TEST-TIME SELF-EVOLVING GUI VISUAL GROUNDING VIA REFLECTION-GUIDED ON-POLICY SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI Visual Grounding is a fundamental capability for GUI agents. Existing models typically freeze their parameters after deployment, limiting their ability to adapt to unseen interfaces. Although recent methods attempt to adapt models via test-time reinforcement learning, they cannot reflect upon failed exploration. To overcome this, we propose a Test-Time Self-Evolving framework that enables models to improve after deployment without human-annotated ground truth. It constructs a closed-loop of Exploration, Evaluation, Reflection, and Internalization. Specifically, the agent first explores unseen interfaces by predicting grounding coordinates for given instructions. To evaluate these explorations, we introduce an MLLM-based Reflector to assess the generated results and provide the corresponding reasoning reflections. To internalize reflection knowledge into the model weights, we propose Reflection-Guided On-Policy Self-Distillation, which translates high-level reasoning into dense token-level supervision via a conditioned self-teacher. Furthermore, we design a Contrastive Calibration method to prevent incorrect auto-regressive prefixes from corrupting the supervisory signals during failed explorations. Extensive experiments across six benchmarks demonstrate our framework's effectiveness, achieving an average accuracy improvement of 7.4% over the base model. To the best of our knowledge, this is the first work to successfully exploit on-policy self-distillation for test-time adaptation in GUI visual grounding. By filling the gap in post-deployment adaptation, our framework completes the self-evolving capability of GUI agents. The code will be released.

1 INTRODUCTION

GUI agents capable of navigating and interacting with digital environments have recently emerged as a promising direction for human-computer interaction (Xie et al., 2025). As a fundamental capability of GUI agents, GUI Visual Grounding aims to localize UI elements according to natural-language instructions. Recent advances have substantially improved grounding performance through large-scale supervised fine-tuning (Qin et al., 2025; Cheng et al., 2024), specialized action heads (Wu et al., 2025; Lin et al., 2025), and reinforcement learning (Luo et al., 2025a; Yuan et al., 2025).

Despite these advances, existing GUI grounding models remain static after deployment, with model parameters frozen once training is completed. Consequently, when encountering unseen applications or interface layouts, their performance cannot improve through interaction. A recent test-time adaptation method (Du et al., 2026) attempts to alleviate this limitation by optimizing the model online with reinforcement learning. This method relies on sparse scalar rewards that merely indicate whether an exploration succeeds. Such feedback provides little information about *why* a prediction fails or how the model should correct it.

A natural solution is to train the model with the reflection on its own explorations. Rather than merely assigning a success or failure signal, reflection explains why a prediction is incorrect by identifying the intended UI element, examining the predicted location, and diagnosing the failure. Such reasoning provides richer supervisory information than scalar rewards. However, existing reinforcement learning algorithms cannot directly leverage these textual reflections based on scalar rewards.

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.

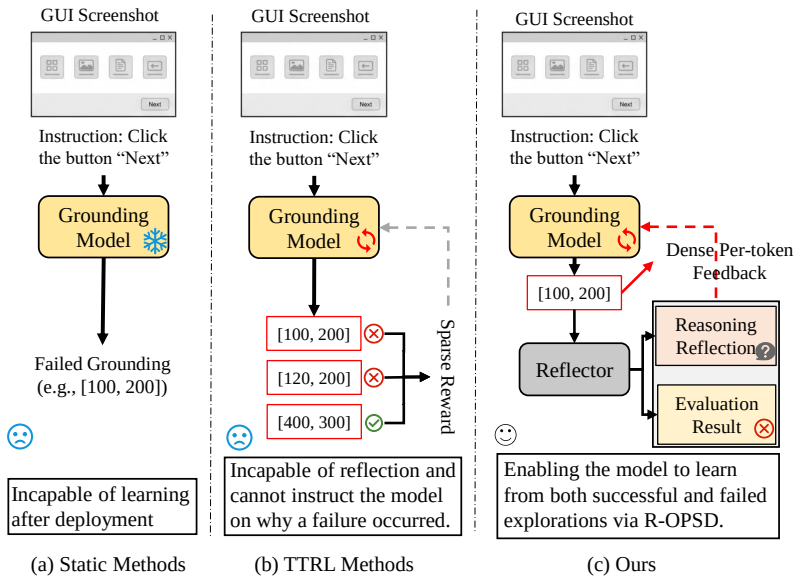


Figure 1: (a) Static methods operate on frozen parameters after deployment and cannot learn from exploration. (b) TTRL Methods adapt the model using sparse scalar rewards, but lack a mechanism to reason about failures and internalize textual reflections. (c) Test-Time Self-Evolving introduces a Reflector to provide detailed textual reflections, which are translated into dense token-level feedback via R-OPSD, enabling the model to self-evolve.

To overcome this, we propose a Test-Time Self-Evolving framework. It constructs a closed-loop that equips the agent with four core capabilities: **Exploration** in unknown interfaces, **Evaluation** of generated coordinates, **Reflection** upon evaluation results, and **Internalization** of these reflections into its parameters. Specifically, upon deployment, the agent first explores the interface with instructions by predicting grounding coordinates. To evaluate this exploration, we introduce an MLLM-based Reflector that estimates whether the prediction is consistent with the instruction while simultaneously generating a reflection explaining its evaluation.

The remaining challenge is how to internalize the reflection into the model parameters. Unlike scalar rewards, textual reflections cannot be directly exploited by policy optimization algorithms. To bridge this gap, we propose Reflection-Guided On-Policy Self-Distillation (R-OPSD). Instead of learning from scalar rewards, R-OPSD conditions a self-teacher on the Reflector's evaluation results and reflections, converting high-level reasoning into dense token-level supervision over the generated coordinate tokens.

However, this dense feedback encounters a bottleneck in the auto-regressive generation of coordinates. During failed explorations, the model predicts incorrect tokens. If the teacher model conditions its subsequent supervision on these prefixes, the supervisory signals gradually become unreliable. To mitigate this issue, we propose a Contrastive Calibration Method. It treats a failed exploration as successful by using an inverse-prompted student. The trajectory is sampled from the model's own policy, conditioning it on a success prompt amplifies its confidence in this specific sequence. Consequently, at the initial error step, the inverse student assigns a high probability to the incorrect token, while the reflection-guided teacher yields a lower probability. This discrepancy produces a negative advantage that suppresses the initial error. As generation continues, the corrupted prefix dominates both models, aligning their output distributions. This decays the advantage to almost zero, preventing corrupted feedback from being internalized.

Extensive experiments across ScreenSpot and ScreenSpot-v2 (Cheng et al., 2024), ScreenSpot-Pro (Li et al., 2025), MMBench-GUI (Wang et al., 2025), OSWorld-G and OSWorld-G-Refine (Xie et al., 2025) demonstrate the effectiveness of our framework. By enabling self-evolving without ground truth annotations, our method achieves an average accuracy improvement of 7.4% over the base model. Furthermore, compared to the recent test-time reinforcement learning method GUI-RCPO, our method achieves an average performance gain up to 7.7%.

