Test-Time Reinforcement Learning for GUI Grounding via Region Consistency

Yong Du[1,2,*], **Yuchen Yan**[1,*], **Fei Tang**[1], **Zhengxi Lu**[1], **Chang Zong**[3], **Weiming Lu**[1,†], **Shengpei Jiang**[4], **Yongliang Shen**[1,†]

[1]Zhejiang University,
[2]Central South University,
[3]Zhejiang University of Science and Technology,
[4]SF Technology
duyong@csu.edu.cn, {yanyuchen, syl}@zju.edu.cn

Abstract

Graphical User Interface (GUI) grounding, the task of mapping natural language instructions to precise screen coordinates, is fundamental to autonomous GUI agents. While existing methods achieve strong performance through extensive supervised training or reinforcement learning with labeled rewards, they remain constrained by the cost and availability of pixel-level annotations. We observe that when models generate multiple predictions for the same GUI element, the spatial overlap patterns reveal implicit confidence signals that can guide more accurate localization. Leveraging this insight, we propose **GUI-RC (Region Consistency)**, a test-time scaling method that constructs spatial voting grids from multiple sampled predictions to identify consensus regions where models show highest agreement. Without any training, GUI-RC improves accuracy by 2-3% across various architectures on ScreenSpot benchmarks. We further introduce **GUI-RCPO (Region Consistency Policy Optimization)**, transforming these consistency patterns into rewards for test-time reinforcement learning. By computing how well each prediction aligns with the collective consensus, GUI-RCPO enables models to iteratively refine their outputs on unlabeled data during inference. Extensive experiments demonstrate the generality of our approach: using only 1,272 unlabeled data, GUI-RCPO achieves 3-6% accuracy improvements across various architectures on ScreenSpot benchmarks. Our approach reveals the untapped potential of test-time scaling and test-time reinforcement learning for GUI grounding, offering a promising path toward more data-efficient GUI agents.

Code — https://github.com/zju-real/gui-rcpo
Project — https://zju-real.github.io/gui-rcpo

Introduction

The rapid advancement of GUI (Graphical User Interface) agents is fundamentally transforming human-device interaction, enabling users to control complex interfaces in digital devices through natural language across diverse applications (Gou et al. 2024; Tang et al. 2025c; Wang et al. 2024). At the heart of these systems lies GUI grounding, the critical capability to accurately map natural language instructions to precise pixel coordinates on UI elements (Cheng et al. 2024; Tang et al. 2025b). This fundamental task determines whether an agent can successfully execute user commands, making it the cornerstone of reliable GUI automation.

Current approaches to GUI grounding have achieved impressive results through extensive train-time optimization. These methods fall into two main categories: supervised fine-tuning (SFT) with large-scale annotated datasets (Qin et al. 2025; Liu et al. 2025a; Wu et al. 2024b) and reinforcement learning with carefully designed reward functions (Lu et al. 2025a; Liu et al. 2025b; Luo et al. 2025a; Tang et al. 2025a). However, these approaches share two fundamental limitations. First, they rely exclusively on train-time scaling while leaving test-time computation underutilized, missing opportunities for performance gains through inference-time optimization. Second, they require extensive labeled data, where each sample demands precise pixel-level annotations, creating a significant bottleneck for scaling to new domains and applications (Chu et al. 2025; Wu et al. 2025b).

This raises a critical question: ***Can we leverage test-time computation to enhance GUI grounding performance without relying on any additional labeled data?*** Recent breakthroughs in large language models have demonstrated the remarkable potential of test-time scaling (Guan et al. 2025; Snell et al. 2024; Muennighoff et al. 2025). Self-consistency (Wang et al. 2023) aggregates multiple reasoning paths through majority voting, achieving substantial improvements in mathematical reasoning. Test-time reinforcement learning (TTRL) (Zuo et al. 2025) enables models to self-improve on unlabeled data by generating experiences and computing rewards during inference. These successes in language domains suggest potential for applying test-time scaling to vision-language tasks like GUI grounding.

However, adapting test-time scaling to GUI grounding presents unique challenges. Unlike text-based reasoning where outputs are discrete tokens, GUI grounding operates in a continuous, high-resolution coordinate space where minor pixel deviations can lead to incorrect element selection (Yang et al. 2025). The visual complexity of modern interfaces, with overlapping elements, dynamic layouts, and varying resolutions, introduces significant prediction uncertainty. The key insight is to transform this uncertainty from

[*]These authors contributed equally.
[†]Corresponding author.

a limitation into an opportunity: when models generate multiple predictions for the same element, the patterns of agreement and disagreement across predictions reveal valuable information about localization confidence of the model.

In this work, we present GUI-RC (GUI Region Consistency), a test-time scaling approach that aggregates spatial information across multiple model predictions to improve grounding accuracy. Our core observation is that when sampling multiple predictions from a model, certain screen regions consistently appear across different outputs, indicating higher confidence in those locations. By constructing a spatial voting mechanism that identifies consensus regions with maximum overlap, GUI-RC achieves significant performance improvements (e.g., +2.75% on OS-Atlas-Base-7B) without any additional training or labeled data.

Building upon this foundation, we introduce GUI-RCPO (GUI Region Consistency Policy Optimization), which enables test-time training through region consistency signals. Inspired by recent advances in TTRL (Zuo et al. 2025), GUI-RCPO computes rewards based on how well each prediction aligns with the consensus across multiple samples, then uses these self-generated rewards to update model parameters during inference. This label-free optimization further improves performance using only 1,272 unlabeled data (e.g., +5.5% on Qwen2.5-VL-3B-Instruct), demonstrating that models can effectively self-improve on unlabeled data.

