

Based on logits $\bar{s}^j$, REAL optimizes the following contrastive-style objective:

$$\mathcal{L}_{REAL} = \log(1 + \sum_{\mathcal{T}^+} e^{-\bar{s}^j/\lambda}) + \log(1 + \sum_{\mathcal{T}^-} e^{\bar{s}^j/\lambda}). \tag{1}$$

where λ is a temperature parameter. This objective encourages positive trajectories and suppresses negative ones, with adaptive gradient allocation determined by $\bar{s}^j$. Specifically, under-optimized positive trajectories with smaller logits receive stronger updates, while over-confident positives are down-weighted; similarly, negative trajectories with larger logits receive stronger penalties. Such adaptive and bounded gradient allocation alleviates the gradient misalignment issues observed in prior RLVR methods (Zhai et al., 2026).

However, Eq. equation 1 only distinguishes trajectories at the task-outcome level and does not capture fine-grained differences within $\mathcal{T}^+$ or $\mathcal{T}^-$. Trajectories with identical outcomes can still exhibit substantially different qualities. For example, two successful GUI trajectories may complete the same task, while one requires only necessary interactions and the other contains redundant exploratory actions. Similarly, among failed trajectories, one may follow the correct procedure for most steps before making a final mistake, whereas another may diverge immediately from the desired behavior. Although these trajectories receive identical binary rewards, they provide different optimization signals. Since REAL does not explicitly model such intra-group preferences, it cannot encourage efficient successful executions or distinguish informative failures from fundamentally incorrect attempts. This limitation motivates our proposed LACL-GUI objective.

3.3.2 POSITIVE GROUP LOGITS PERTURBATION

For fine-grained comparison within successful trajectories, we observe that GUI task execution naturally exhibits an efficiency preference. Among trajectories starting from the same initial state and achieving the same goal, shorter executions are generally preferred, as longer successful trajectories often contain redundant interactions caused by unnecessary exploration, incorrect actions followed by recovery, or repeated operations. For example, in GUI environments, an agent may perform unnecessary exploratory movements, repeatedly click irrelevant regions, or revisit previously completed steps before eventually reaching the correct state. Such inefficiency is particularly undesirable in GUI environments, where each interaction introduces additional inference and execution cost while increasing the possibility of future errors.

Based on this observation, we introduce a length-aware perturbation within the positive group in the logit space. Rather than modifying binary success supervision, the proposed perturbation preserves REAL's success preference while introducing additional efficiency-aware guidance among successful trajectories.

For the positive set $\mathcal{T}^+$, we define the softmax weighting based on logits $\bar{s}_\theta^j$ under current policy π_θ:

$$p^+(\tau^j) = \frac{e^{-\bar{s}_\theta(\tau^j)/\lambda}}{\sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda}}. \tag{2}$$

We then compute the weighted mean trajectory length:

$$\mu_w = \sum_{\tau^j \in \mathcal{T}^+} p^+(\tau^j)\,|\tau^j|. \tag{3}$$

Based on this efficiency preference, we perturb the positive logits as:

$$\tilde{s}_\theta(\tau^j) = \bar{s}_\theta(\tau^j) + \epsilon\left(|\tau^j| - \mu_w\right), \quad \tau^j \in \mathcal{T}^+. \tag{4}$$

where $\epsilon > 0$ is a small perturbation coefficient controlling the magnitude of the logit adjustment. This perturbation introduces a relative efficiency preference within the positive group. Under the exponential weighting $p(\tau^j) \propto \exp(-\tilde{s}_\theta(\tau^j)/\lambda)$, shorter successful trajectories receive relatively larger optimization weights, whereas longer trajectories are down-weighted. Therefore, LACL-GUI encourages efficient successful executions without changing the original success preference defined by REAL.

3.3.3 NEGATIVE GROUP LOGITS PERTURBATION

While REAL suppresses all failed trajectories uniformly, failures may contain different levels of learning value. In GUI tasks, a trajectory that follows successful behaviors for most steps before deviating is generally closer to task completion than one that fails at an early stage. We therefore introduce a *divergence-aware perturbation* to distinguish failed trajectories according to their proximity to successful executions.

For each task, given a negative trajectory $\tau^j \in \mathcal{T}^-$, we compare it with a reference successful trajectory τ_{ref}^+:

$$\begin{aligned} \tau^j &= \{(o_1^j, a_1^j), \ldots, (o_{|\tau^j|}^j, a_{|\tau^j|}^j)\}; \\ \tau_{ref}^+ &= \{(o_1^{ref}, a_1^{ref}), \ldots, (o_{|\tau_{ref}^+|}^{ref}, a_{|\tau_{ref}^+|}^{ref})\}. \end{aligned} \tag{5}$$

The reference trajectory τ^+_{ref} is selected as the shortest successful trajectory among the success buffer and the current rollout group. Since trajectories of the same task share an identical initial state, their observations can be compared step by step to identify the first divergence point.

We compute the observation similarity $\mathrm{SIM}(o_t^{ref}, o_t^j)$ at each step using Structural Similarity Index (SSIM) (Brunet et al., 2012), following (Lin et al., 2026). Given a threshold Θ, the divergence length is defined as:

$$d^j = \min\{t|\mathrm{SIM}(o_t^{ref}, o_t^j) < \Theta\} - 1. \tag{6}$$

If no divergence is detected, we set $d^j = |\tau^j|$.

Based on the divergence length, we define the number of *erroneous steps*

$$E^j = |\tau^j| - d^j + 1. \tag{7}$$

which measures the remaining execution length after deviating from the successful trajectory.

Analogous to the positive group, we compute the weighted average erroneous steps:

$$\begin{aligned} p^-(\tau^j) &= \frac{e^{\bar{s}_\theta(\tau^j)/\lambda}}{\sum_{\tau^j \in \mathcal{T}^-} e^{\bar{s}_\theta(\tau^j)/\lambda}}; \\ \mu_e &= \sum_{\tau^j \in \mathcal{T}^-} p^-(\tau^j)\,|E^j|. \end{aligned} \tag{8}$$

The perturbed logits are then given by

$$\hat{s}_\theta(\tau^j) = \bar{s}_\theta(\tau^j) + \epsilon\left(E^j - \mu_e\right), \quad \tau^j \in \mathcal{T}^-. \tag{9}$$

This perturbation introduces a *relative error-severity preference* within the negative group. Failures with larger erroneous-step counts, corresponding to earlier divergence from successful executions, receive larger logits and therefore stronger suppression under the negative loss. In contrast, near-success trajectories with fewer post-divergence steps are penalized more gently. Compared with the uniform negative suppression in REAL, LACL-GUI preserves informative near-success failures while still reducing the influence of fundamentally incorrect execution patterns.

3.3.4 Overall Loss

Combining the positive and negative logit perturbations, the LACL-GUI objective is formulated as

$$\mathcal{L}_{\text{LACL-GUI}} = \log\big(1 + \sum_{\tau^j \in \mathcal{T}^+} e^{-\tilde{s}_\theta(\tau^j)/\lambda}\big) + \log\big(1 + \sum_{\tau^j \in \mathcal{T}^-} e^{\hat{s}_\theta(\tau^j)/\lambda}\big), \tag{10}$$

where $\tilde{s}_\theta(\tau)$ and $\hat{s}_\theta(\tau)$ denote the perturbed logits defined for the positive and negative groups, respectively.

In practice, the group-wise statistics μ_w and μ_e are detached from the computational graph using the stop-gradient operator. Consequently, gradients are propagated only through the perturbed logits, while the group statistics are treated as constants during optimization.

Theorem 1. *(First-order Invariance). Assume that group-wise statistics μ_w and μ_e hold as constants via the stop-gradient operation. Then LACL-GUI objective satisfies:*

$$\mathcal{L}_{LACL-GUI} = \mathcal{L}_{REAL} + \mathcal{O}(\epsilon^2). \tag{11}$$

Theorem 1 (See proof in Appendix) shows that LACL-GUI preserves the first-order optimization objective of REAL while introducing higher-order logit adjustments. Therefore, proposed perturbations maintain the original success-failure preference encoded by REAL at the first order, while reshaping gradient allocation and local optimization geometry within each outcome group. Consequently, LACL-GUI refines optimization without changing the underlying success-failure classification boundary. Unlike explicit length penalties that modify the optimization target, LACL-GUI introduces efficiency and error-severity preferences effectively through gradient modulation. LACL-GUI augments trajectory-level classification with fine-grained within-group preferences. By exploiting relative efficiency and divergence information, LACL-GUI provides more informative optimization signals while retaining the stability benefits of classification-based RLVR.

