You Only Show Once: Robust GUI Automation via One-Shot Human Demonstration

Anonymous ACL submission

Abstract

Autonomous GUI Agents have emerged as a promising paradigm for automating complex digital workflows, yet existing approaches remain heavily dependent on training with massive data to achieve generalization a requirement that fundamentally limits their adaptability in novel or unseen environments. In this paper, we present YOSO-Agent (You Only Show Once), an end-to-end multi-agent framework that enables robust GUI automation through a single human demonstration. To be specific, YOSO-Agent constructs structured reference data by segmenting demonstration trajectories into atomic actions and annotated key frames, and leverages this information to guide execution at inference time. The framework comprises three collaborative agents: a Task Manager Agent that aligns the current execution state with visual references for precise step tracking; an Action Agent that reproduces demonstration actions with accurate visual grounding; and a Reflection Agent that detects execution deviations and triggers adaptive recovery strategies. All agents are built upon fine-tuned vision-language models to handle cross-device visual variations. Comprehensive experiments demonstrate that YOSO-Agent achieves superior grounding accuracy and reliable end-to-end task execution over prior methods. Our results establish demonstration-guided learning as a practical and scalable pathway for deploying robust GUI Agents in real-world settings. Codes and datasets will be publicly available.

1 Introduction

The recent emergence of AI computer-use frameworks marks a significant milestone in autonomous GUI Agents. This capability promises to revolutionize user productivity by delegating complex, multi-step operations to intelligent assistants. These methods have the potential to transform how users engage with diverse applications and digital workflows.(Li et al., 2024; Tang et al., 2025)

However, despite growing adoption and theoretical promise, current approaches struggle to maintain reliability across diverse scenarios. As digital environments grow increasingly complex and dynamic, GUI Agents frequently misinterpret interface elements and exhibit poor resilience when confronted with intricate user commands. Representative works such as AppAgent (Zhang et al., 2025), SeeClick (Cheng et al., 2024), MobileAgent (Wang et al., 2024a,b, 2025; Ye et al., 2025), and Uground (Gou et al., 2024) utilize large-scale corpora to equip agents with zero-shot capabilities, assuming that training data sufficiently covers the distribution of target environments (Nguyen et al., 2025). They aim to enhance agent capabilities through general knowledge rather than specific guidance, which limits their adaptability when encountering scenarios that lie outside training distribution.

To address these limitations, recent efforts have shifted toward learning from task-specific demonstrations rather than pre-training, as explored in LearnAct (Liu et al., 2025), Instruction Agent (Li et al., 2025c), and ShowUI-Aloha (Zhang et al., 2026b). While this paradigm reduces reliance on large-scale training data, it introduces critical vulnerabilities: the transformation of visual demonstrations into simplified textual abstractions inevitably loses the detailed visual grounding in the original demonstration, and such textual references are prone to semantic drift when observations deviate from the demonstrated context. Moreover, existing methods generally lack strict mechanisms to maintain robust recovery protocols when execution deviates from the demonstrated trajectory.

To address these limitations, we propose YOSO-Agent (You Only Show Once), an end-to-end multi-agent framework that enhances GUI Agents through demonstrations. Unlike prior methods that rely on distilling demonstrations into intermediate textual representations for execution, YOSO-Agent maintains visual grounding throughout execution

by continuously referencing the demonstration's visual context, rather than depending solely on parsed instructions or workflows. It comprises three components: a Task Manager Agent that validates step correspondence by comparing screenshots against reference images and descriptions; an Action Agent that executes operations by reusing demonstration actions with visual grounding; and a Reflection Agent that triggers backtracking or retry upon detecting mismatches between the current and reference states. All agents are powered by finetuned vision-language models to accommodate stylistic variations across devices, ensures robust adaptation to out-of-distribution scenarios.

Moreover, we validate our approach through experiments on two distinct levels. At the model level, we benchmark our model against both larger-scale vision-language models (Qwen3.5,122B-A10B, Gemini 3 Flash), as well as smaller baselines (Qwen3.5-4B, Qwen3 4B), demonstrating superior reference grounding accuracy despite our significantly smaller model size At the agent system level, evaluations against one-shot method ShowUI-Aloha and zero-shot baselines confirm YOSO-Agent's reliable end-to-end execution capabilities. Collectively, these results demonstrate that demonstration-based learning offers a practical pathway for deploying robust GUI Agents.