Our main contributions are:

- We propose GUI-RC, a test-time scaling method for GUI grounding that leverages spatial voting across multiple predictions to improve localization accuracy without additional training or labeled data.
- We introduce GUI-RCPO, a test-time reinforcement learning method that uses region consistency as a self-supervised reward signal, enabling models to improve grounding capabilities through policy optimization on unlabeled GUI screenshots.
- We demonstrate consistent improvements across multiple benchmarks and model architectures. GUI-RC improves accuracy by 2-3%, while GUI-RCPO achieves further gains of 3-6% through label-free optimization.
- We reveal that further applying GUI-RC after GUI-RCPO yields additional performance gains, demonstrating that our methods support progressive, self-bootstrapping improvement without external supervision, and provide a complementary alternative to train-time optimization for GUI automation.

Related Works

GUI Grounding

GUI grounding refers to the task of locating target elements on a screen based on natural language instructions. Given a screenshot s and an instruction i, the model M is required to output the specific position of the target element. Current approaches primarily fall into two paradigms. The first formulates GUI grounding as point prediction, directly outputting the coordinate (x, y). These include supervised fine-tuning methods (Cheng et al. 2024; Lin et al. 2024; Wu et al. 2024a; Xu et al. 2025) and reinforcement learning methods (Shi et al. 2025a,b; Lu et al. 2025b; Liu et al. 2025d). The second paradigm predicts bounding boxes (x^-, y^-, x^+, y^+) representing the region that best matches the instruction. Representative works include OS-Atlas (Wu et al. 2024b) for cross-platform corpus development, and reinforcement learning approaches (Tang et al. 2025a; Zhou et al. 2025) for region-based grounding optimization. While all existing methods rely heavily on train-time optimization with labeled data, our work takes an orthogonal approach by leveraging test-time computation to improve grounding accuracy without additional training.

Test-Time Scaling

Test-time strategies for LLMs have shown that increasing inference computation can substantially enhance output accuracy without altering model weights (Wei et al. 2022; Madaan et al. 2023; Liu et al. 2025c). Representative strategies include self-consistency voting that aggregates multiple outputs to select the most confident (Wang et al. 2023), tree-based methods exploring diverse reasoning paths (Yao et al. 2023; Muennighoff et al. 2025), and test-time reinforcement learning (TTRL) enabling models to iteratively refine their outputs using self-generated rewards (Zuo et al. 2025). Extending these strategies to GUI grounding introduces challenges due to continuous nature of coordinate predictions. Existing test-time strategies for GUI grounding primarily rely on zoom-based refinement, such as methods proposed by Wu et al. (2025a) and Luo et al. (2025b). Our methods introduce a spatial voting mechanism that transforms the uncertainty in continuous predictions into reliable consensus regions, enabling effective test-time scaling that works across both point-based and region-based grounding paradigms, without requiring specialized preprocessing.

Methods

We first formalize the GUI grounding task and identify the key challenges in applying test-time scaling to this domain. We then present GUI-RC, our test-time scaling approach that leverages spatial consistency across multiple predictions to improve grounding accuracy. Finally, we introduce GUI-RCPO, which extends region consistency to reward signals for test-time reinforcement learning on unlabeled GUI data. Figure 1 provides an overview of both methods.

Problem Formulation

GUI grounding aims to map natural language instructions to precise locations on graphical interfaces. Formally, given a screenshot $s \in \mathbb{R}^{H \times W \times 3}$ and an instruction i, a model M outputs the spatial location of the UI element that best matches the instruction. As discussed in the previous section, this task has two primary formulations:

$$\text{Point-based: } M(s, i) \rightarrow (x, y) \tag{1}$$

$$\text{Region-based: } M(s, i) \rightarrow (x^-, y^-, x^+, y^+) \tag{2}$$

Both formulations face a fundamental challenge: the continuous nature of coordinate prediction introduces inherent uncertainty. This uncertainty is compounded by the complex-