Model	Max Steps	gimp	calc	impress	writer	os	thunderbird	vlc	vs_code	Overall
Closed-source Models										
OpenAI CUA o3 (OpenAI, 2025)	100	38.5	10.6	10.6	30.4	62.5	26.7	39.2	39.1	27.8
OpenAI CUA (OpenAI, 2025)	50	34.6	14.9	29.7	26.1	70.8	66.7	11.6	69.6	36.5
Claude 3.7 Sonnet (Anthropic, 2025a)	50	38.5	31.9	36.1	43.5	50.0	53.3	23.5	56.5	40.1
Seed1.5-VL-250717 (Guo et al., 2025b)	100	50.0	34.8	48.9	56.5	39.1	73.3	35.3	56.5	45.1
Claude 4 Sonnet (Anthropic, 2025b)	50	50.0	31.9	46.7	60.9	45.8	73.3	41.2	60.9	47.3
DeepMiner-Mano-7B (Fu et al., 2025)	100	69.2	27.7	42.5	56.5	50.0	73.3	35.3	78.3	47.9
Open-source Models										
OpenCUA-7B (Wang et al., 2026a)	50	43.6	13.2	32.6	33.3	43.5	42.2	28.3	47.1	32.9
UI-TARS-1.5-7B (Qin et al., 2025)	100	51.9	9.6	38.2	39.1	31.3	40.0	22.4	47.8	33.0
UI-TARS-72B-dpo (Qin et al., 2025)	100	73.1	6.4	23.8	34.8	37.5	60.0	17.7	52.2	33.4
OpenCUA-32B (Wang et al., 2026a)	100	66.7	18.4	37.6	36.2	55.1	46.7	33.3	63.3	41.6
DART-GUI-7B-0924 (Li et al., 2025)	30	80.8	14.9	44.7	47.8	54.2	66.7	27.2	69.6	46.7
Baselines										
Qwen3-VL-4B-Thinking (Bai et al., 2025)	50	69.2	25.3	19.1	43.5	56.5	45.8	60.0	56.3	41.2
Qwen3-VL-4B-GRPO	50	73.1	10.6	31.7	56.5	54.2	66.7	46.1	47.8	42.2
Qwen3-VL-4B-REAL	50	76.9	17.0	27.5	52.2	50.0	66.7	66.4	47.8	43.0
Qwen3-VL-8B-Thinking (Bai et al., 2025)	50	76.9	12.8	29.6	52.2	62.5	60.0	23.5	65.2	42.7
Qwen3-VL-8B-GRPO	50	65.4	25.5	33.8	56.5	66.7	73.3	34.6	60.9	47.2
Qwen3-VL-8B-REAL	50	76.9	21.3	33.8	69.6	66.7	60.0	41.2	47.8	47.3
Proposed										
LACL-GUI-4B	50	69.2	21.3	31.7	52.2	58.3	66.7	41.2	56.5	44.5
LACL-GUI-8B	50	76.9	27.7	36.0	65.2	70.8	66.7	47.1	47.8	50.0

Table 1: Task success rate (%) on the OSWorld benchmark across different application domains. Results are reported following the official OSWorld evaluation protocol. Qwen3-VL-GRPO and Qwen3-VL-REAL denote models trained with GRPO and REAL objectives, respectively, based on the corresponding Qwen3-VL backbones.

4 EXPERIMENTS

Settings of Benchmark: We evaluate LACL-GUI on **OSWorld** (Xie et al., 2024), a comprehensive benchmark for evaluating MLLM-based GUI agents in realistic computer environments. Following (Wang et al., 2026b), we select 222 tasks from the original 369-task benchmark for training and evaluation (detailed settings are provided in the Appendix). This subset removes tasks affected by external instability, such as unreliable network conditions and corrupted files, which may introduce evaluation noise. The selected tasks cover four application domains: *(i) Operating System (OS), (ii) Office* (LibreOffice Calc, Impress, Writer), *(iii) Daily-use software* (VLC Player, Thunderbird), and *(iv) Professional tools* (VS Code, Gimp). This diversity ensures a comprehensive coverage of realistic user scenarios. We follow the standard OSWorld evaluation protocol (Xie et al., 2024), where execution-based validation scripts determine task success and assign trajectory rewards $R(\tau) \in \{0, 1\}$.

Implementation: We adopt Qwen3-VL-8B-Thinking and Qwen3-VL-4B-Thinking (Bai et al., 2025) as backbone models for LACL-GUI and baselines training. Each trajectory is limited to a maximum of 50 interaction steps. At each iteration, we sample 6 tasks and collect $N = 8$ trajectories per task after filtering by the *Data Manager*. The temperature parameter λ and perturbation coefficient ϵ are fixed to 1.0 and 0.03, respectively, while the history horizon of h_t is set to 5 steps following DART (Li et al., 2025). All models are trained with 8 NVIDIA H100 GPUs for up to 50 iterations. We report the best checkpoint selected according to evaluation performance across training iterations.

4.1 MAIN RESULTS

Overall Performance. Table 1 presents the performance comparison on the OSWorld benchmark across 8 applications. Unless otherwise specified, LACL-GUI models are trained with a similarity threshold $\Theta = 0.85$. LACL-GUI achieves competitive performance against both open-source and closed-source baselines. In particular, LACL-GUI-8B achieves an overall success rate of 50.0%, improving the backbone model Qwen3-VL-8B-Thinking by 7.3%. It also surpasses several strong open-source and closed-source models, including DART-GUI-7B (46.7%) and DeepMiner-Mano-

7B (47.9%). Moreover, under the same backbone architecture, LACL-GUI consistently outperforms RLVR baselines, achieving improvements over GRPO (47.2%) and REAL (47.3%). These results demonstrate the effectiveness of incorporating trajectory-level quality information into contrastive RLVR optimization.

Effect over RL Baselines. Compared with existing RL-based objectives, LACL-GUI consistently improves performance across different model scales. On the 8B backbone, LACL-GUI improves over REAL and GRPO by 2.7 and 2.8 percentage points, respectively. Similarly, on the 4B backbone, LACL-GUI achieves a higher success rate than REAL and GRPO (44.5% vs. 43.0%, 42.2%). These gains indicate that explicitly modeling preferences within successful and failed trajectory groups provides more informative optimization signals than outcome-only supervision.

Domain-wise Improvements. LACL-GUI achieves substantial improvements across several application domains. In particular, LACL-GUI-8B obtains strong performance on *OS* (70.8%) and *Calc* (27.7%), while also achieving competitive results on *Gimp* (76.9%), *Writer* (65.2%), and *Thunderbird* (66.7%). These improvements demonstrate the effectiveness of LACL-GUI in handling diverse GUI scenarios.

Summary. Overall, LACL-GUI provides more informative trajectory-level supervision than existing RL objectives, leading to consistent improvements in GUI agent performance across varying scales and application domains.

4.2 Ablation Studies

Configuration	Success Rate (%)	Gain (%)
Prompt Baseline	42.7	–
+ Decoupled Architecture	45.4	+2.7
+ Success Buffer	46.5	+1.1
+ Trajectory Filter	47.2	+0.7
+ LACL-GUI Objective	**50.0**	**+2.8**

Table 2: Ablation study of different components in the LACL-GUI framework. Each row incrementally introduces one additional component over the previous configuration. The prompt-based baseline uses Qwen3-VL-8B-Thinking, while GRPO is adopted for all training configurations except the proposed LACL-GUI objective.