In summary, our main contributions are three-fold: (a) We introduce a Test-Time Self-Evolving framework, which empowers grounding agents with exploration, evaluation, reflection, and parameter internalization in unseen environments. (b) We propose Reflection-Guided On-Policy Self-Distillation, which converts textual reflections into dense token-level supervision, together with a Contrastive Calibration method that alleviates corrupted supervision caused by incorrect prefixes. (c) Extensive experiments and ablations validate our framework. This is the first work to applying OPSD in test-time scenarios of GUI grounding, establishing a promising direction.

2 Related Works

GUI Visual Grounding. GUI visual grounding locates UI elements based on natural language instructions. Recent advancements span three paradigms. **Supervised fine-tuning methods** establish baseline alignment using large-scale datasets (Qin et al., 2025; Cheng et al., 2024; Gou et al., 2024). To enhance visual perception, models selectively extract UI-related visual tokens (Lin et al., 2025; Ouyang et al., 2026) or employ attention-based action heads to bypass direct coordinate generation (Wu et al., 2025). **Reinforcement learning methods** further enhance the performance. Recent works adapt GRPO (Shao et al., 2024) with rule-based coordinate rewards (Luo et al., 2025a; Lu et al., 2026), or design complex reward functions incorporating attention maps, box sizes, and bounding-box distributions (Yuan et al., 2025; Zhou et al., 2025; Tang et al., 2026). **Test-time scaling methods** allocate additional inference compute to enhance performance. RegionFocus (Luo et al., 2025b) dynamically zooms into the image for coarse-to-fine UI grounding, a process similarly guided by attention maps in ZoomUI (Liu et al., 2026). Additionally, GUI-RC (Du et al., 2026) aggregates multiple predictions via region consistency, further refining the model with test-time reinforcement learning (Zuo et al., 2026).

On-policy Distillation. To internalize reasoning capabilities into model weights, on-policy distillation (OPD) trains the student model with token-level feedback on its own generation trajectories. Typically, this feedback is derived either from a stronger teacher model (Gu et al., 2024; Agarwal et al., 2024) or a model conditioned on privileged information (Zhang et al., 2026a; Zhao et al., 2026). However, incorrect prefixes generated by the student can degrade the quality of the feedback. To address this issue, ESR (Ziheng et al., 2026) proposes an early stopping rollout mechanism, while EOPD (Jin et al., 2026) assesses the reliability of the teacher model using token entropy. Alternatively, RLSD (Yang et al., 2026) and RLCSD (Pan et al., 2026) leverage token-level feedback to only determine update magnitudes. The potential of OPD has also been explored across various vision-language tasks, such as Video Temporal Grounding (Li et al., 2026), Visual Question Answering (Yuan et al., 2026), and GUI Visual Grounding (Zhang et al., 2026b).

Unlike test-time reinforcement learning methods (Du et al., 2026; Zuo et al., 2026) that rely on sparse scalar rewards and fail to leverage reflection reasoning, our framework empowers the model to learn from rich reflections via R-OPSD. Furthermore, Contrastive Calibration effectively migrates the corrupted supervision issue caused by incorrect prefixes, ensuring that the model can safely internalize knowledge from failed explorations.

3 Methodology

3.1 Overview

Given a screenshot I and a language instruction L, the GUI grounding model π_{G} is required to locate the corresponding UI element $B = \pi_{\mathrm{G}}(I, L)$, where B denotes the bounding box $[x_1, y_1, x_2, y_2]$ or the center point (x, y) of the target element. Existing GUI grounding models typically remain static after deployment, with model parameters frozen once training is completed. Consequently, when deployed on unseen applications, the model cannot improve from its own interaction experience.

To enable post-deployment adaptation without human annotations, we propose a Test-Time Self-Evolving framework that constructs a closed-loop consisting of four stages: **Exploration** in unknown interfaces, **Evaluation** of its generated coordinates, **Reflection** upon its evaluation result, and **Internalization** of these reflections into its parameters:

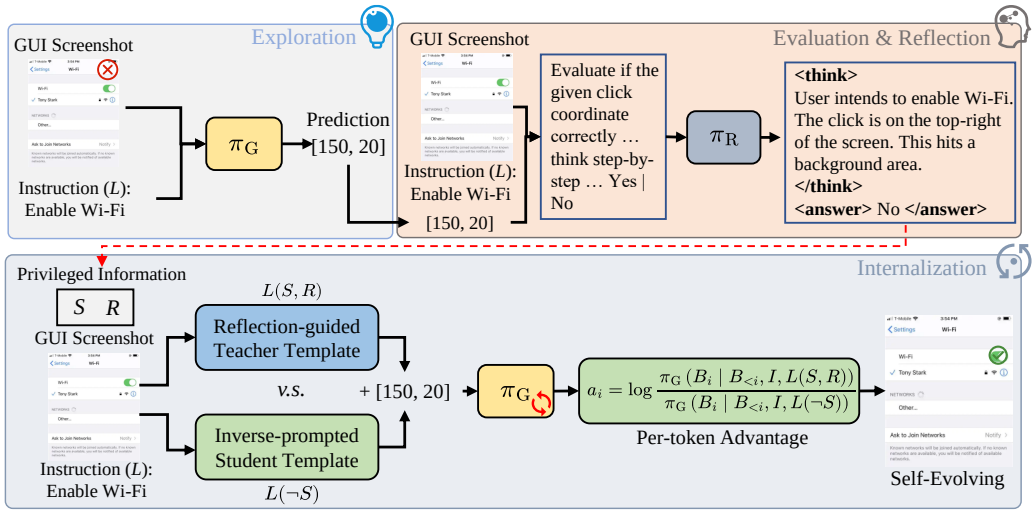


Figure 2: **Test-Time Self-Evolving Framework** consists of four core stages: Exploration, Evaluation, Reflection, and Internalization. During test-time adaptation, π_G explores the environment to predict coordinates. The Reflector π_R then assesses the prediction to generate a binary score S and a step-by-step reasoning R. The reflection knowledge is internalized via R-OPSD to update the grounding policy. To optimize memory efficiency, both π_G and π_R share a base model and we alternate between the roles of π_G and π_R by switching the active LoRA adapters.

- **Exploration:** The GUI grounding model π_G explores the unseen interface, *i.e.*, given the UI screenshot I and instruction L, $B = \pi_G(I, L)$.

- **Evaluation & Reflection:** To assess the exploration in the absence of ground truth, we introduce a Reflector π_R. It takes the screenshot, instruction, and predicted coordinates as input to output an evaluation score S and a detailed reasoning process R,

$$S, R = \pi_R(I, L, B), \tag{1}$$

  where $S \in \{0, 1\}$ indicates whether the current exploration is successful, and R represents the reasoning behind this evaluation. Subsequently, S and R are utilized to guide the internalization.