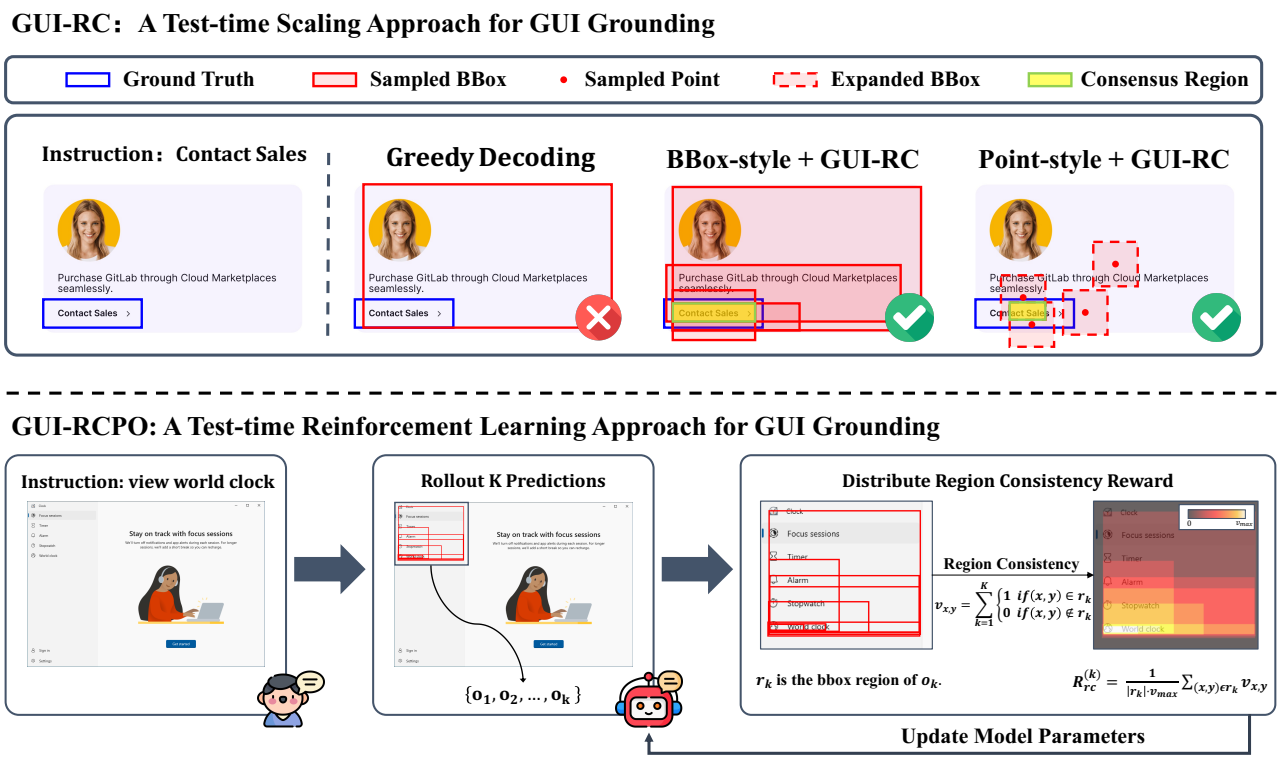

Figure 1: Overview of our test-time scaling methods for GUI grounding. **Upper**: GUI-RC aggregates K sampled predictions through spatial voting to extract a consensus region, achieving more accurate localization than greedy decoding. **Lower**: GUI-RCPO computes region consistency rewards based on the voting heatmap and uses these self-supervised signals to update model parameters, enabling label-free improvement through test-time reinforcement learning.

ity of modern interfaces with overlapping elements, varying resolutions, and ambiguous natural language descriptions.

Our key insight is that, rather than regarding this uncertainty as a limitation, it can be leveraged for improvement. When a model generates multiple predictions for the same input, their spatial distribution reveals implicit confidence patterns that can guide more accurate localization.

GUI-RC: Test-Time Scaling via Spatial Voting

GUI-RC transforms the uncertainty in individual predictions into robust consensus through spatial aggregation. As illustrated in the upper part of Figure 1, while greedy decoding fails to locate the correct element, GUI-RC successfully identifies the target by aggregating information across multiple samples. The method operates in three stages: multi-sample generation, spatial voting, and consensus extraction.

Multi-Sample Generation. Given an input pair (s, i), we sample K predictions from the model using temperature-based sampling:

$$\{M_k(s,i)\}_{k=1}^{K} \sim p_\theta(\cdot|s,i) \tag{3}$$

This sampling process naturally produces variation in predictions due to the continuous output space and model uncertainty. Crucially, we observe that despite this variation, certain screen regions appear consistently across samples, indicating higher model confidence.

Spatial Voting Mechanism. To quantify this consistency, we construct a spatial voting grid $v \in \mathbb{R}^{H \times W}$ matching the screenshot resolution. Each sampled prediction contributes votes to this grid:

$$v_{x,y} = \sum_{k=1}^{K} \mathbb{1}[(x,y) \in R_k] \tag{4}$$

where R_k represents the region from the k-th prediction. For region-based models, R_k is simply the predicted bounding box. For point-based models, we approximate the implicit attention region by expanding the predicted point (x_k, y_k) into a square region of size

$$R_k = [x_k - \alpha/2, x_k + \alpha/2] \times [y_k - \alpha/2, y_k + \alpha/2] \tag{5}$$

This expansion is necessary because point predictions, while precise, do not explicitly encode the spatial extent of the target element. As shown in Figure 1, the expanded regions for point-style models (dashed red boxes) enable meaningful spatial voting similar to bbox-style models.

Consensus Extraction. The voting grid v now encodes the collective attention of the model across samples. We extract the consensus region through a principled selection process that first identifies the maximum vote count across the entire grid as $v_{\max} = \max_{x,y} v_{x,y}$. This maximum value represents the highest level of agreement among predictions.

We then find all contiguous regions where every pixel has this maximum vote count, forming the set $\mathcal{R}_{v_{\max}} = \{r : \forall(x,y) \in r, v_{x,y} = v_{\max}\}$. Among these high-confidence regions, we select the one with the largest area as our final consensus region: $\hat{r}_{\text{cons}} = \arg\max_{r \in \mathcal{R}_{v_{\max}}} |r|$. The consensus region $\hat{r}_{\text{cons}}$ represents the area where the model shows highest and most consistent attention, providing a more reliable grounding prediction than any individual sample.

GUI-RCPO: Test-Time Reinforcement Learning via Region Consistency

While GUI-RC improves performance through inference-time aggregation, we further explore whether region consistency can guide model improvement through test-time training. As illustrated in the lower part of Figure 1, GUI-RCPO transforms region consistency into a self-supervised reward signal for policy optimization.

Region Consistency as Reward. The key insight is that predictions aligning with high-consistency regions should be reinforced, while outliers should be suppressed. For each sampled prediction r_k in the rollout, we compute its region consistency reward:

$$R_{rc}^{(k)} = \frac{1}{|r_k| \cdot v_{\max}} \sum_{(x,y) \in r_k} v_{x,y} \tag{6}$$

This reward measures the average vote density within the predicted region, normalized by the region size and maximum possible votes. As visualized in Figure 1, the heatmap representation shows how different regions receive varying levels of votes, with warmer colors indicating higher consistency. Predictions that overlap with these high-vote regions receive higher rewards, encouraging the model to converge toward consensus areas.

Policy Optimization. We formulate GUI grounding as a reinforcement learning problem where the VLM acts as policy π_θ. Using Group Relative Policy Optimization (GRPO) (Shao et al. 2024), we optimize the expected region consistency reward:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(s,i)\sim D}\mathbb{E}_{r\sim\pi_\theta(\cdot|s,i)}[A(r)\log\pi_\theta(r|s,i)] \tag{7}$$

where $A(r)$ is the advantage computed from relative rewards within each group of samples. GRPO's group-relative formulation is particularly suitable for our setting as it normalizes rewards across different inputs, preventing optimization bias toward easier examples.

A unique property of GUI-RCPO is its ability to progressively improve without external supervision. As the model updates its parameters based on region consistency rewards, its predictions become more concentrated around high-confidence regions, which in turn provides stronger and more reliable reward signals for further optimization. This self-bootstrapping process continues until the model converges to a stable distribution centered on consensus regions.