In summary, our contributions are as follows:

- We propose YOSO-Agent, an end-to-end GUI automation framework that leverages demonstration to enable reliable task execution, allowing rapid adaptation to different scenarios.
- We design a collaborative multi-agent architecture with three specialized agents: a Task Manager Agent for visual alignment; an Action Agent with a reference-guided grounding model for accurate localization; and a Reflection Agent for self-correction via adaptive backtracking strategies.
- We build a large-scale dataset pairing demonstration reference images with real-world observations, which grounds target localization across UI variations and facilitates accurate action execution by the learned model.
- Extensive experiments demonstrate that YOSO-Agent achieves significant performance gains over strong baselines, with improvements in grounding accuracy and end-to-end task success rate, validating the effectiveness of visual demons-tration-based learning for real-world GUI automation.

2 Related Work

Demonstration-Free GUI Automation Early frameworks relied on static structural representations such as webpage DOMs or accessibility trees. Subsequent HTML-based approaches like Mind2Web (Deng et al., 2023), WebAgent (Gur et al., 2023), and AutoWebGLM (Lai et al., 2024) employed fine-tuned language models to process structured representations before action prediction. These methods struggle on platforms lacking metadata, which motivated visual benchmarks like Multimodal-Mind2Web, VisualWebArena (Koh et al., 2024), and OSWorld (Xie et al., 2024). SeeAct (Zheng et al., 2024) demonstrated GPT-4V's feasibility for visual navigation, sparking a shift toward end-to-end multimodal agents. This paradigm expanded across platforms, with concurrent efforts addressing mobile interfaces (Hong et al., 2024; Lin et al., 2025; Yang et al., 2025; Wang et al., 2024a,b, 2025; Ye et al., 2025), visual web navigation (Long et al., 2025), and desktop automation (Agashe et al., 2025; Cristescu et al., 2025), with subsequent benchmarks extending coverage to cross-application and SaaS workflows (Li et al., 2026; Shi et al., 2026) and analyses of zero-shot robustness (Zhang et al., 2026c). Despite these advances, zero-shot methods require pre-training on massive trajectory corpora, a scalability barrier where agents fail to generalize to unseen UI variations (Liu et al., 2025). YOSO-Agent addresses this by enabling task-specific adaptation from a single demonstration, complementing pre-trained models without fine-tuning or large-scale data.

Demonstration-Based GUI Automation Demonstration-based methods extract textual references, falling into two categories. *Knowledge-based retrieval* aggregates demonstrations into knowledge bases, as seen in LearnAct (Liu et al., 2025), IFR-Agent (Wu et al., 2025), and KGRAG (Guan et al., 2025). *Single-reference interpretation* converts demonstrations into workflows, exemplified by ShowUI-Aloha (Zhang et al., 2026b) via planner-actor pipelines and (Li et al., 2025c) via step-by-step instruction generation with verifiers. While many approaches assume structured trajectories, recent benchmarks explore mining raw instructional videos via