- **Internalization:** The reflection knowledge is then internalized into π_G through Reflection-Guided On-Policy Self-Distillation. Specifically, the evaluation score and reasoning process are leveraged as *privileged information* to construct a self-teacher that provides per-token feedback on the generated coordinates. Furthermore, to alleviate failed explorations from corrupting the feedback, we propose a **Contrastive Calibration Method**.

In the following parts, we will first introduce the Evaluation & Reflection stages, and further propose the Internalization stage. The overall framework is illustrated in Fig.2.

3.2 Evaluation and Reflection with Reflector

During test-time adaptation, ground-truth annotations are unavailable. Therefore, the model requires an alternative mechanism to estimate whether its exploration is successful. To this end, we introduce an MLLM-based Reflector π_R. Its role is to evaluate the alignment between the predicted coordinates B and the language instruction L within the context of the visual screenshot I. Besides producing an estimated evaluation result, it also generates a reflection explaining the reasoning behind its judgment. This reflection serves as privileged information for the subsequent internalization stage.

Inference. To achieve this goal, we design a prompt template that forces the Reflector to engage in step-by-step reasoning before outputting a final result:

Reflector Prompt Template

Evaluate if the given click coordinate correctly executes the user instruction on the provided UI image. Answer with Yes or No.
Instruction: $\{L\}$
Click Coordinate: $\{B\}$
Before providing your final answer, you must think step-by-step. Address the following in your reasoning:
1. Intent Analysis: What specific UI element (text, icon, button) does the instruction intend to interact with?
2. Coordinate Grounding: Look at the exact location of the Click Coordinate [x,y] on the UI. What specific element or visual feature is located at that coordinate?
3. Verification: Does the element at the given coordinate match the intended element from the instruction? Is it the correct target, or did it hit the background, a different element, or an empty space?
4. Conclusion: Explicitly state whether the verification passed (Yes) or failed (No).
Output the thinking process in <think></think>and final answer in <answer></answer>tags.
You must strictly format your output as follows:
<think>Your step-by-step reasoning and conclusion here </think>
<answer>Yes|No </answer>

Once receiving the generated response, we parse the output according to the specified tags to obtain R and S. Specifically, the text extracted from `<think>...</think>` and `<answer>...</answer>` tags serves as the reflection R and S, respectively.

Training. To improve the reliability of the Reflector, we optimize it using GRPO (Shao et al., 2024) on an offline collected dataset. Each sample in this dataset comprises a UI screenshot I, a language instruction L, a sampled click coordinate B, and a ground-truth label $S^* \in \{0, 1\}$ indicating whether the coordinate successfully executes the instruction. The construction details of the dataset are shown in the **Appendix A.1**. We employ the format and binary rewards during training, *i.e.*, $r = r_{\text{format}} + r_{\text{binary}}$:

- r_{format} ensures the model strictly adheres to the requested output template.
- r_{binary} supervises discrimination capability:

$$r_{\text{binary}} = \begin{cases} 1, & \text{if } S = S^* \text{ and format is valid} \\ 0, & \text{if } S \neq S^* \text{ or format is invalid} \end{cases} \tag{2}$$

After training, the Reflector remains frozen throughout test-time adaptation.

3.3 Internalization with Reflection-Guided On-Policy Self-Distillation

After obtaining the evaluation result S and reflection R, the final stage is to internalize this knowledge into the grounding model π_{G}. The remaining challenge is how to transform textual reflection into effective optimization signals. Unlike reinforcement learning, which is based on scalar rewards, reflection consists of free-form natural language and therefore cannot be directly exploited by standard policy optimization algorithms. To bridge this gap, we propose Reflection-Guided On-Policy Self-Distillation (R-OPSD), which converts reflection into dense token-level supervision.

R-OPSD. Given a rollout $B \sim \pi_{\text{G}}(\cdot \mid I, L)$ generated by the policy, OPD trains the policy π_{G} on its own rollout with the token-level supervisory signals provided by a teacher model π_{T}. Following the standard policy optimization formulation, the token-level advantage is defined as:

$$a_i = \log \frac{\pi_{\text{T}}(B_i \mid B_{<i}, I, L)}{\pi_{\text{G}}(B_i \mid B_{<i}, I, L)}. \tag{3}$$

The optimization objective maximizes the sum of token-level advantages:

$$\mathcal{L}_{\text{OPD}} = -\frac{1}{T} \sum_{i=1}^{T} \text{sg}(a_i) \log \pi_{\text{G}}(B_i \mid B_{<i}, I, L), \tag{4}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator.

To internalize the knowledge from the Reflector, we construct a self-teacher by utilizing the π_{G} itself conditioned on the evaluation result S and reflection R as the privileged information, which enables

the teacher to produce a more informative conditional distribution over the generated coordinate sequence. Specifically, the input prompt $L(S, R)$ is formulated as follows,

Reflection-guided Teacher Prompt Template

If Success ($S = 1$)
The previous prediction $\{B\}$ has been verified to correctly fulfill the instruction. Outline the position corresponding to the instruction: $\{L\}$. The output should be only [x1,y1,x2,y2].

If Failure ($S = 0$)
Previous wrong prediction: $\{B\}$
Feedback: $\{R\}$
Based on the feedback, outline the position corresponding to the instruction: $\{L\}$. The output should be only [x1,y1,x2,y2].

The token-level advantage is then computed as:

$$a_i = \log \frac{\pi_{\text{G}}\left(B_i \mid B_{<i}, I, L(S, R)\right)}{\pi_{\text{G}}\left(B_i \mid B_{<i}, I, L\right)}. \tag{5}$$

Contrastive Calibration Method. MLLM-based GUI grounding is formulated as an auto-regressive generation of coordinate tokens. Performing R-OPSD on failed explorations introduces a challenge. If the teacher model conditions its subsequent feedback on these incorrect prefixes, its predicted probabilities become meaningless.

To mitigate this, we propose a **Contrastive Calibration Method (CC)** to calibrate the learning advantage based on the state of the generated prefix: it suppresses the initial incorrect token and decays the advantage to zero as the tokens progressively drift. To achieve this, we introduce an *inverse-prompted student* to contrast against the teacher. Specifically, for failed explorations, we construct a contrastive prompt $L(\neg S)$ to misinform the model that the prediction is correct,

Inverse-prompted Student Prompt Template

If Failure ($S = 0$)
Previous correct prediction: $\{B\}$
Based on the feedback, outline the position corresponding to the instruction: $\{L\}$. The output should be only [x1,y1,x2,y2].

With $L(\neg S)$, we reformulate the advantage in Eq. 5 into the contrastive calibration advantage:

$$a_i = \log \frac{\pi_{\text{G}}\left(B_i \mid B_{<i}, I, L(S, R)\right)}{\pi_{\text{G}}\left(B_i \mid B_{<i}, I, L(\neg S)\right)}. \tag{6}$$

CC is only applied to failed explorations, as successful samples avoid the incorrect prefix issue.