Experiments

Experiment Setup

Models. We evaluate our methods on a diverse VLMs to demonstrate their generality across different architectures and training paradigms. For general models, we use Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct (Bai et al. 2025), as well as InternVL3-2B-Instruct and InternVL3-8B-Instruct (Zhu et al. 2025), which represent state-of-the-art vision-language models at different scales. For GUI-specific models that have been explicitly trained on GUI tasks, we evaluate UGround-V1-7B (Gou et al. 2024), OS-Atlas-Base-7B (Wu et al. 2024b), UI-TARS-1.5-7B (Qin et al. 2025), and GUI-G2-7B (Tang et al. 2025a). These models span both point-based and region-based prediction paradigms, allowing us to assess the effectiveness of our methods across different output formats.

Evaluation Benchmarks and Metrics. We evaluate our methods on three GUI grounding benchmarks: ScreenSpot (Cheng et al. 2024), ScreenSpot-v2 (Wu et al. 2024b), and ScreenSpot-Pro (Li et al. 2025). ScreenSpot and ScreenSpot-v2 assess model's grounding performance in general GUI environments spanning Mobile, Web, and Desktop platforms. ScreenSpot-Pro specifically focuses on high-resolution and professional interfaces. Following standard evaluation protocols, we adopt grounding accuracy as our primary metric: a prediction is considered correct if the predicted point or the center of the predicted bounding box falls in the ground-truth bounding box (Cheng et al. 2024).

Implementation Details. For GUI-RC, we sample 64 outputs using a temperature of 0.5 and a top_p of 0.95 for voting, the hyperparameter α is set to 50. For the baselines, we employ greedy decoding with temperature 0. For GUI-RCPO, we adopt the VLM-R1 (Shen et al. 2025) framework and conduct TTRL training on the Screenspot-v2 benchmark without using the ground-truth data. For each input, 16 samples are generated with a temperature of 0.7 and top_p of 0.95. We train the models for 2 epochs (approx. 40 steps) with a global batch size of 64, learning rate of 1e-6, and KL penalty $\beta = 0.04$. All training and evaluation are conducted on 8 NVIDIA A100-80GB GPUs.

Main Results

Evaluation Results of GUI-RC Experiments

GUI-RC consistently improves the end-to-end grounding performance. We compared the performance of the base models on three benchmarks before and after applying GUI-RC. Table 1 presents the evaluation results. It can be observed that GUI-RC consistently improves the overall grounding capability across different models, regardless of its output style and whether the model is specifically trained for GUI tasks. For instance, OS-Atlas-Base-7B achieves an overall improvement of 2.75%, with a notable 6.28% increase in icon localization in mobile scenarios. Moreover, for general models like Qwen2.5-VL-3B/7B-Instruct that output in bbox-style, GUI-RC brings even greater improvements on the ScreenSpot-Pro compared to ScreenSpot and ScreenSpot-v2. This suggests that GUI-RC is particularly effective in helping models tackle more challenging grounding tasks involving high-resolution and professional GUIs.

GUI-RC achieves greater improvements when applied to bbox-style prediction models. Another observation is that GUI-RC provides greater improvements for models that

	SSv2.Mobile		SSv2.Desktop		SSv2.Web		SSv2.avg	SSv1.avg	SSPro.avg
	Text	Icon	Text	Icon	Text	Icon			
General Models									
InternVL3-2B-Instruct	89.92	76.44	38.89	26.19	46.43	25.32	52.75	51.02	1.03
w/ **GUI-RC**	89.92	77.49↑	38.33	24.60	46.07	27.00↑	52.91+0.16	52.20+1.18	1.33+0.30
InternVL3-8B-Instruct	94.19	79.58	79.44	53.17	91.07	71.73	80.97	79.72	13.28
w/ **GUI-RC**	94.19	81.15↑	80.56↑	56.35↑	91.07	71.73	81.68+0.71	80.03+0.31	12.46-0.82
Qwen2.5-VL-3B-Instruct	97.67	75.92	85.56	59.52	84.64	65.82	80.11	76.97	20.18
w/ **GUI-RC**	98.84↑	77.49↑	90.00↑	64.29↑	87.14↑	67.93↑	82.63+2.52	78.46+1.49	23.59+3.41
Qwen2.5-VL-7B-Instruct	98.84	84.29	86.67	73.81	88.57	78.90	86.48	84.20	19.80
w/ **GUI-RC**	99.92↑	85.86↑	91.11↑	73.02	91.79↑	81.43↑	88.52+2.04	85.53+1.33	23.97+4.17
GUI-specific Models									
UGround-V1-7B	96.51	82.72	96.11	82.54	92.50	83.12	89.62	87.11	31.50
w/ **GUI-RC**	96.51	83.77↑	95.56	84.13↑	92.86↑	81.43	89.62+0.00	87.34+0.23	31.63+0.13
UI-TARS-1.5-7B	96.51	86.39	95.00	87.30	88.21	86.50	90.17	87.74	40.92
w/ **GUI-RC**	96.12	86.91↑	96.11↑	90.48↑	90.36↑	86.50	91.12+0.95	88.52+0.78	41.18+0.26
OS-Atlas-Base-7B	91.47	72.25	88.33	64.29	86.43	72.57	80.82	79.80	18.41
w/ **GUI-RC**	91.47	78.53↑	88.89↑	68.25↑	89.29↑	76.37↑	83.57+2.75	81.45+1.65	19.67+0.16
GUI-G2-7B	99.61	92.15	95.56	88.89	95.00	88.61	93.79	91.51	46.43
w/ **GUI-RC**	99.61	93.19↑	95.56	88.1	95.00	88.61	93.87+0.08	91.90+0.39	47.88+1.45

Table 1: Performance (%) of the proposed test-time scaling method **GUI-RC** across GUI-Grounding benchmarks. Icons refer to output styles: Point ( ), Bounding-box ( ).