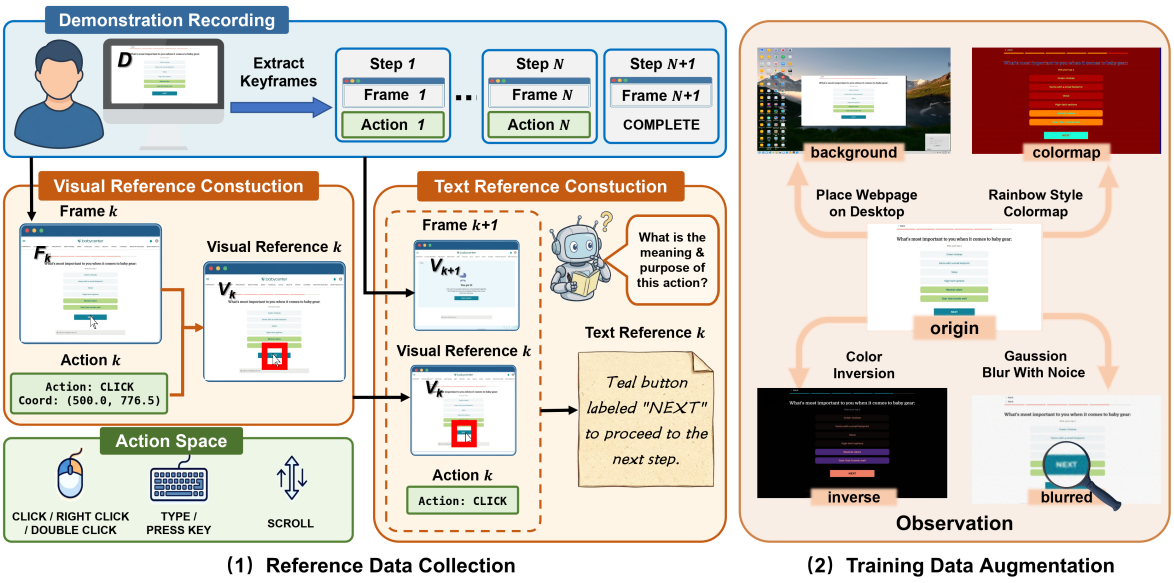

Figure 1: This figure illustrates the reference data construction pipeline and visual perturbation evaluation. (1)Left panel demonstrates Reference Data Collection: keyframes extracted from demonstrations based on action timestamps; visual references use bounding boxes to highlight target UI elements; text references describe action intent by analyzing frame transitions. (2) Right panel illustrates various visual augmentations used to enrich the training dataset. This strategy effectively boosts the model's grounding generalization under varied execution environments.

inverse dynamics, including VideoGUI (Lin et al., 2024), MONDAY (Jang et al., 2025), LearnGUI (Liu et al., 2025),AutoRPA(Chen et al., 2026) and GPA(Zhao et al., 2026b). However, textual references remain vulnerable to semantic drift under stylistic variations, as indirect descriptions lose the fine-grained visual details in original demonstrations. YOSO-Agent adopts single-reference interpretation but retains direct visual guidance throughout execution, ensuring robust task completion despite interface changes.

Visual Grounding for GUI Automation Visual grounding maps language intent to interaction coordinates, remaining a critical bottleneck for agent reliability. Current approaches span two categories: *coordinate generation* models predict positions from visual input via pretraining (Cheng et al., 2024; Qin et al., 2025; Gou et al., 2024; Yang et al., 2025), while *visual prompting* methods overlay markers on UI elements to guide attention (Yang et al., 2023). These deep learning techniques have superseded classical computer vision methods like SIFT or template matching, which rely on low-level pixel correlations and lack robustness to scale, rotation, and lighting variations (OMahony et al., 2019). Nevertheless, coordinate generation depends on text specifications that are often ambiguous for visually similar elements, and visual prompting relies on detection modules that compound errors upon failure. Recent test-time scaling via zoom-in (Luo et al., 2025) or reinforcement learning (Lu et al., 2026) still does not resolve the ambiguity of text-only specifications under domain shift (Zhao et al., 2026a; Mittal et al., 2026; Zhang et al., 2026a). In contrast, YOSO-Agent grounds actions using direct visual references, matching current interfaces against their demonstration appearance. This visual-to-visual alignment provides a robust spatial anchor, effectively mitigating the semantic ambiguity inherent in text-only approaches.

3 Demonstration Data Pipeline

3.1 Task Definition

GUI tasks require agents to navigate digital ecosystems to achieve specific objectives. Humans often demonstrate these tasks via screen recordings paired with operation sequences. We formalize this as a demonstration-guided decision-making process $\mathcal{P} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{D})$, where $\mathcal{S}$ denotes device states, $\mathcal{O}$ represents the observation space (visual screenshots), and $\mathcal{A}$ is the action space. The interaction function $\mathcal{F} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ governs state transitions, while $\mathcal{R}$ evaluates goal fulfillment. Crucially, $\mathcal{D}$ comprises human demonstrations (synchronized screen recordings and operation sequences) that serve as reference trajectories.