As shown in Fig. 4, CC addresses the incorrect prefix issue across two phases. First, when the prefix $B_{<i}$ is correct, but the current token B_i is incorrect, the inverse-prompted student under $L(\neg S)$ is misled to treat the prediction as successful, assigning a high probability to B_i. In contrast, the teacher under $L(S, R)$ leverages the reflection to predict the true target, yielding a lower probability for B_i, producing a negative advantage $a_i < 0$, driving the policy gradient to suppress this token. Second, As more incorrect tokens are generated, both models are conditioned on increasingly similar prefixes, leading to similar probabilities $\pi_{\text{G}}(\cdot \mid L(S, R)) \approx \pi_{\text{G}}(\cdot \mid L(\neg S))$, driving the advantage to almost zero ($a_i \to 0$), thereby gradually reducing the magnitude of the token-level advantage.

Direction-based Advantage Clamping. Directly applying token-level feedback of R-OPSD can sometimes destabilize training. To mitigate this, the token-level updates should be aligned with the evaluation result, *i.e.*, if a prediction is successful, its probability should be encouraged. If it fails, its probability should be suppressed. Therefore, we apply direction-based clamping to the advantage,

$$a_i = \begin{cases} \max(0, a_i), & S = 1 \\ \min(0, a_i), & S = 0 \end{cases}. \tag{7}$$

In addition, this token-level advantage can be seamlessly integrated with the query-level advantage from GRPO. Given a sampled group of size K, the GRPO reward is defined as $r_{\text{GRPO}} = r_{\text{format}} +$

$r_{\text{correctness}}$, where $r_{\text{correctness}}$ is evaluation S of the predicted coordinates. The query-level advantage $\hat{A}_{\text{GRPO}}[k]$ is computed by standardizing these rewards:

$$\hat{A}_{\text{GRPO}}[k] = \frac{r_{\text{GRPO}}[k] - \text{mean}\{r_{\text{GRPO}}[1], \ldots, r_{\text{GRPO}}[K]\}}{\text{std}\{r_{\text{GRPO}}[1], \ldots, r_{\text{GRPO}}[K]\}}. \tag{8}$$

By incorporating the token-level advantage into the advantage, the final integrated objective is:

$$\hat{A}[k] = \hat{A}_{\text{GRPO}}[k] + \lambda a_i, \tag{9}$$

where λ controls the integration strength.

Although the proposed framework relies on the Reflector to estimate grounding correctness, it does not require perfect evaluation. Instead, R-OPSD only assumes that the Reflector provides sufficiently informative evaluations and reflections to improve the conditional distribution of the teacher. As demonstrated in our experiments, this assumption is sufficient to improve grounding performance across multiple GUI benchmarks consistently. We also provide quantitative and qualitative evaluations of the Reflector in Table 5 and Fig. 5.

4 Experiments

4.1 Experimental Setup

Evaluation Benchmarks and Metrics. We conduct experiments across diverse benchmarks covering various platforms and instruction complexities: ScreenSpot and ScreenSpot-v2, ScreenSpot-Pro, MMBench-GUI, as well as OSWorld-G and OSWorld-G-Refine. We adopt *Element Accuracy* as the evaluation metric, where a prediction is correct if the predicted point falls within the bounding box of the target UI element.

Implementation Details. Our method is built upon the Qwen2.5-VL (Team, 2025) and Qwen3-VL (Bai et al., 2025). To optimize memory efficiency, both the grounding model (π_{G}) and the Reflector (π_{R}) share a base model and are fine-tuned using LoRA (Hu et al., 2022). During training, we alternate between the roles of π_{G} and π_{R} by switching the active LoRA adapters. This parameter-efficient design restricts the GPU memory to approximately 10GB for 3B/2B models and 30GB for 7B/8B models. During the rollout phase, we set the temperature to 1.0 and top-p to 0.95. For the Reflector π_{R}, we train for 1 epoch via GRPO on 10K samples from GroundCUA (Feizi et al., 2025), with a learning rate of 1×10^{-4}, a group size of 8, and a batch size of 64. For the internalization stage of π_{G}, R-OPSD integrates the GRPO advantage with a strength coefficient $\lambda = 0.2$. The model is trained for 2 epochs with a learning rate of 1×10^{-4} and a batch size of 64. To simulate post-deployment self-evolving, we leverage data from ScreenSpot-v2 or MMBench-GUI, strictly omitting their ground-truth annotations.

4.2 Main Results

The evaluation results across six GUI visual grounding benchmarks are presented in Table 1. Overall, our proposed Test-Time Self-Evolving framework consistently enhances the grounding capabilities of the base models, outperforming the existing test-time adaptation method, *e.g.*, GUI-RCPO. Our method demonstrates performance gains across different base models. When deployed on the unseen SSv2 dataset, our framework boosts the average accuracy of Qwen2.5-VL-3B from 50.2% to 57.4%. When applied to the highly capable Qwen3-VL-2B, our framework still raises the average accuracy to 69.4% (+3.7%) and 70.3% (+4.6%) when adapting on SSv2 and MMG, respectively. This validates that our framework achieves the self-evolving without requiring ground-truth.

Adapting on complex datasets like MMG frequently results in rollout groups where all test-time explorations fail. GRPO used in GUI-RCPO relies on relative rewards, they fail to optimize from failed groups, causing negative transfer. In contrast, R-OPSD leverages reflections to provide token-level feedback, guiding the model even from complete failures. Supported by CC to filter out corrupted supervision, our framework achieves performance gains (+7.4% and +4.6%) with MMG.

Table 1: GUI grounding accuracy on six benchmarks including ScreenSpot-V2 (SSv2), ScreenSpot-Pro (SSP), OSWorld-G (OSW-G), OSWorld-G_R (OSW-GR), and MMBenchGUI (MMG).

Method	Adaptation Datasets	SS	SSv2	SSP	OSW-G	OSW-GR	MMG	Avg.
GUI-specific Models								
UGround-V1-2B (Gou et al., 2024)	-	77.7	78.8	22.7	-	-	-	-
ShowUI-2B (Lin et al., 2025)	-	75.1	-	-	-	-	-	-
GUI-Actor-3B (Wu et al., 2025)	-	89.7	91.0	42.2	-	-	-	-
FocusUI-3B (Ouyang et al., 2026)	-	-	91.5	43.8	53.4	-	-	-
InfiGUI-R1-3B (Liu et al., 2025)	-	87.5	-	35.7	-	-	-	-
SE-GUI-3B (Yuan et al., 2025)	-	-	-	35.9	-	-	-	-
GUI-G^2-3B (Tang et al., 2026)	-	90.8	91.0	37.8	-	-	-	-
Test-Time Adaptation								
Qwen2.5-VL-3B (Team, 2025)	-	78.3	80.4	20.3	27.1	37.7	57.5	50.2
+ GUI-RCPO (Du et al., 2026)	SSv2	82.3	85.4	24.8	29.3	38.4	60.2	53.4+3.2
+ Ours	SSv2	**85.9**	**88.8**	**30.5**	32.6	39.7	66.6	57.4+7.2
+ GUI-RCPO (Du et al., 2026)	MMG	79.2	80.6	19.8	25.7	36.1	58.2	49.9-0.3
+ Ours	MMG	84.6	87.5	30.1	**33.7**	**41.7**	**68.2**	**57.6+7.4**
Qwen3-VL-2B (Bai et al., 2025)	-	83.5	87.6	42.8	47.3	60.8	72.2	65.7
+ GUI-RCPO (Du et al., 2026)	SSv2	84.7	89.3	46.3	51.3	61.3	72.8	67.6+1.9
+ Ours	SSv2	**88.5**	**91.6**	47.6	52.1	61.2	75.4	69.4+3.7
+ GUI-RCPO (Du et al., 2026)	MMG	84.2	86.9	42.3	46.8	61.8	71.6	65.6-0.1
+ Ours	MMG	88.2	91.5	**49.3**	**52.2**	**63.2**	**77.6**	**70.3+4.6**

Table 2: Component ablation and reward comparison using Qwen2.5-VL-3B. Column groups indicate the adaptation datasets.