	SSv2.Mobile		SSv2.Desktop		SSv2.Web		SSv2.avg	SSv1.avg	SSPro.avg
	Text	Icon	Text	Icon	Text	Icon			
General Models									
Qwen2.5-VL-3B-Instruct	97.67	75.92	85.56	59.52	84.64	65.82	80.11	76.97	20.18
w/ **GUI-RCPO**	98.06↑	81.68↑	91.11↑	65.08↑	90.71↑	73.42↑	85.14+5.03	82.47+5.50	24.67+4.49
Qwen2.5-VL-7B-Instruct	98.84	84.29	86.67	73.81	88.57	78.90	86.48	84.20	19.80
w/ **GUI-RCPO**	98.84	87.43↑	91.11↑	76.19↑	92.5↑	80.17↑	88.92+2.48	86.64+2.44	25.93+6.13
GUI-specific Models									
UI-TARS-1.5-7B	96.51	86.39	95.00	87.30	88.21	86.50	90.17	87.74	40.92
w/ **GUI-RCPO**	97.29↑	86.39	97.22↑	82.54	91.07↑	87.34↑	90.96+0.79	88.60+0.86	41.43+0.51

Table 2: Performance (%) of the proposed test-time reinforcement learning method **GUI-RCPO** across GUI-Grounding benchmarks. Icons refer to output styles: Point ( ), Bounding-box ( ).

output bounding boxes compared to those output points. This is because when models predict bounding boxes, the bounding boxes inherently reflect the regions that the models are attending to. In contrast, for models predict points, when we manually expand a point into a bounding box, we are simulating the model's attention region. This may fail to accurately represent the actual region that the model focuses on, which further introduces biases in identifying the consensus regions, thus limiting the performance of GUI-RC. Nevertheless, GUI-RC still brings improvements to most point-style prediction models, indicating its robustness.

Evaluation Results of GUI-RCPO Experiments

GUI-RCPO is supervised by GUI-RC yet outperforms it. We compare the performance of base models with and without further TTRL training via GUI-RCPO on the three

benchmarks. As Table 2 shows, GUI-RCPO also brings consistent improvements and even outperforms GUI-RC. For instance, GUI-RC brings an improvement of 1.49% for Qwen2.5-VL-3B-Instruct on ScreenSpot, while after GUI-RCPO training, it achieves an impressive gain of 5.5%. Intuitively, the performance upper bound of GUI-RCPO should be that of GUI-RC, as it utilizes the region consistency as a reward signal for RL. However, in practice, GUI-RCPO not only matches but exceeds GUI-RC, which aligns with prior findings in TTRL (Zuo et al. 2025). This indicates that the model practically learns a more effective GUI grounding strategy through GUI-RCPO, rather than merely fitting to the consensus region. Notably, GUI-RCPO further brings performance gains for models that have already been specifically trained on GUI tasks, indicating the effectiveness of introducing the region consistency reward.

GUI-RCPO generalizes well in out-of-distribution scenarios. Although models are trained on Screenspot-v2, GUI-RCPO also shows significant improvement on ScreenSpot-Pro, which is an out-of-distribution benchmark featuring high-resolution and domain-specific GUIs. This further proves that GUI-RCPO does not rely on overfitting but genuinely enhances the model's general GUI grounding capability. Unlike direct fine-tuning on labeled data, which risks overfitting to the resolution and layout of the training set, GUI-RCPO enables robust generalization across different screen resolutions and interface layouts.

Analysis

Ablation Studies on Decoding Strategy of GUI-RC

We conduct ablation studies on the decoding strategy of GUI-RC to analyze how different parameters affect its GUI grounding performance. Specifically, we first fix the temperature, sampling number, and expand size hyperparameter α to 0.5, 64, and 50 respectively, then vary each parameter individually to observe its impact on GUI-RC. We employ Qwen2.5-VL-3B-Instruct (representing bbox-style prediction models), UI-TARS-1.5-7B and InternVL3-2B-Instruct (representing point-style prediction models) for ablation studies on the ScreenSpot-v2 benchmark. The results are shown in Figure 2.

Temperature. The performance of GUI-RC generally exhibits an increasing-then-decreasing trend as the temperature rises. As temperature controls the diversity of sampled outputs during decoding, a high temperature encourages broad exploration but also increases instability in model generation. Therefore, a moderate increase in temperature helps the model explore broader regions while generating relatively concentrated outputs. However, further increasing the temperature would cause the predicted regions to become overly dispersed, making it difficult to obtain a focused and reliable consensus region.

Sampling Number. As the number of sampled predictions increases, the performance of GUI-RC initially improves and then gradually plateaus. This is because with more samples, the distribution of predicted regions becomes more stable, leading the consensus region to converge toward a fixed area. Once the predictions reach sufficient diversity and coverage, additional samples contribute diminishing improvements, leading to performance saturation.

Hyperparameter α. The performance of GUI-RC follows an increasing-then-decreasing trend as α grows. As the hyperparameter α primarily affects the estimation of the attention area for point-style prediction models, both overly small and large expansion sizes introduce deviations in estimating the area models actually attend to. This results in larger deviations in the computed consensus region, thereby impairing the grounding accuracy.

GUI-RCPO Enables Consistent Improvements during Test-time Training

We observe the performance trajectories of Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct during GUI-RCPO training on three benchmarks, as shown in Figure 3. As training steps increase, the models' accuracy stably improves across all three GUI-Grounding benchmarks and converges around 80 steps. In particular, although the models are trained solely on ScreenSpot-v2, they do not exhibit overfitting to the training data or degradation on other benchmarks like ScreenSpot-Pro, demonstrating the robust generalization of GUI-RCPO.

Applying GUI-RC after GUI-RCPO Leads to Further Improvements

Model	SSv2.avg	SSPro.avg
General Models		
Qwen2.5-VL-3B-Instruct+GUI-RCPO	85.14	24.67
w/ **GUI-RC**	86.32+1.18	26.19+1.52
Qwen2.5-VL-7B-Instruct+GUI-RCPO	88.92	25.93
w/ **GUI-RC**	89.78+0.86	26.69+0.76

Table 3: Performance of applying GUI-RC to bbox-style prediction models after GUI-RCPO on ScreenSpot-v2 and ScreenSpot-Pro benchmarks. In this table, GUI-RC is performed with temperature = 1.0.