4.2.1 Effect of Architecture Components

We first investigate the contribution of each component in LACL-GUI by progressively adding the decoupled architecture, success buffer, trajectory filtering, and finally the LACL-GUI objective to the prompt-based Qwen3-VL-8B-Thinking baseline. Results are shown in Table 2.

The architecture components improve performance from 42.7% to 47.2%, demonstrating the benefits of efficient rollout management, successful trajectory reuse, and informative contrastive groups under sparse rewards. Introducing the LACL-GUI objective achieves the largest single gain (+2.8%), further improving performance to 50.0%. Since this improvement is obtained on top of the complete GRPO-based framework, the gain mainly comes from the proposed length-aware preference modeling and divergence-aware failure refinement rather than architectural changes alone.

4.2.2 Effect of Length-aware Preference Modeling

We further evaluate whether LACL-GUI objective improves trajectory efficiency beyond outcome-based optimization. To isolate trajectory preference from success-rate differences, we only analyze tasks where all methods achieve the same outcome. Results are shown in Figure 2.

For successful trajectories, LACL-GUI reduces the average length from 9.13 steps (REAL) to 8.19 steps, achieving a 10.3% reduction, while the base model requires 19.84 steps. This verifies our

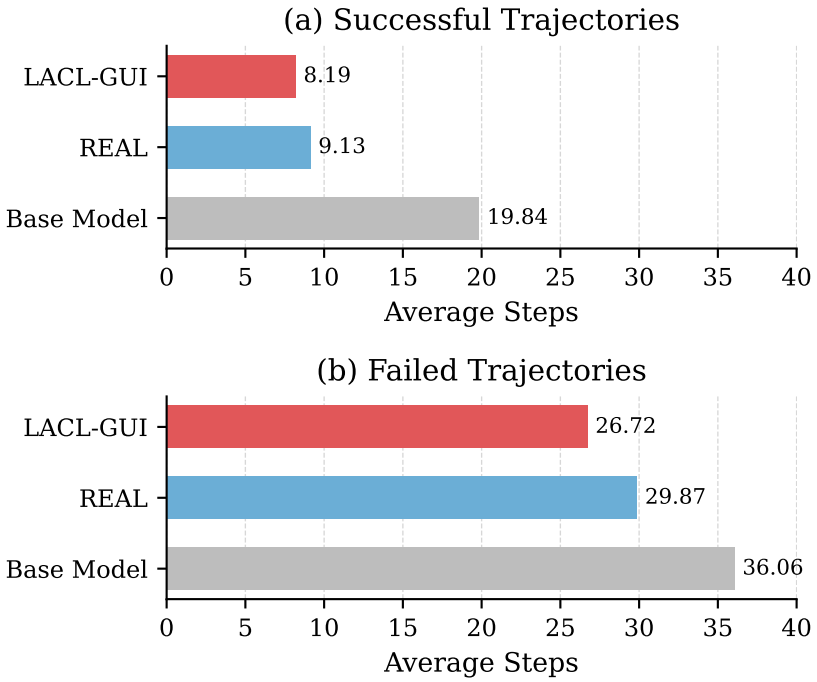

Figure 2: Effect of length-aware preference modeling in LACL-GUI. We compare average trajectory lengths on tasks where all three methods achieve the same outcome.

motivation that, under the same initial state and goal, shorter successful executions generally indicate fewer redundant interactions and higher-quality solutions. For failed trajectories, LACL-GUI similarly reduces the average length from 29.87 to 26.72 steps (10.5% reduction). Unlike global length penalties that may favor short failures over successful behaviors, LACL-GUI performs length modulation within the negative group through divergence-aware erroneous-step modeling, suppressing prolonged ineffective failed trajectories while preserving near-success trajectories as informative signals.

4.2.3 Sensitivity Analysis of Divergence Estimation

The divergence estimation in LACL-GUI relies on a similarity threshold Θ to determine the deviation point between failed and successful trajectories. We evaluate its sensitivity by varying Θ from 0.80 to 0.90 and report the resulting performance across different domains in Figure 3.

LACL-GUI maintains stable performance across different threshold settings. The overall success rate changes only marginally (49.5%, 50.0%, and 49.9%), despite larger variations in individual domains. This demonstrates that the effectiveness of divergence-aware failure refinement does not rely on precise threshold selection. Instead, the relative ordering of failed trajectories according to their divergence behaviors remains stable, enabling reliable identification of near-successful failures.

5 Conclusion

In this work, we present LACL-GUI, a length-aware contrastive learning framework for training GUI agents under sparse trajectory-level supervision. LACL-GUI introduces fine-grained preference modeling within both successful and failed trajectory groups. It encourages efficient successful behaviors through length-aware positive trajectory refinement and distinguishes informative near-successful failures from early-diverging attempts through divergence-aware failure modeling. To support optimization in long-horizon GUI environments, a decoupled asynchronous training architecture is employed with trajectory filtering and a successful trajectory buffer, enabling efficient rollouts collection and providing successful references for divergence estimation. Extensive exper-

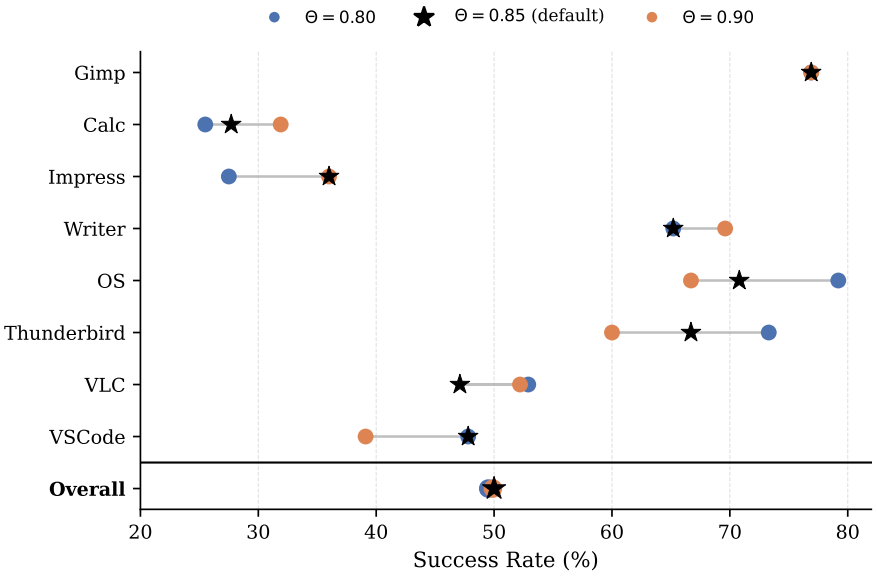

Figure 3: Sensitivity analysis of the similarity threshold Θ for divergence estimation in LACL-GUI. The default setting ($\Theta = 0.85$) is highlighted, and results are reported across different domains.

iments on the OSWorld benchmark demonstrate that LACL-GUI consistently improves GUI task completion performance over existing RL-based baselines. Future work will explore more adaptive trajectory quality estimation strategies and extend to broader interactive environments.

REFERENCES

Anthropic. Claude 3.7 sonnet system card. https://www-cdn.anthropic.com/9ff93dfa8f445c932415d335c88852ef47f1201e/claude-3-7-sonnet-system-card.pdf, 2025a.

Anthropic. System card: Claude opus 4 & claude sonnet 4. System card, Anthropic, May 2025b. URL https://www-cdn.anthropic.com/4263b940cabb546aa0e3283f35b686f4f3b2ff47/claude-opus-4-and-claude-sonnet-4-system-card.pdf.

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

Dominique Brunet, Edward R Vrscay, and Zhou Wang. On the mathematical properties of the structural similarity index. *IEEE Trans. Image Process.*, 21(4):1488–1499, 2012.

Sijia Cui, Pengyu Cheng, Jiajun Song, Yongbo Gai, Guojun Zhang, Zhechao Yu, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. Clipo: Contrastive learning in policy optimization generalizes rlvr. *arXiv preprint arXiv:2603.10101*, 2026.

Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. On group relative policy optimization collapse in agent search: The lazy likelihood-displacement. In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*, 2026.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

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.