(a) Component Ablation

	SSv2			MMG		
Settings	**SSv2**	**SSP**	**MMG**	**SSv2**	**SSP**	**MMG**
Base Model	80.4	20.3	57.5	80.4	20.3	57.5
(a) None	×	×	×	×	×	×
(b) CC	81.2	21.4	58.7	80.8	21.1	59.3
(c) C	79.5	18.0	57.2	78.6	18.3	57.1
(d) CC+C	84.2	24.6	61.7	82.3	23.5	61.5
(e) CC+C+Q	84.2	25.4	62.6	82.8	23.4	62.3
(f) R	×	×	×	×	×	×
(g) R+CC	84.0	26.8	62.9	84.0	27.7	64.3
(h) R+C	82.1	21.5	58.5	80.2	19.8	57.5
(i) R+CC+C	86.8	28.5	64.2	86.0	29.9	67.4
(j) R+CC+C+Q	**88.8**	**30.5**	**66.6**	**87.5**	**30.1**	**68.2**

(b) Reward Comparison

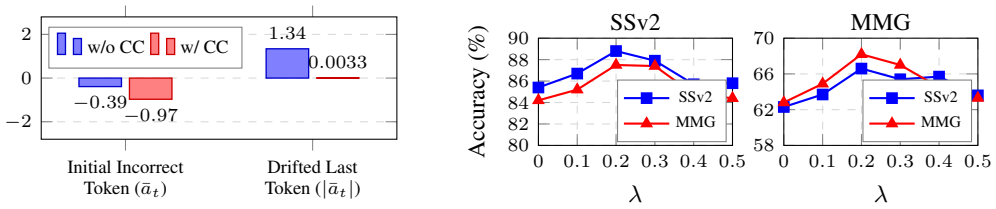
	SSv2			MMG		
Settings	**SSv2**	**SSP**	**MMG**	**SSv2**	**SSP**	**MMG**
Base Model	80.4	20.3	57.5	80.4	20.3	57.5
GUI-RCPO	85.4	24.8	60.2	80.6	19.8	58.2
Binary+$Avg.B$	84.3	24.6	61.7	79.1	21.2	57.5
IoU+$Avg.B$	84.7	22.4	61.3	80.8	20.4	59.4
GUI-G^2+$Avg.B$	83.0	21.2	59.9	78.6	19.4	55.8
Binary+S	85.4	25.9	62.3	84.2	25.7	62.8
GUI-SD+$Avg.B$	78.5	16.7	52.3	×	×	×
Ours+(S, R)	**88.8**	**30.5**	**66.6**	**87.5**	**30.1**	**68.2**

R: Reflection; **CC**: Contrastive Calibration; **C**: Direction-based advantage clamping; **Q**: Query-level advantage integration. $Avg.B$ uses aggregated bounding box predictions within a rollout group; S denotes the Reflector's evaluation results. × indicates policy collapse.

4.3 Ablation Study

Discussion of Teacher and Best-of-N performance (Table 3 and 4), qualitative and quantitative analysis of the Reflector (Table 5, Fig 5), computational overhead analysis (Table 6), scalability to larger base models (Table 7), comparisons with bounding box prompting (Table 8 and 9), and performance on the unsupervised training setting (Table 10) can be found in **Appendix**.

Effectiveness of Each Component. To validate the effectiveness of each module in our Test-Time Self-Evolving framework, we conduct ablation studies in Table 2 (a). We establish (a) as our baseline, which applies OPSD using only evaluation results as privileged information. This naive baseline suffers from catastrophic policy collapse. Comparing (d)(e) and (i)(j), incorporating Reflection yields a substantial performance gain, *e.g.*, when adapting on SSv2, Reflection boosts the accuracy from 24.6% to 28.5% on SSP. Utilizing token-level feedback in auto-regressive generation intro-



(a) Token-level advantage

(b) Sensitivity to integration strength

Figure 3: (a) Token-level advantage on failed explorations ($S = 0$): average advantage ($\bar{a}_t$) at the initial incorrect token and average absolute advantage ($|\bar{a}_t|$) at the drifted last tokens. (b) Sensitivity analysis of the integration strength λ. The subplots indicate the evaluation datasets, and the colored lines denote the adaptation datasets.

duces the incorrect prefix issue. (f) utilizes reflection but lacks CC, causing it to collapse. By incorporating CC, (g) filters out this corrupted supervision, turning the negative transfer into massive gains, reaching 64.3% on MMG. A similar substantial boost is observed in (h) *vs.* (i), confirming CC is important for learning from failed explorations. (i) demonstrates that direction-based advantage clamping prevents token-level updates from contradicting the evaluation result, providing consistent improvements. Building upon this, (j) shows the benefit of integrating the query-level advantage.

Necessity of R-OPSD over Standard RL. To demonstrate the necessity of R-OPSD, we compare it against GRPO using various scalar rewards in Table 2 (b). Following TTRL (Zuo et al., 2026), relying on averaged predicted bounding boxes (*Avg* B) as pseudo-labels suffers negative transfer on complex datasets like MMG. This occurs because when some predictions drift towards visually similar but incorrect UI elements, averaging these bounding boxes leads to completely invalid pseudo-labels. The evaluation result S from Reflector is more robust to this situation, leading to better performance. However, GRPO guided by the binary S reward remains limited by the sparsity of scalar signals and cannot instruct the model on how to correct specific coordinate tokens.

Furthermore, to verify that our gains stem from the reflection design rather than OPSD, we evaluate GUI-SD (Zhang et al., 2026b), which draws the bounding box in the image as the privileged information. Its severe performance degradation confirms that drawing invalid pseudo-labels directly corrupts the visual context. In contrast, R-OPSD leverages both the evaluation result and the detailed textual reflection (S, R), achieving the highest accuracy.