We further apply GUI-RC to the bbox-style prediction models after being trained with GUI-RCPO, and evaluate their performance on ScreenSpot-v2 and ScreenSpot-Pro. It is important to note that for GUI-RC in this experiment, we keep all other parameters consistent with previous settings, except increasing the sampling temperature to 1.0. This is because the reward signals in GUI-RCPO training encourage the model to predict more concentrated regions. Therefore, during GUI-RC, a higher decoding temperature is needed to encourage the model to explore broader regions.

The evaluation results are shown in Table 3. It can be observed that even after GUI-RCPO training, applying GUI-RC voting mechanism still leads to additional performance gains. For instance, Qwen2.5-VL-3B-Instruct can gain an additional 1.52% performance on ScreenSpot-Pro through GUI-RC even after GUI-RCPO. This indicates that our

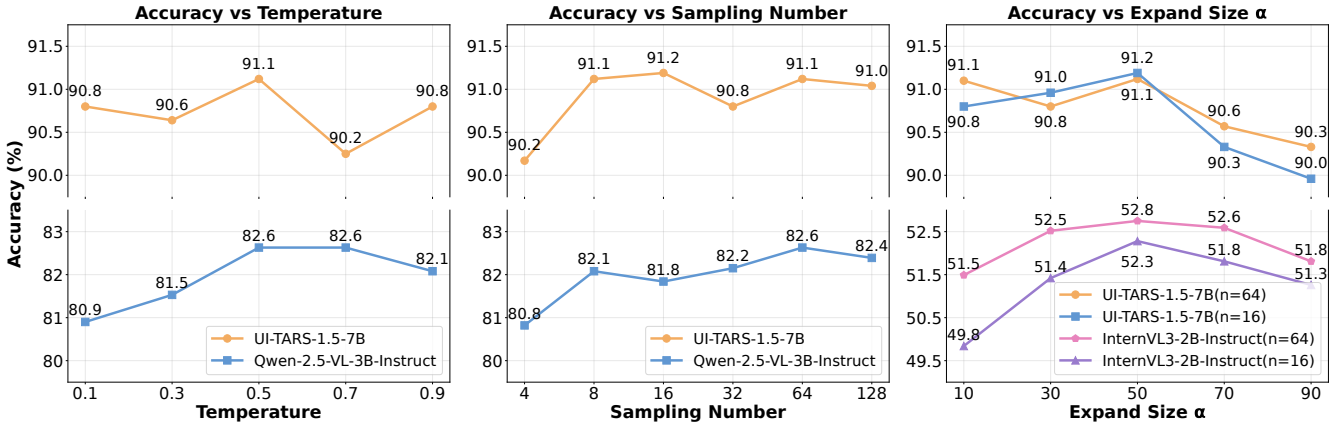

Figure 2: Ablation study results on ScreenSpot-v2 with varying temperature, sampling number, and hyperparameter α.

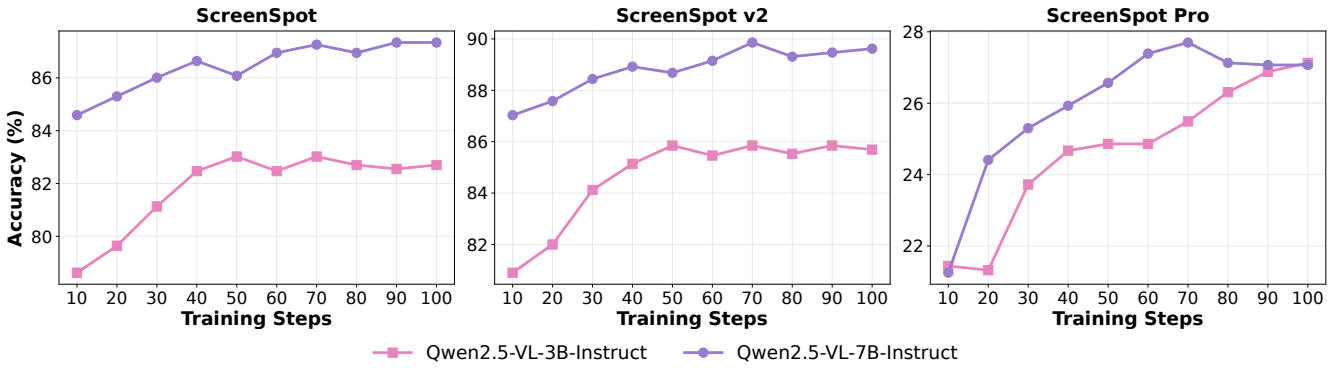

Figure 3: Accuracy (%) across training steps of Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct throughout GUI-RCPO.

methods enable the models to recursively improve themselves without relying on any external supervision.

Limitations

Performance gain limited in point-style grounding. Despite its effectiveness, GUI-RC has several limitations. As analyzed in the experiment section, GUI-RC brings relatively limited improvements for models with point-style outputs. Although most existing GUI Agents adopt point-style grounding in order to seamlessly integrate with subsequent action execution, an increasing number of recent studies (Wu et al. 2025b; Zhou et al. 2025) have pointed out the limitations of point-based prediction and the advantages of region-level supervision. Therefore, we expect that the strengths of GUI-RC will be amplified in future research.

Rely on the model's inherent capabilities. Moreover, GUI-RC primarily addresses misleading and biased hallucinations in grounding, but it is hard to resolve confusion hallucinations (i.e., the predicted region fails to match any valid UI element). In other words, GUI-RC assumes that the model has a certain ability to recognize the target element. It can tolerate the model's predictions to be imprecise or biased, but not completely random or unrelated. Therefore, GUI-RC requires the model to be familiar with the GUI environment, but it does not require the model to be specifically trained on GUI tasks.