Tianyu Fu, Anyang Su, Chenxu Zhao, Hanning Wang, Minghui Wu, Zhe Yu, Fei Hu, Mingjia Shi, Wei Dong, Jiayao Wang, et al. Mano technical report. *arXiv preprint arXiv:2509.17336*, 2025.

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

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

Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, et al. Seed1. 5-vl technical report. *arXiv preprint arXiv:2505.07062*, 2025b.

Junan Hu, Jian Liu, Jingxiang Lai, Jiarui Hu, Yiwei Sheng, Shuang Chen, Jian Li, Dazhao Du, and Song Guo. Gui agents with reinforcement learning: Toward digital inhabitants. *arXiv preprint arXiv:2604.27955*, 2026.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*, 2023.

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

Zichuan Lin, Feiyu Liu, Yijun Yang, Jiafei Lyu, Yiming Gao, Yicheng Liu, Zhicong Lu, Yangbin Yu, Mingyu Yang, Junyou Li, et al. Ui-voyager: A self-evolving gui agent learning via failed experience. *arXiv preprint arXiv:2603.24533*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

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

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

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

Youssef Mroueh. Reinforcement learning with verifiable rewards: Grpo's effective loss, dynamics, and success amplification. *arXiv preprint arXiv:2503.06639*, 2025.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

OpenAI. Openai o3 and o4-mini system card. Technical report, OpenAI, 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf.

Prasanna Parthasarathi, Mathieu Reymond, Boxing Chen, Yufei Cui, and Sarath Chandar. Grpo-lambda: Credit assignment improves llm reasoning. *arXiv preprint arXiv:2510.00194*, 2025.

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Fawaz Sammani, Boris Joukovsky, and Nikos Deligiannis. Visualizing and understanding contrastive learning. *IEEE Transactions on Image Processing*, 33:541–555, 2023.

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.

Step, Prompt Intermediate. Branpo: Scalable contrastive branch sampling for long-horizon agentic reinforcement learning. 2026.

Jiaqi Tang, Yu Xia, Yi-Feng Wu, Yuwei Hu, Chen Yuhui, Qing-Guo Chen, Xiaogang Xu, Xiangyu Wu, Hao Lu, Yanqing Ma, et al. Lpo: Towards accurate gui agent interaction via location preference optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14617–14628, 2026.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

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

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, 2026a.

Yinjie Wang, Tianbao Xie, Ke Shen, Mengdi Wang, and Ling Yang. Rlanything: Forge environment, policy, and reward model in completely dynamic rl system. *arXiv preprint arXiv:2602.02488*, 2026b.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th annual international conference on Mobile computing and networking*, pp. 543–557, 2024.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *Advances in Neural Information Processing Systems*, 38:15101–15128, 2026.

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

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.

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. *arXiv preprint arXiv:2412.04454*, 2024.

Qinan Yu, Alexa Tartaglini, Peter Hase, Carlos Guestrin, and Christopher Potts. Outcome rewards do not guarantee verifiable or causally important reasoning. *arXiv preprint arXiv:2604.22074*, 2026a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026b.

Zongji Yu, Wenshui Luo, Yiliu Sun, Hao Fang, Runmin Cong, Chaochao Lu, and Chen Gong. Harmony in diversity: Multi-domain contrastive policy optimization for large reasoning models. *arXiv preprint arXiv:2605.25443*, 2026c.

Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Se-gui: Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *Advances in Neural Information Processing Systems*, 38:127658–127679, 2026.

Zepeng Zhai, Meilin Chen, Jiaxuan Zhao, Junlang Qian, Lei Shen, and Yuan Lu. Rewards as labels: Revisiting rlvr from a classification perspective. *arXiv preprint arXiv:2602.05630*, 2026.

Yuliang Zhan, Xinyu Tang, Jian Li, Dandan Zheng, Weilong Chai, Jingdong Chen, Jun Zhou, Ge Wu, Wenyue Tang, and Hao Sun. Graphpo: Graph-based policy optimization for reasoning models, 2026. URL `https://arxiv.org/abs/2606.18954`.

Feng Zhang, Xinhong Ma, Ziqiang Dong, Xi Leng, Jianfei Zhao, Xin Sun, Yang Yang, and Guanjun Jiang. Revisiting reinforcement learning with verifiable rewards from a contrastive perspective. *arXiv preprint arXiv:2605.12969*, 2026.

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

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Congmin Zheng, Xiaoyun Mo, Xinbei Ma, Qiqiang Lin, Yin Zhao, Jiachen Zhu, Xingyu Lou, Jun Wang, Zhaoxiang Wang, Weiwen Liu, et al. Adaptive milestone reward for gui agents. *arXiv preprint arXiv:2602.11524*, 2026.

Yuqi Zhou, Sunhao Dai, Shuai Wang, Kaiwen Zhou, Qinglin Jia, and Jun Xu. Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents. *Advances in Neural Information Processing Systems*, 38:95683–95705, 2026.

A Appendix

A.1 Proof of Theorem 1

We first prove Lemma 1:

Lemma 1 (Weighted Zero-Mean Property)**.** *For the proposed logit perturbations in LACL-GUI, the weighted perturbations satisfy*

$$\sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda}\big(|\tau^j| - \mu_w\big) = 0 \tag{12}$$

and

$$\sum_{\tau^j \in \mathcal{T}^-} e^{\bar{s}_\theta(\tau^j)/\lambda}\big(E^j - \mu_e\big) = 0. \tag{13}$$

Proof. For the positive group, the weighted mean trajectory length is defined as

$$\mu_w = \sum_{\tau^j \in \mathcal{T}^+} p^+(\tau^j)\,|\tau^j|, \tag{14}$$

where

$$p^+(\tau^j) = \frac{e^{-\bar{s}_\theta(\tau^j)/\lambda}}{\sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda}}. \tag{15}$$

Therefore,

$$\begin{aligned}
&\sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda}\big(|\tau^j| - \mu_w\big) \\
&= \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda}\,|\tau^j| - \mu_w \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda} \\
&= \mu_w \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda} - \mu_w \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda} \\
&= 0.
\end{aligned}$$

The negative-group identity follows by the same argument with the corresponding weighted mean μ_e and weights $e^{\bar{s}_\theta(\tau^j)/\lambda}$. □

Theorem 1. *(First-order Invariance). Assume that group-wise statistics μ_w and μ_e hold as constants via the stop-gradient operation. Then LACL-GUI objective satisfies:*

$$\mathcal{L}_{LACL-GUI} = \mathcal{L}_{REAL} + \mathcal{O}(\epsilon^2). \tag{16}$$

Proof. We denote the positive-group loss of REAL and LACL-GUI by $\mathcal{L}^+_{\mathrm{REAL}}$ and $\tilde{\mathcal{L}}^+$, respectively, and the negative-group loss of REAL and LACL-GUI by $\mathcal{L}^-_{\mathrm{REAL}}$ and $\tilde{\mathcal{L}}^-$, respectively.

We first consider the positive group. The original REAL positive loss (in log-sum-exp form) is

$$\mathcal{L}^+_{\mathrm{REAL}} = \log\Big(1 + \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda}\Big). \tag{17}$$

After perturbation it becomes

$$\tilde{\mathcal{L}}^+ = \log\Big(1 + \sum_{\tau^j \in \mathcal{T}^+} e^{-(\bar{s}_\theta(\tau^j) + \epsilon g(\tau^j))/\lambda}\Big). \tag{18}$$

where $g(\tau^j)) = |\tau^j| - \mu_w$.