Visualization of Token-level Advantage. To validate CC, we analyze token-level advantage during failed explorations by calculating the average advantage ($\bar{a}_t$) at the initial incorrect token and the average absolute advantage ($|\bar{a}_t|$) at the drifted last token. As shown in Fig. 3 (a), without CC, R-OPSD applies a marginal penalty at the initial error and generates corrupted advantages $|\bar{a}_t| = 1.34$ on drifted tokens. In contrast, CC generates a strong negative penalty at the initial incorrect token. As tokens progressively drift, the update magnitude decays towards zero, preventing the internalization of corrupted supervision.

Sensitivity Analysis of Integration Strength. We investigate the impact of λ, which controls the magnitude of the distillation advantage during policy updates. As shown in Fig. 3 (b), $\lambda = 0.2$ leads to the best performance, regardless of the adaptation and evaluation datasets.

5 Conclusion

In this paper, we introduce a Test-Time Self-Evolving framework for GUI visual grounding that enables post-deployment adaptation without requiring ground truth annotations. Our framework introduces a closed-loop of Exploration, Evaluation, Reflection, and Internalization, allowing the model to learn from both the outcomes of its explorations and the reasons behind its successes and failures. To internalize such reflection knowledge, we propose R-OPSD, which transforms natural-language reflections into token-level supervision through a self-teacher conditioned on privileged information. Furthermore, we introduce a Contrastive Calibration method to alleviate corrupted supervision caused by incorrect auto-regressive prefixes during failed explorations. Extensive experiments across diverse benchmarks and ablations demonstrate that our framework can improve the performance of the model in a self-evolving manner.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for language editing and text polishing. We have not used generative AI tools for formulating research ideas, or writing code, and data generation and figure creation. Additionally, we used generative AI tools for improving the grammatical correctness and general readability of the manuscript. We have reviewed all AI-assisted work. Specifically, all AI-edited text was carefully reviewed by the authors line-by-line to ensure that the original scientific meaning was perfectly preserved, no inaccurate statements were introduced, and no technical claims were altered. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

See the ICLR 2027 AI Policy for Authors for more details. This statement should not be more than 1 page.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, volume 2024, pp. 21246–21263, 2024.

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

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *ACL*, pp. 9313–9332, 2024.

Yong Du, Yuchen Yan, Fei Tang, Zhengxi Lu, Chang Zong, Weiming Lu, Shengpei Jiang, and Yongliang Shen. Test-time reinforcement learning for gui grounding via region consistency. In *AAAI*, volume 40, pp. 30593–30601, 2026.

Aarash Feizi, Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Kaixin Li, Rabiul Awal, Xing Han Lù, Johan Obando-Ceron, Juan A Rodriguez, Nicolas Chapados, et al. Grounding computer use agents on human demonstrations. *arXiv preprint arXiv:2511.07332*, 2025.

Boyu Gou, 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. *arXiv preprint arXiv:2410.05243*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *ICLR*, volume 2024, pp. 32694–32717, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Woogyeol Jin, Taywon Min, Yongjin Yang, Swanand Ravindra Kadhe, Yi Zhou, Dennis Wei, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. *arXiv preprint arXiv:2603.07079*, 2026.

Jiaze Li, Hao Yin, Haoran Xu, Boshen Xu, Wenhui Tan, Zewen He, Jianzhong Ju, Zhenbo Luo, and Jian Luan. Video-opd: Efficient post-training of multimodal large language models for temporal video grounding via on-policy distillation. *arXiv preprint arXiv:2602.02994*, 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 *ACM MM*, pp. 8778–8786, 2025.

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

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

Ziwei Liu, Tao Feng, Borui Kang, Yanbing Yang, and Jun Luo. Zoom to essence: Trainless gui grounding by inferring upon interface elements. *arXiv preprint arXiv:2603.14448*, 2026.

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 *AAAI*, 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*, 2025a.

Tiange Luo, Lajanugen Logeswaran, Justin Johnson, and Honglak Lee. Visual test-time scaling for gui agent grounding. In *ICCV*, pp. 19989–19998, 2025b.

Mingyu Ouyang, Kevin Qinghong Lin, Mike Zheng Shou, and Hwee Tou Ng. Focusui: Efficient ui grounding via position-preserving visual token selection. *arXiv preprint arXiv:2601.03928*, 2026.

Leyi Pan, Shuchang Tao, Yunpeng Zhai, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, and Lijie Wen. Rlcsd: Reinforcement learning with contrastive on-policy self-distillation. *arXiv preprint arXiv:2606.11709*, 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.

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.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, et al. Gui-g^2: Gaussian reward modeling for gui grounding. In *AAAI*, volume 40, pp. 33214–33222, 2026.

Qwen Team. Qwen2.5-vl, January 2025. URL `https://qwenlm.github.io/blog/qwen2.5-vl/`.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, et al. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. *arXiv preprint arXiv:2507.19478*, 2025.

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. *arXiv preprint arXiv:2506.03143*, 2025.

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*, 2025.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026.

Qianhao Yuan, Jie Lou, Xing Yu, Hongyu Lin, Le Sun, Xianpei Han, and Yaojie Lu. Vision-opd: Learning to see fine details for multimodal llms via on-policy self-distillation. *arXiv preprint arXiv:2605.18740*, 2026.

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

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

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

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 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. *arXiv preprint arXiv:2505.15810*, 2025.

Zhou Ziheng, Jiaqi Li, Huacong Tang, Ying Nian Wu, and Demetri Terzopoulos. Less is more: Early stopping rollout for on-policy distillation. *arXiv preprint arXiv:2605.27028*, 2026.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *NeurIPS*, 38:131459–131483, 2026.

A APPENDIX

This Appendix provides the details of construction of the Reflector training dataset, additional qualitative and quantitative analysis of the Reflector, computational overhead analysis, scalability to larger base models, extension of our method to the unsupervised training setting, and comparisons with visual bounding box prompting.

A.1 CONSTRUCTION OF THE REFLECTOR TRAINING DATASET

To train the MLLM-based Reflector π_R to accurately evaluate test-time explorations and provide diagnostic reasoning, we construct a training dataset based on the *Functional Split* of GroundCUA (Feizi et al., 2025). **The constructed Reflector training dataset will be released.** It is designed to capture a diverse set of predictions while maintaining balanced labels, proceeding as follows:

Prediction Generation. For each instruction-image pair in the GroundCUA, we employ the base model to perform inference. To encourage a diverse range of predictions, we set the sampling temperature to 1.0 and sample 8 distinct coordinate predictions per instruction.

Label Assignment. Each generated coordinate is evaluated against the ground-truth bounding box. A discrete binary label $S^* \in \{0, 1\}$ is assigned to each prediction, where $S^* = 1$ indicates a successful grounding (the predicted point falls within the target bounding box) and $S^* = 0$ indicates a failure.

Sample Filtering. We group the 8 predictions by their instruction and apply a filtering strategy: we exclusively retain the groups that exhibit mixed results. Specifically, an instruction is kept only if its 8 rollouts contain at least one correct prediction $S^* = 1$ and at least one incorrect prediction $S^* = 0$.