Conclusion

We introduce GUI-RC, a test-time scaling approach for GUI grounding that leverages region consistency across multiple predictions to enhance model performance without requiring additional training. Building on this idea, we further proposed GUI-RCPO, a test-time reinforcement learning method that transforms region consistency into a self-supervised reward signal, enabling models to self-improve during inference without the need for labeled data. Extensive experiments across a wide range of general and GUI-specific models demonstrate that our methods consistently improve GUI grounding performance and generalize well to out-of-distribution scenarios. Our findings reveal the untapped potential of test-time training for GUI agents and suggest a promising direction toward more robust and data-efficient GUI automation systems.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62506332), National Key Research and Development Project (No. 2024YFB3312900), the Key Research and Development Program of Zhejiang Province, China (No. 2024C01034), the Fundamental Research Funds for the Central Universities (226-2024-00170), MOE Engineering Research Center of Digital Library, CIPS-LMG Huawei Innovation Fund and ZJU Kunpeng&Ascend Center of Excellence.

References

Bai, S.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; Song, S.; Dang, K.; Wang, P.; Wang, S.; Tang, J.; Zhong, H.; Zhu, Y.; Yang, M.; Li, Z.; Wan, J.; Wang, P.; Ding, W.; Fu, Z.; Xu, Y.; Ye, J.; Zhang, X.; Xie, T.; Cheng, Z.; Zhang, H.; Yang, Z.; Xu, H.; and Lin, J. 2025. Qwen2.5-VL Technical Report. arXiv:2502.13923.

Cheng, K.; Sun, Q.; Chu, Y.; Xu, F.; Li, Y.; Zhang, J.; and Wu, Z. 2024. SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents. arXiv:2401.10935.

Chu, T.; Zhai, Y.; Yang, J.; Tong, S.; Xie, S.; Schuurmans, D.; Le, Q. V.; Levine, S.; and Ma, Y. 2025. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. arXiv:2501.17161.

Gou, B.; Wang, R.; Zheng, B.; Xie, Y.; Chang, C.; Shu, Y.; Sun, H.; and Su, Y. 2024. Navigating the Digital World as Humans Do: Universal Visual Grounding for GUI Agents. arXiv:2410.05243.

Guan, X.; Zhang, L. L.; Liu, Y.; Shang, N.; Sun, Y.; Zhu, Y.; Yang, F.; and Yang, M. 2025. rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking. arXiv:2501.04519.

Li, K.; Meng, Z.; Lin, H.; Luo, Z.; Tian, Y.; Ma, J.; Huang, Z.; and Chua, T.-S. 2025. ScreenSpot-Pro: GUI Grounding for Professional High-Resolution Computer Use.

Lin, K. Q.; Li, L.; Gao, D.; Yang, Z.; Wu, S.; Bai, Z.; Lei, W.; Wang, L.; and Shou, M. Z. 2024. ShowUI: One Vision-Language-Action Model for GUI Visual Agent. arXiv:2411.17465.

Liu, Y.; Li, P.; Wei, Z.; Xie, C.; Hu, X.; Xu, X.; Zhang, S.; Han, X.; Yang, H.; and Wu, F. 2025a. InfiGUIAgent: A Multimodal Generalist GUI Agent with Native Reasoning and Reflection. arXiv:2501.04575.

Liu, Y.; Li, P.; Xie, C.; Hu, X.; Han, X.; Zhang, S.; Yang, H.; and Wu, F. 2025b. InfiGUI-R1: Advancing Multimodal GUI Agents from Reactive Actors to Deliberative Reasoners.

Liu, Y.; Li, Z.; Fang, Z.; Xu, N.; He, R.; and Tan, T. 2025c. Rethinking the Role of Prompting Strategies in LLM Test-Time Scaling: A Perspective of Probability Theory. *arXiv preprint arXiv:2505.10981*.

Liu, Y.; Liu, Z.; Zhu, S.; Li, P.; Xie, C.; Wang, J.; Hu, X.; Han, X.; Yuan, J.; Wang, X.; Zhang, S.; Yang, H.; and Wu, F. 2025d. InfiGUI-G1: Advancing GUI Grounding with Adaptive Exploration Policy Optimization. arXiv:2508.05731.

Lu, Z.; Chai, Y.; Guo, Y.; Yin, X.; Liu, L.; Wang, H.; Xiao, H.; Ren, S.; Xiong, G.; and Li, H. 2025a. UI-R1: Enhancing Efficient Action Prediction of GUI Agents by Reinforcement Learning.

Lu, Z.; Ye, J.; Tang, F.; Shen, Y.; Xu, H.; Zheng, Z.; Lu, W.; Yan, M.; Huang, F.; Xiao, J.; et al. 2025b. Ui-s1: Advancing gui automation via semi-online reinforcement learning. *arXiv preprint arXiv:2509.11543*.

Luo, R.; Wang, L.; He, W.; and Xia, X. 2025a. GUI-R1 : A Generalist R1-Style Vision-Language Action Model For GUI Agents.

Luo, T.; Logeswaran, L.; Johnson, J.; and Lee, H. 2025b. Visual Test-time Scaling for GUI Agent Grounding. arXiv:2505.00684.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36: 46534–46594.

Muennighoff, N.; Yang, Z.; Shi, W.; Li, X. L.; Fei-Fei, L.; Hajishirzi, H.; Zettlemoyer, L.; Liang, P.; Candès, E.; and Hashimoto, T. 2025. s1: Simple test-time scaling. arXiv:2501.19393.

Qin, Y.; Ye, Y.; Fang, J.; Wang, H.; Liang, S.; Tian, S.; Zhang, J.; Li, J.; Li, Y.; Huang, S.; Zhong, W.; Li, K.; Yang, J.; Miao, Y.; Lin, W.; Liu, L.; Jiang, X.; Ma, Q.; Li, J.; Xiao, X.; Cai, K.; Li, C.; Zheng, Y.; Jin, C.; Li, C.; Zhou, X.; Wang, M.; Chen, H.; Li, Z.; Yang, H.; Liu, H.; Lin, F.; Peng, T.; Liu, X.; and Shi, G. 2025. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. arXiv:2501.12326.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Shen, H.; Liu, P.; Li, J.; Fang, C.; Ma, Y.; Liao, J.; Shen, Q.; Zhang, Z.; Zhao, K.; Zhang, Q.; Xu, R.; and Zhao, T. 2025. VLM-R1: A Stable and Generalizable R1-style Large Vision-Language Model. arXiv:2504.07615.