Using the first-order Taylor expansion

$$e^{-\epsilon g(\tau^j)/\lambda} = 1 - \frac{\epsilon g(\tau^j)}{\lambda} + \mathcal{O}(\epsilon^2), \tag{19}$$

we can expand the sum as

$$\begin{aligned}
&\sum_{\tau^j \in \mathcal{T}^+} e^{-(\bar{s}_\theta(\tau^j) + \epsilon g(\tau^j))/\lambda} \\
&= \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda} \cdot e^{-\epsilon g(\tau^j)/\lambda} \\
&= \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda} \left(1 - \frac{\epsilon g(\tau^j)}{\lambda} + \mathcal{O}(\epsilon^2)\right) \\
&= \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda} - \frac{\epsilon}{\lambda} \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda}\, g(\tau^j) \\
&\quad + \mathcal{O}(\epsilon^2).
\end{aligned}$$

The key observation is that the coefficient of the first-order term vanishes. By Lemma 1, we have the weighted zero-mean property

$$\sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda}\, g(\tau^j) = 0, \tag{20}$$

Consequently, the linear term in ϵ is exactly zero, and we obtain

$$\sum_{\tau^j \in \mathcal{T}^+} e^{-(\bar{s}_\theta(\tau^j)+\epsilon g(\tau^j))/\lambda} = \sum_{\tau^j \in \mathcal{T}^+} e^{-\bar{s}_\theta(\tau^j)/\lambda} + \mathcal{O}(\epsilon^2). \tag{21}$$

Substituting this back into the log-sum-exp loss immediately yields

$$\tilde{\mathcal{L}}^+ = \mathcal{L}^+_{\mathrm{REAL}} + \mathcal{O}(\epsilon^2). \tag{22}$$

We now turn to the negative group. The original REAL negative loss is

$$\mathcal{L}^-_{\mathrm{REAL}} = \log\Big(1 + \sum_{\tau^j \in \mathcal{T}^-} e^{\bar{s}_\theta(\tau^j)/\lambda}\Big). \tag{23}$$

After applying the perturbation $\hat{s}_\theta(\tau^j) = \bar{s}_\theta(\tau^j) + \epsilon y(\tau^j)$, where $y(\tau^j) = E(\tau^j) - \mu_e$, it becomes

$$\tilde{\mathcal{L}}^- = \log\Big(1 + \sum_{\tau^j \in \mathcal{T}^-} e^{(\bar{s}_\theta(\tau^j)+\epsilon y(\tau^j))/\lambda}\Big). \tag{24}$$

Using the first-order expansion

$$e^{\epsilon y(\tau^j)/\lambda} = 1 + \frac{\epsilon y(\tau^j)}{\lambda} + \mathcal{O}(\epsilon^2) \tag{25}$$

and following the same algebraic steps as in the positive group, we obtain

$$\begin{aligned}
&\sum_{\tau^j \in \mathcal{T}^-} e^{(\bar{s}_\theta(\tau^j)+\epsilon y(\tau^j))/\lambda} \\
&\quad = \sum_{\tau^j \in \mathcal{T}^-} e^{\bar{s}_\theta(\tau^j)/\lambda} + \frac{\epsilon}{\lambda} \sum_{\tau^j \in \mathcal{T}^-} e^{\bar{s}_\theta(\tau^j)/\lambda}\, y(\tau^j) + \mathcal{O}(\epsilon^2).
\end{aligned}$$

By the weighted zero-mean property in Lemma 1 , the coefficient of the linear term vanishes. Therefore,

$$\tilde{\mathcal{L}}^- = \mathcal{L}^-_{\mathrm{REAL}} + \mathcal{O}(\epsilon^2). \tag{26}$$

Combining the results for both groups, we conclude that

$$\mathcal{L}_{\text{LACL-GUI}} = \mathcal{L}_{\mathrm{REAL}} + \mathcal{O}(\epsilon^2). \tag{27}$$

Thus the proposed perturbations preserve the REAL objective up to first order while redistributing optimization emphasis in higher-order within each outcome group. □

A.2 Experiment Details

A.2.1 OSWorld Benchmark Settings

We evaluate on a subset of 222 tasks selected from the full OSWorld benchmark of 369 tasks. All experiments are conducted in Baidu's RLE virtual machine environment. We exclude *Chrome* and *Multi-app* tasks. This task filtering is motivated by the need for stable training and evaluation conditions. First, *Chrome* tasks require reliable external network connectivity, which is unstable in our RLE environment and frequently leads to non-reproducible outcomes. Second, *Multi-app* tasks often depend on external asset files hosted on the official huggingface cache (for example, task `3f05f3b9-29ba-4b6b-95aa-2204697ffc06`, which requires the audio file "Zhou Xuan - Nights in Shanghai.mp3"). These assets were frequently missing or incomplete in our setup, resulting in systematic environment failures rather than genuine agent errors. Both issues introduce significant instability into both training and testing, and the corresponding tasks are therefore excluded.

After removing the affected *Chrome* and *Multi-app* tasks, the remaining 222 tasks span eight application domains. Their distribution is summarized in Table 3.

Domain	Number of Tasks
Gimp	26
LibreOffice Calc	47
LibreOffice Impress	47
LibreOffice Writer	23
OS	24
Thunderbird	15
VLC	17
VS Code	23
Total	222

Table 3: Distribution of the 222 selected OSWorld tasks across application domains.

Domain	Base	LACL-GUI	REAL	Rel. to REAL
Gimp	36.46	12.08	13.38	-9.7%
Calc	17.50	10.17	16.00	-36.4%
Impress	4.25	4.25	4.25	0.0%
Writer	10.73	5.36	5.27	$+1.7\%$
OS	21.67	4.25	5.50	-22.7%
Thunderbird	7.86	6.71	7.29	-8.0%
VLC	16.75	10.75	5.00	$+115.0\%$
VS Code	30.67	13.44	15.56	-13.6%
Average	19.84	8.19	9.13	-10.3%

Table 4: Average trajectory lengths on the 70 tasks successfully completed by all three methods. The last column shows the relative change of LACL-GUI with respect to REAL.

A.2.2 Effect of Length-aware Preference Modeling

To further examine the effect of length-aware preference modeling, we conduct a fine-grained analysis of average trajectory lengths on tasks where the base model Qwen3-VL-8B-Thinking, REAL, and LACL-GUI all achieve the same outcome. This isolates the impact of trajectory preference from differences in task success rate. Results are reported separately for successful and failed trajectories.

Table 4 presents the average trajectory lengths on the 70 tasks that are successfully completed by all three methods. LACL-GUI reduces the overall average length from 9.13 steps (REAL) to 8.19 steps (a 10.3% reduction). The reduction holds in the majority of application domains (*Gimp*, *LibreOffice Calc*, *OS*, *Thunderbird*, and *VS Code*), although the effect is not uniform: *LibreOffice Writer* shows a slight increase and *VLC* exhibits a clear increase. These results largely support the design motivation that, under identical initial states and goals, shorter successful executions generally correspond to higher-quality solutions with fewer redundant interactions.

Table 5 reports the corresponding statistics on the 93 tasks where all three methods fail. LACL-GUI again yields a shorter average trajectory (26.72 steps) compared with REAL (29.87 steps), corresponding to a 10.5% reduction. Length reduction is observed in six out of eight domains, with particularly large improvements in *Gimp* and *OS*. Exceptions occur in *LibreOffice Writer* and *VS Code*, where LACL-GUI produces longer failed trajectories. Because length modulation is applied only within the negative group via divergence-aware erroneous-step modeling, LACL-GUI tends to suppress prolonged ineffective failures while still preserving near-success trajectories as informative learning signals.

A.3 Real-world Trajectories

To provide an intuitive illustration of why length-aware preference within successful trajectories and divergence-aware modeling within failed trajectories are beneficial, we present concrete trajectory examples from the same *Gimp* task:

Please help change GIMP's theme from dark to light.

Domain	Base	LACL-GUI	REAL	Rel. to REAL
Gimp	33.80	5.20	25.00	-79.2%
Calc	39.91	30.12	33.97	-11.3%
Impress	30.48	27.86	31.24	-10.8%
Writer	37.83	46.67	42.67	$+9.4\%$
OS	35.67	10.67	22.67	-52.9%
Thunderbird	50.00	26.75	27.75	-3.6%
VLC	33.14	13.14	15.57	-15.6%
VS Code	36.86	25.14	16.57	$+51.7\%$
Average	36.06	26.72	29.87	-10.5%

Table 5: Average trajectory lengths on the 93 tasks where all three methods fail. The last column shows the relative change of LACL-GUI with respect to REAL.