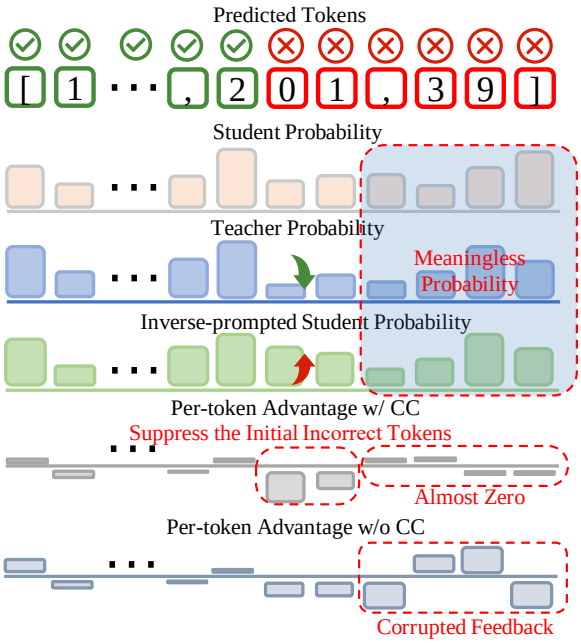


Figure 4: Illustration of Contrastive Calibration Method. During auto-regressive generation, an incorrect prediction, *e.g.,* the initial incorrect token '0', causes subsequent tokens to drift. Without CC, computing advantages on these drifted tokens results in corrupted feedback. By contrast, CC leverages an inverse-prompted student. This produces a negative penalty to suppress the initial incorrect tokens. As errors accumulate, the conditioned incorrect prefix forces both the teacher and the inverse-prompted student's distributions to align, decaying the advantage to near zero.

Table 3: Prompt ablation for the student and self-teacher.

Prompt setting	SSv2	SSP	MMG
Student: L	80.4	20.3	57.5
Teacher: $L(S)$	83.7	25.3	61.4
Teacher: $L(S, R)$	85.4	26.9	65.3

Table 4: Best-of-8 *v.s.* weight internalization. The supervision and computation columns specify the source of feedback and the number of rollouts for each input.

Setting	Feedback	Rollouts	Weight update	SSv2	SSP	MMG
Single rollout	None	1	No	80.4	20.3	57.5
Reflector best-of-8	S	8	No	87.9	29.1	64.4
Oracle best-of-8 (Upper Bound)	Ground-truth	8	No	91.2	36.6	73.9
R-OPSD (internalized)	S, R	1	Yes	88.8	30.5	66.6

Balanced Sampling. To ensure a balanced distribution of positive ("Yes") and negative ("No") evaluation results, we randomly select exactly one correct prediction $S^* = 1$ and one incorrect prediction $S^* = 0$ from the 8 generated rollouts for each retained instruction.

The final Reflector training dataset consists of approximately $10,160$ prediction-label pairs, yielding a 1:1 ratio of positive to negative samples.

Discussion on Cold-Start SFT. Additionally, we have explored a supervised fine-tuning cold-start strategy by synthesizing 30K high-quality reasoning trajectories using Qwen3.7-plus. However, results indicated that incorporating this cold-start phase yielded no noticeable performance gain compared to directly applying RL to the base model for the training of the Reflector.

A.2 Additional Diagnostic Experiments

To verify that the reflection-conditioned teacher provides information beyond the student's prior, we compare the grounding performance of the student and teachers under different prompts on the same inputs. The S-only condition isolates the contribution of the binary evaluation, while the (S, R) condition additionally provides the free-form reflection. As shown in Table 3, conditioning on S improves accuracy over the unconditioned student by $+3.3$, $+5.0$, and $+3.9$ on SSv2, SSP, and MMG, respectively. Adding the reflection R produces a further improvement, reaching $+5.0$, $+6.6$, and $+7.8$ over the student. This consistent gain, especially on MMG, indicates that the reflection supplies corrective grounding information.

We also compare parameter internalization with the annotation-free best-of-8 inference baseline, which reranks the existing eight candidates using the Reflector score. The oracle best-of-8 result is included as an upper-bound reference. The best-of-8 results in Table 4 show that Reflector-based reranking alone improves the single-rollout baseline by $+7.5$, $+8.8$, and $+6.9$ on SSv2, SSP, and MMG. R-OPSD obtains higher accuracy than reranking on all three datasets, while requiring only one rollout after internalization.

A.3 More Experimental Results

Reliability Evaluation of the MLLM-based Reflector. The efficacy of our Test-Time Self-Evolution framework relies heavily on the quality of the evaluation score S and diagnostic reasoning R provided by the Reflector π_{R}. We evaluate the Reflector on a held-out test set consisting of 1,000 randomly sampled instruction-prediction pairs from SSv2, SSP, and MMG. We maintain a balanced 1:1 ratio of successful and failed groundings. We assess its performance as a binary classifier $S \in \{0, 1\}$ using three metrics: Accuracy, Precision, and Recall. Here, the positive class represents successful groundings $S^* = 1$.

Table 5: Performance metrics of the MLLM-based Reflector.

Model	Accuracy	Precision	Recall
Qwen2.5-VL-3B (Zero-shot)	76.5	72.9	85.4
Trained Reflector π_R	**89.5**	**86.9**	**93.3**
Qwen3-VL-2B (Zero-shot)	80.5	78.9	86.0
Trained Reflector π_R	**91.7**	**88.8**	**95.7**

Table 6: Training time Analysis. The Relative Cost is normalized against the standard GRPO baseline.

Method	Time (m)	Relative Cost
GRPO + GUI-RCPO	449	$1.00\times$
GRPO + Binary	447	$1.00\times$
R-OPSD (w/o CC)	129	$\approx 0.29\times$
R-OPSD (w/ CC)	151	$\approx 0.34\times$
R-OPSD (w/ CC & QA)	778	$\approx 1.73\times$

As shown in Table 5, after training via GRPO with format and binary rewards, our Reflector achieves an evaluation Accuracy of 89.5% and 91.7%, demonstrating its capability in determining whether a predicted coordinate aligns with the user instruction.

Computational Overhead and Training Time Analysis. To provide a comprehensive understanding of the computational overhead, we conduct a training time analysis using the Qwen2.5-VL-3B model with 4 NVIDIA A100 40GB GPUs. As shown in Table 6, to compute relative advantages, GRPO samples a group of trajectories $K = 8$ for every single query, leading to the total training time of 449 minutes. R-OPSD operates on a different mechanism. By utilizing the Reflector's feedback as privileged information to construct a self-teacher, R-OPSD performs a token-level knowledge distillation. The model only needs to sample a single trajectory per query. Even with the additional forward passes required for the self-teacher and the inverse-prompted student in Contrastive Calibration (CC), the R-OPSD w/ CC only takes 151 minutes, reducing the training time by more than 34% compared to GRPO.