Shi, Y.; Yu, W.; Li, Z.; Wang, Y.; Zhang, H.; Liu, N.; Mi, H.; and Yu, D. 2025a. MobileGUI-RL: Advancing Mobile GUI Agent through Reinforcement Learning in Online Environment. arXiv:2507.05720.

Shi, Y.; Yu, W.; Li, Z.; Wang, Y.; Zhang, H.; Liu, N.; Mi, H.; and Yu, D. 2025b. MobileGUI-RL: Advancing Mobile GUI Agent through Reinforcement Learning in Online Environment. *arXiv preprint arXiv:2507.05720*.

Snell, C.; Lee, J.; Xu, K.; and Kumar, A. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. arXiv:2408.03314.

Tang, F.; Gu, Z.; Lu, Z.; Liu, X.; Shen, S.; Meng, C.; Wang, W.; Zhang, W.; Shen, Y.; Lu, W.; Xiao, J.; and Zhuang, Y. 2025a. GUI-G^2: Gaussian Reward Modeling for GUI Grounding. arXiv:2507.15846.

Tang, F.; Shen, Y.; Zhang, H.; Chen, S.; Hou, G.; Zhang, W.; Zhang, W.; Song, K.; Lu, W.; and Zhuang, Y. 2025b. Think

Twice, Click Once: Enhancing GUI Grounding via Fast and Slow Systems. arXiv:2503.06470.

Tang, F.; Xu, H.; Zhang, H.; Chen, S.; Wu, X.; Shen, Y.; Zhang, W.; Hou, G.; Tan, Z.; Yan, Y.; Song, K.; Shao, J.; Lu, W.; Xiao, J.; and Zhuang, Y. 2025c. A Survey on (M)LLM-Based GUI Agents. arXiv:2504.13865.

Wang, J.; Xu, H.; Ye, J.; Yan, M.; Shen, W.; Zhang, J.; Huang, F.; and Sang, J. 2024. Mobile-Agent: Autonomous Multi-Modal Mobile Device Agent with Visual Perception. arXiv:2401.16158.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. arXiv:2203.11171.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Wu, H.; Chen, H.; Cai, Y.; Liu, C.; Ye, Q.; Yang, M.-H.; and Wang, Y. 2025a. DiMo-GUI: Advancing Test-time Scaling in GUI Grounding via Modality-Aware Visual Reasoning. arXiv:2507.00008.

Wu, Q.; Cheng, K.; Yang, R.; Zhang, C.; Yang, J.; Jiang, H.; Mu, J.; Peng, B.; Qiao, B.; Tan, R.; et al. 2025b. GUI-Actor: Coordinate-Free Visual Grounding for GUI Agents. *arXiv preprint arXiv:2506.03143*.

Wu, Z.; Han, C.; Ding, Z.; Weng, Z.; Liu, Z.; Yao, S.; Yu, T.; and Kong, L. 2024a. OS-Copilot: Towards Generalist Computer Agents with Self-Improvement. arXiv:2402.07456.

Wu, Z.; Wu, Z.; Xu, F.; Wang, Y.; Sun, Q.; Jia, C.; Cheng, K.; Ding, Z.; Chen, L.; Liang, P. P.; and Qiao, Y. 2024b. OS-ATLAS: A Foundation Action Model for Generalist GUI Agents. arXiv:2410.23218.

Xu, Y.; Wang, Z.; Wang, J.; Lu, D.; Xie, T.; Saha, A.; Sahoo, D.; Yu, T.; and Xiong, C. 2025. Aguvis: Unified Pure Vision Agents for Autonomous GUI Interaction. arXiv:2412.04454.

Yang, Y.; Li, D.; Dai, Y.; Yang, Y.; Luo, Z.; Zhao, Z.; Hu, Z.; Huang, J.; Saha, A.; Chen, Z.; Xu, R.; Pan, L.; Savarese, S.; Xiong, C.; and Li, J. 2025. GTA1: GUI Test-time Scaling Agent. arXiv:2507.05791.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.

Zhou, Y.; Dai, S.; Wang, S.; Zhou, K.; Jia, Q.; and Xu, J. 2025. GUI-G1: Understanding R1-Zero-Like Training for Visual Grounding in GUI Agents.

Zhu, J.; Wang, W.; Chen, Z.; Liu, Z.; Ye, S.; Gu, L.; Tian, H.; Duan, Y.; Su, W.; Shao, J.; Gao, Z.; Cui, E.; Wang, X.; Cao, Y.; Liu, Y.; Wei, X.; Zhang, H.; Wang, H.; Xu, W.; Li, H.; Wang, J.; Deng, N.; Li, S.; He, Y.; Jiang, T.; Luo, J.; Wang, Y.; He, C.; Shi, B.; Zhang, X.; Shao, W.; He, J.; Xiong, Y.; Qu, W.; Sun, P.; Jiao, P.; Lv, H.; Wu, L.; Zhang, K.; Deng, H.; Ge, J.; Chen, K.; Wang, L.; Dou, M.; Lu, L.; Zhu, X.; Lu, T.; Lin, D.; Qiao, Y.; Dai, J.; and Wang, W. 2025. InternVL3: Exploring Advanced Training and Test-Time Recipes for Open-Source Multimodal Models. arXiv:2504.10479.

Zuo, Y.; Zhang, K.; Sheng, L.; Qu, S.; Cui, G.; Zhu, X.; Li, H.; Zhang, Y.; Long, X.; Hua, E.; Qi, B.; Sun, Y.; Ma, Z.; Yuan, L.; Ding, N.; and Zhou, B. 2025. TTRL: Test-Time Reinforcement Learning. arXiv:2504.16084.