(task ID: `7767eef2-56a3-4cea-8c9f-48c070c7d65b`). This task requires navigating *Gimp*'s preference menus to switch the interface theme. Although the goal is relatively straightforward, different rollouts can exhibit substantial variation in efficiency and in how closely failed attempts approach a successful solution. We therefore examine both successful and failed trajectories produced for this task to highlight the fine-grained quality differences that outcome-level supervision alone cannot capture.

Positive Group: Efficient vs. Redundant Successful Trajectories. Although both trajectories receive the same success reward, successful executions can exhibit substantially different levels of efficiency. We illustrate this phenomenon using the *Gimp* theme-switching task. The following example compares a concise successful trajectory generated by LACL-GUI-8B (6 steps) with a longer successful trajectory generated by REAL (27 steps).

The 6-step trajectory follows the optimal interaction path: it directly opens the Edit menu, enters Preferences, selects the Theme option, switches to Light mode, and confirms the change. No exploratory or recovery actions are involved.

```
Step 1:  Click on the "Edit" menu in the top menu bar.
pyautogui.click(124, 72)

Step 2:  Click on the "Preferences" option in the Edit menu.
pyautogui.click(180, 640)

Step 3:  Click on the "Theme" option under the Interface section.
pyautogui.click(157, 667)

Step 4:  Select the "Light" theme option.
pyautogui.click(363, 622)

Step 5:  Click the "OK" button to apply the theme.
pyautogui.click(832, 867)

Step 6:  Terminate (task completed successfully).
DONE
```

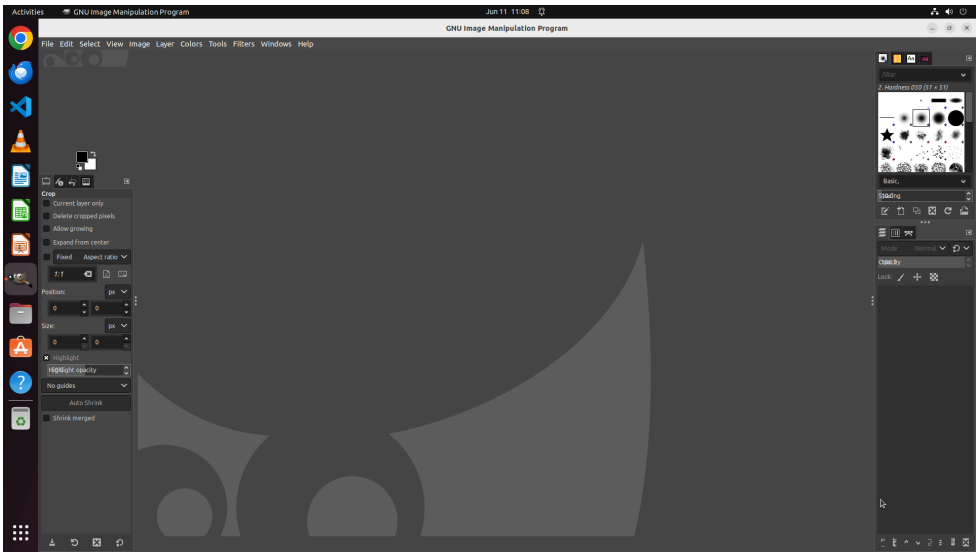

Figure 4: LACL-GUI Step 1 (Success)

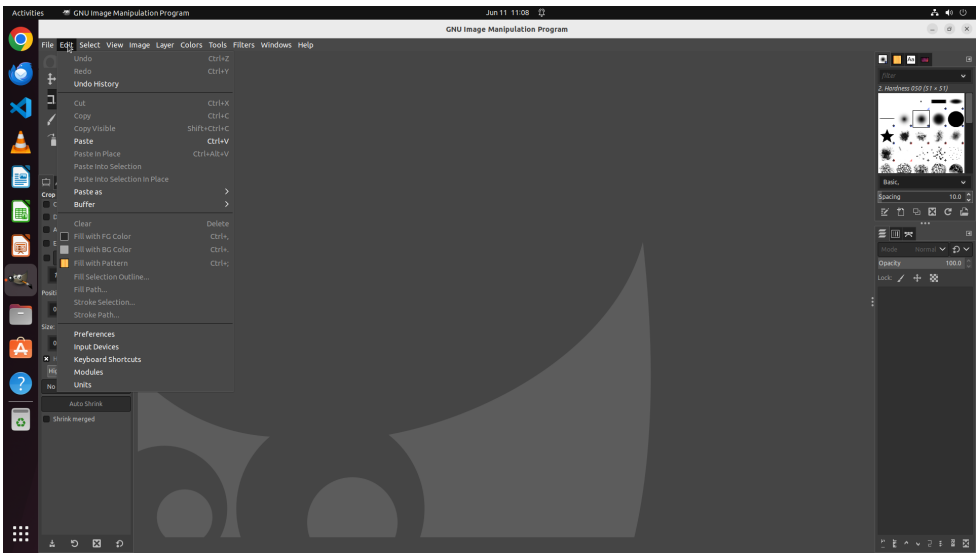

Figure 5: LACL-GUI Step 2 (Success)

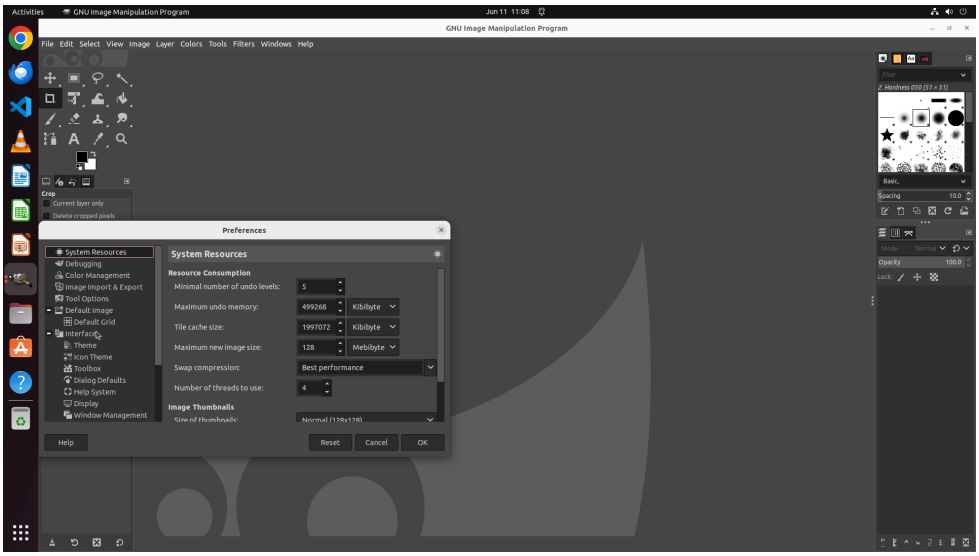

Figure 6: LACL-GUI Step 3 (Success)

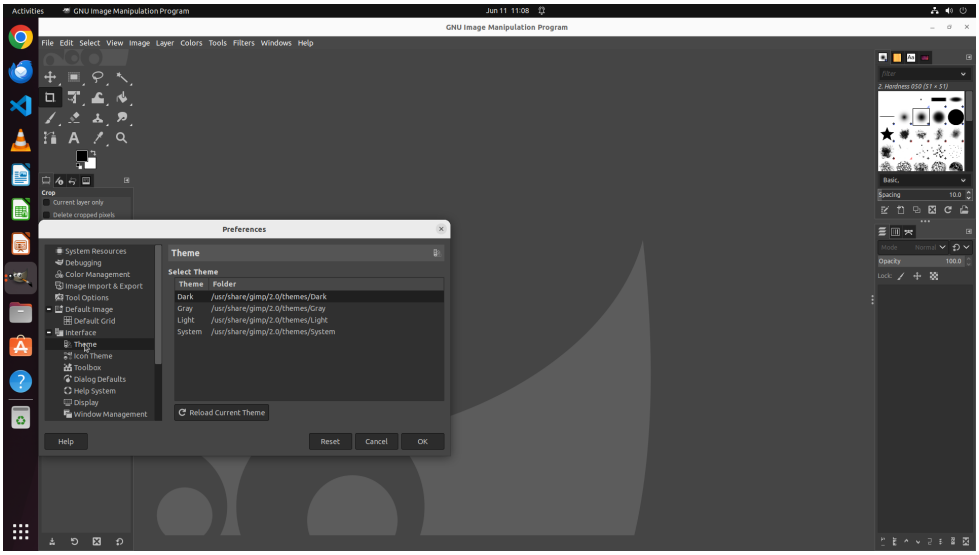

Figure 7: LACL-GUI Step 4 (Success)