Our full framework (R-OPSD w/ CC & QA) achieves the optimal performance by integrating the token-level advantage. Re-introducing QA necessitates returning to the $K = 8$ group rollout setting, which increases the training time to 778 minutes ($1.73\times$). For resource-constrained deployment environments, R-OPSD w/ CC serves as a highly efficient learning method that operates at $1/3$ the cost of standard RL. The full framework leverages the combined strengths of both token-level and query-level advantages, offering a computational trade-off.

Scalability to Larger Base Models. In the main text, our experiments are conducted on the 3B and 2B parameter scales, *e.g.,* Qwen2.5-VL-3B and Qwen3-VL-2B. To further demonstrate the scalability of our Test-Time Self-Evolution framework, we extend our evaluation to a larger base model. The evaluation results are shown in Table 7. The Qwen2.5-VL-7B and Qwen3-VL-8B base models exhibit stronger initial grounding capabilities than 3B/2B versions. However, even with this baseline, our framework consistently yields performance improvements. For instance, adapting the 7B model on MMG using our framework boosts the accuracy from 68.2% to 79.2%. Our method still outperforms the TTRL method GUI-RCPO. These results further demonstrate the scalability of our framework.

Comparison with Visual Bounding Box Prompting. Many methods directly draw a bounding box or point on the UI screenshot for representing grounding coordinates (Zhang et al., 2026b; Luo et al., 2025b). To evaluate this design, we conduct an ablation study comparing textual coordinate representation against visual bounding box representation in two critical components of our framework: the Reflector's evaluation and the teacher model's privileged information prompt. For the Reflector + V and Ours + V, instead of feeding the textual coordinate token B in the prompt, we draw a red bounding box corresponding to the prediction on the input image I.

Table 7: Scalability evaluation using Qwen2.5-VL-7B and Qwen3-VL-8B base models. We report the target dataset accuracy when adapting on SSv2 and MMG, respectively.

Method	Adapt on SSv2			Adapt on MMG		
	SSv2	SSP	MMG	SSv2	SSP	MMG
Qwen2.5-VL-7B	86.8	19.9	68.2	86.8	19.9	68.2
+ GUI-RCPO	89.2	25.9	70.3	88.3	25.5	70.9
+ Ours	**92.7**	**31.4**	**77.9**	**92.5**	**32.3**	**79.2**
Qwen3-VL-8B	92.9	53.5	82.7	92.9	53.5	82.7
+ GUI-RCPO	93.2	53.8	83.3	93.5	54.6	83.4
+ Ours	**94.5**	**55.8**	**84.3**	**94.1**	**56.3**	**84.7**

Table 8: Performance comparison of the Reflector with visual bounding box prompt and textual coordinate prompt.

Model	Accuracy	Precision	Recall
Qwen2.5-VL-3B (Zero-shot)	76.5	72.9	85.4
Trained Reflector π_R	**91.7**	**88.8**	**95.7**
Qwen2.5-VL-3B + V (Zero-shot)	73.2	73.5	78.6
Trained Reflector + V π_R	**78.3**	**78.5**	**79.5**

Table 8 summarizes the evaluation results. Our trained Reflector with textual coordinates achieves a remarkable accuracy of 91.7%, outperforming the zero-shot base model by 15.2%. Conversely, employing visual prompts (+ V), the Trained Reflector + V only reaches 78.3% accuracy. The negative impact of visual prompting is amplified during the parameter internalization stage. As demonstrated in Table 9, leveraging drawn images (Ours + V) as the privileged visual context for the self-teacher in R-OPSD leads to negative transfer.

Extension beyond Test-Time Data. While our Test-Time Self-Evolving framework is designed to adapt models to unseen interfaces after deployment, its core internalization mechanism, R-OPSD, inherently serves as an unsupervised training algorithm. Instead of episodic exploration on test sets, we train the base model directly on the training set of GUI-R1 (Luo et al., 2025a), which contains about 3000 samples. To simulate the unsupervised training setting, we only leverage the image and instruction. The results are summarized in Table 10. It is shown that our method consistently outperforms the base model across all three datasets, demonstrating its effectiveness in unsupervised training setting.

A.4 Qualitative Analysis of the Reflector

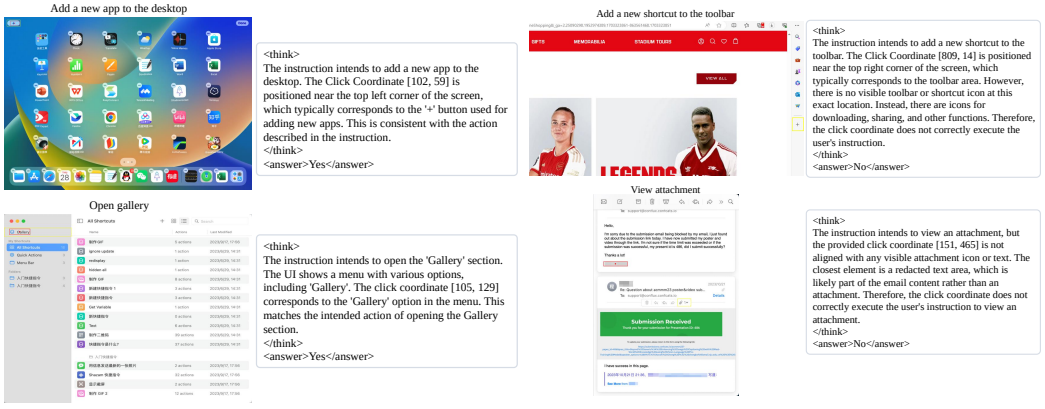
To provide an intuitive understanding of the Reflector's capabilities, we present qualitative visualizations of its evaluation process in Fig. 5. The visualization showcases the Reflector's multi-modal reasoning across two typical test-time exploration scenarios: a successful grounding ($S = 1$) and a failed grounding ($S = 0$). As illustrated in the successful case, the Reflector accurately parses the user's intent, spatially grounds the agent's predicted coordinate to the corresponding UI element, and logically confirms the match. More importantly, in the failed case, the Reflector does not merely

Table 9: Performance comparison of our method with visual bounding box prompt and textual coordinate prompt.

Method	Adapt on SSv2			Adapt on MMG		
	SSv2	SSP	MMG	SSv2	SSP	MMG
Qwen2.5-VL-3B	80.4	20.3	57.5	80.4	20.3	57.5
+ Ours	88.8	30.5	66.6	87.5	30.1	68.2
+ Ours + V	77.3	18.3	53.2	77.5	17.4	54.7

Table 10: Performance of our method on the unsupervised training setting.

Method	GUI-R1 Training Set		
	SSv2	SSP	MMG
Qwen2.5-VL-3B	80.4	20.3	57.5
+ Ours	**87.8**	**29.2**	**67.2**



(a) Successful Exploration ($S = 1$)

(b) Failed Exploration ($S = 0$)

Figure 5: Qualitative visualizations of the Reflector's evaluation and reasoning. The red bounding boxes denote the predicted coordinates. The yellow bounding boxes highlight the corresponding UI element.

output a binary rejection. Instead, it explicitly pinpoints the exact cause of the failure within its step-by-step reflection R.