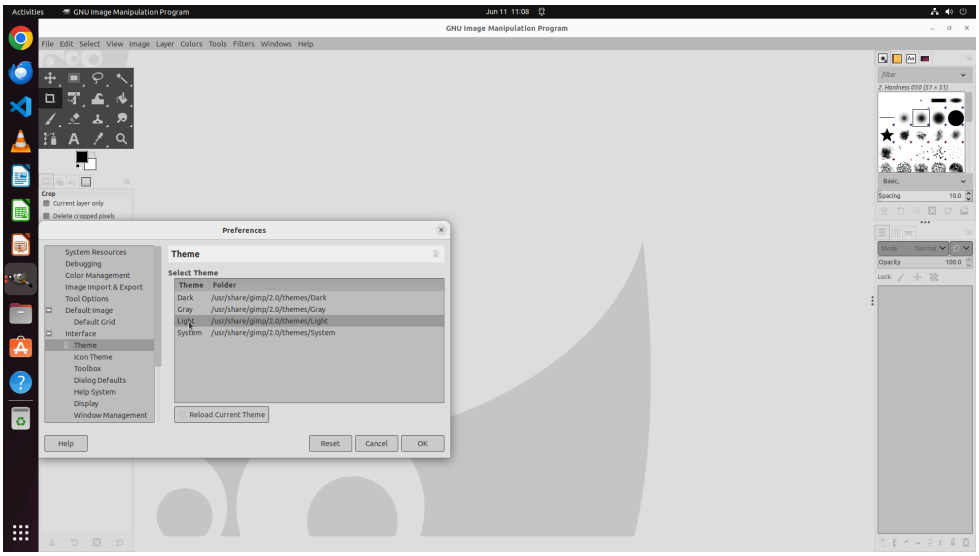

Figure 8: LACL-GUI Step 5 (Success)

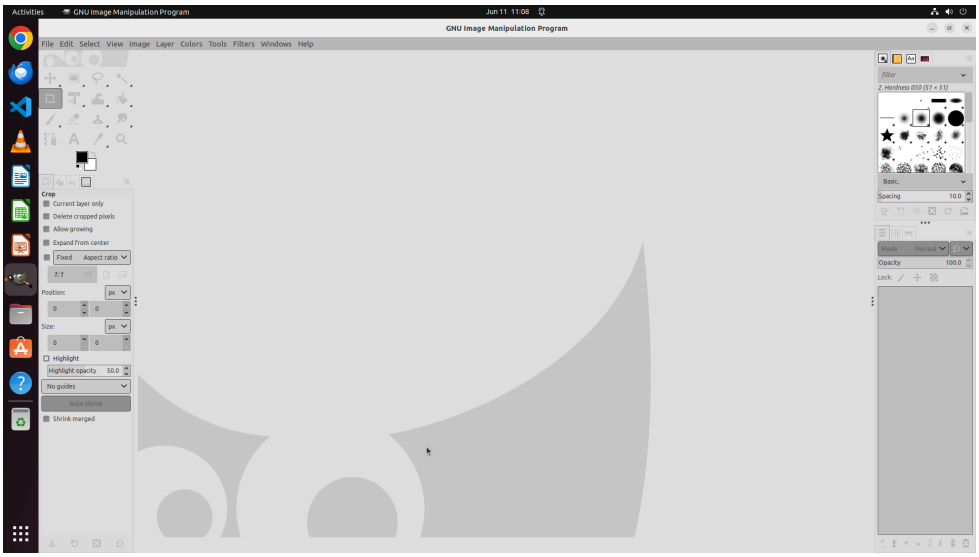

Figure 9: LACL-GUI Step 6 (Success)

In contrast, the 27-step trajectory produced by Qwen3-VL-8B-REAL also successfully completes the task but contains extensive unnecessary interactions:

```
Step 1-4:  Navigate to the Theme panel correctly.

Step 5-15:  Repeatedly click the "Theme" item in the left sidebar.
pyautogui.click(163, 667)   (repeated)

Step 16:  Click the "Theme" item again.

Step 17-24:  Continue clicking the same screen location after the UI
state has changed.
pyautogui.click(167, 667)   (repeated)

Step 25:  Select the "Light" theme option.
pyautogui.click(359, 620)

Step 26:  Click the "OK" button to apply the change.
pyautogui.click(830, 859)

Step 27:  Terminate (task completed successfully).
DONE
```

The long trajectory reveals two types of inefficiency. First, from Step 5 to Step 15, the agent repeatedly executes a valid action without making progress, producing redundant interactions. Second, from Step 17 to Step 24, the agent continues clicking the previous target location even after the interface state has changed, resulting in erroneous actions caused by outdated state understanding.

These additional steps do not contribute to task completion and are not desirable behaviors to reinforce, despite the trajectory receiving the same binary success reward. Therefore, length-aware positive preference in LACL-GUI assigns stronger preference to concise successful trajectories, encouraging the policy to learn efficient solutions rather than merely optimizing for eventual task completion.

The screenshots below highlight representative failure patterns in the long successful trajectory. Step 5 demonstrates redundant interactions where the agent repeatedly selects an already active option, while Step 17 illustrates incorrect repeated clicking after the UI state has changed.

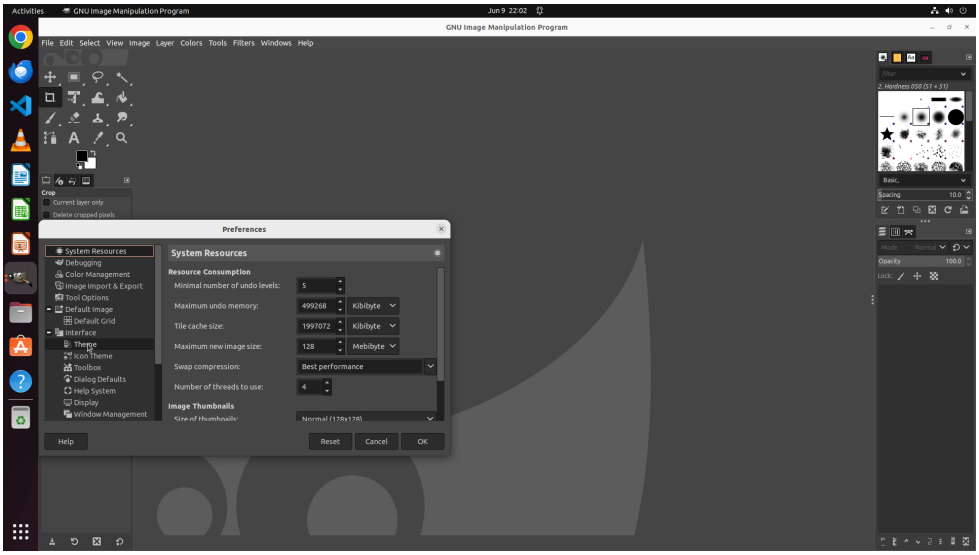

Figure 10: A redundant interaction in the long successful trajectory (Step 5).

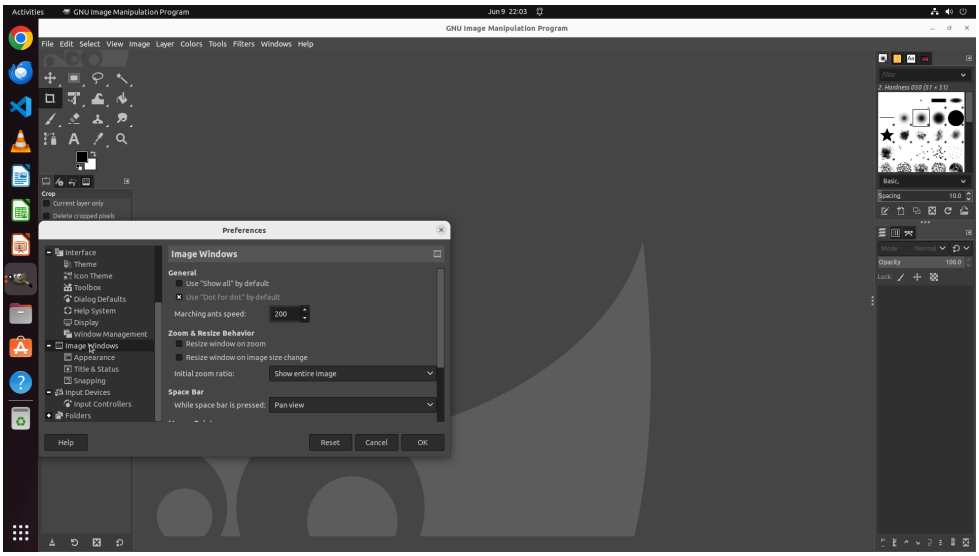

Figure 11: An erroneous repeated action after the UI state changes (Step 17).

Negative Group: Different Failure Patterns Beyond Binary Rewards. While successful trajectories can differ in efficiency, failed trajectories can also exhibit substantially different qualities despite receiving the same binary reward. We illustrate this phenomenon using two failed trajectories generated by Qwen3-VL-8B-Thinking on the same *Gimp* theme-switching task. Both trajectories obtain a reward of 0, but they exhibit different failure behaviors after deviating from the successful execution path.

For reference, the successful trajectory generated by LACL-GUI-8B changes the theme by selecting the "Theme" option at Step 4, followed by choosing the "Light" theme and confirming the change. Both failed trajectories diverge at this critical step: although the agent reaches the Preferences dialog and interacts with the Theme option, it fails to correctly complete the theme switch.

The first failure terminates prematurely after only four steps. The agent correctly navigates to the Theme option but incorrectly assumes that the task has been completed without verifying whether the desired state has been achieved.

```
Step 1:  Click the "Edit" menu in the top menu bar.
pyautogui.click(124, 72)

Step 2:  Click the "Preferences" option in the Edit menu.
pyautogui.click(180, 640)

Step 3:  Click the "Theme" option under the Interface section.
pyautogui.click(157, 669)

Step 4:  Terminate (task incorrectly considered completed).
DONE
```

In contrast, the second failure continues interacting after the same divergence point. Although the agent performs additional actions, these steps do not recover the correct execution path and only introduce ineffective exploration before termination.

```
Step 1-3:  Navigate to the Theme option following the correct menu
hierarchy.

Step 4:  Click the "Theme" option again in the Preferences dialog.
pyautogui.click(167, 667)

Step 5:  Click the "OK" button without successfully applying the Light
theme.
pyautogui.click(830, 859)

Step 6:  Terminate (task incorrectly considered completed).
DONE
```

These examples reveal two common failure modes that are indistinguishable under binary rewards. The first corresponds to premature termination (result in early divergence from reference successful trajectory) after an incomplete state transition, while the second corresponds to unnecessary post-divergence interactions that increase trajectory length without improving task completion. Therefore, evaluating failed trajectories requires finer-grained preference modeling beyond success or failure labels. By incorporating trajectory structure and ineffective actions, LACL-GUI assigns more informative preferences within negative groups, encouraging the policy to avoid both early mistakes and unproductive exploration.

The screenshots below show representative failure cases. The first example demonstrates premature termination after the agent reaches the Theme option, while the second illustrates unnecessary post-divergence interactions.

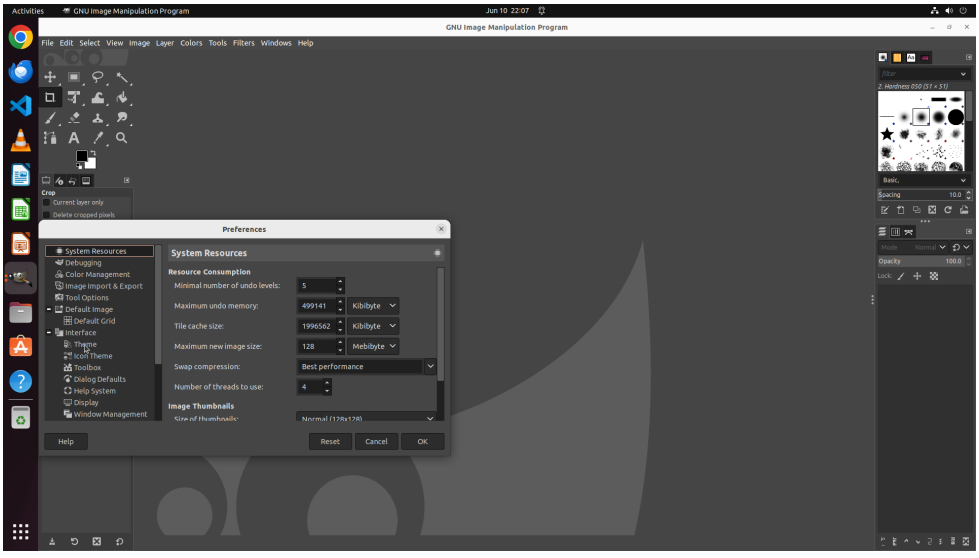

Figure 12: Premature termination after an incomplete state transition (Step 4 in 4-step Failed Trajectory).

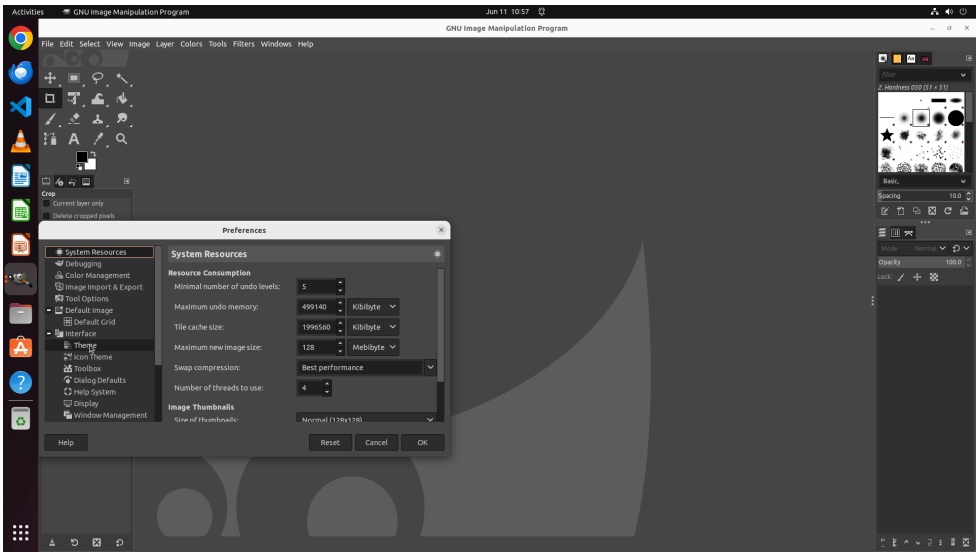

Figure 13: Additional ineffective actions after divergence from the successful execution (Step 5 in 6-step Failed Trajectory).

These failure cases also demonstrate the motivation of the negative-group modulation in LACL-GUI. For each failed trajectory, LACL-GUI first identifies the divergence point from the reference successful trajectory using observation similarity and then estimates the number of erroneous steps after divergence. Although both trajectories deviate at the same decision point in this example, they exhibit different error severities: the 4-step trajectory terminates immediately after divergence, whereas the 6-step trajectory continues with additional ineffective interactions. Consequently, the latter receives a larger erroneous-step count and stronger suppression through the perturbed logits in Eq. (9), preventing the policy from reinforcing long unsuccessful exploration. Meanwhile, this fine-grained modeling preserves informative failures that remain closer to successful execution, instead of treating all failed trajectories uniformly as in REAL.