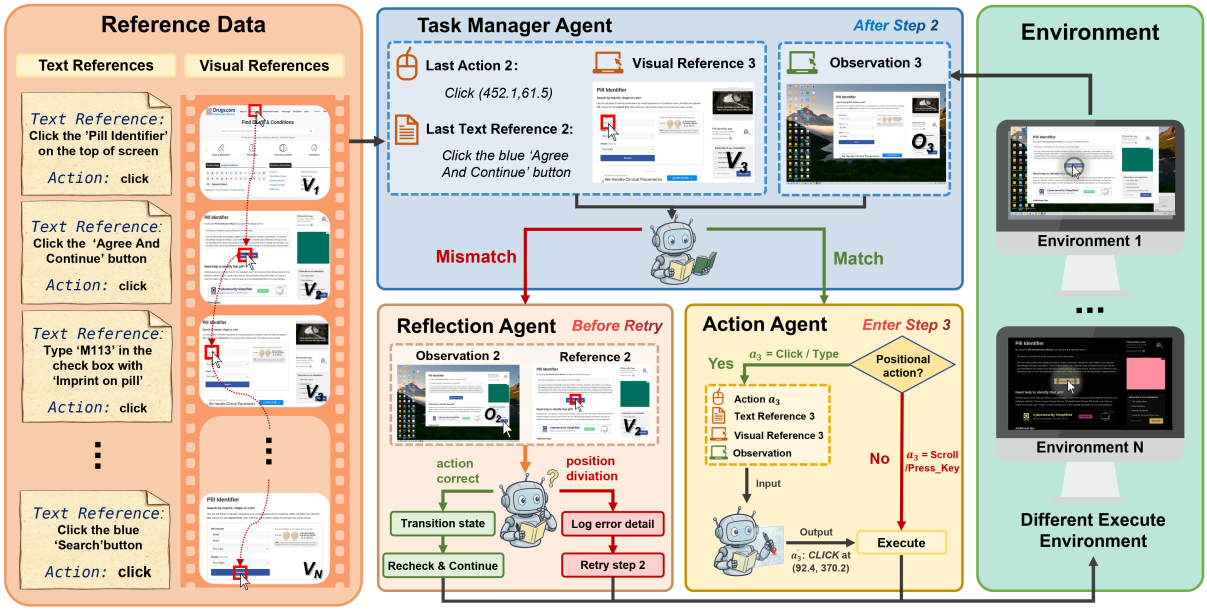

Figure 2: Overview of YOSO-Agent illustrated through Step 2 verification and Step 3 transition. The system orchestrates execution through three core components: (i) Task Manager Agent performing step alignment by fetching current step's reference data and verifying state matches; (ii) Reflection Agent analyzing execution failures by combining previous actions with past observations against references; (iii) Action Agent composing actions by reusing the demonstration's action type and generating coordinates when required.

This data directly informs step correspondence and decision-making. Specifically, $\mathcal{D}$ acts as a reference to verify if the current state matches the intended step, enabling policy-driven action execution upon alignment. However, stylistic variations like UI updates or resolution shifts render rigid imitation insufficient. Consequently, $\mathcal{D}$ also guides a reflection process upon misalignment, analyzing discrepancies to direct corrective actions. This mechanism ensures precise action grounding and reliable execution.

3.2 Reference Data Collection

To ensure high grounding accuracy, we transform the demonstration set $\mathcal{D}$ (synchronized videos and interaction sequences) into structured, step-level data for precise step management. We automatically extract stable pre- and post-action frames from $\mathcal{D}$ as keyframes without manual annotation. Formally, each trajectory comprises frames $\{F_1, \ldots, F_{n+1}\}$ and actions $\{a_1, \ldots, a_n\}$, where a_i transitions $F_i \rightarrow F_{i+1}$, capturing visual causality for one-shot learning.

From these sequences, we construct multi-modal references per step i. A **visual reference** $V_i = (F_i, B_i)$ is generated, where B_i is a red square centered at the click coordinate (x, y) of a_i, with side length $\epsilon \cdot W$ (W denotes image width, ϵ a scaling parameter). This marker highlights the target UI element for precise localization. Complementing this, a **text reference** T_i is produced by a vision-language model $\Psi : \mathcal{V} \times \mathcal{F} \times \mathcal{A} \rightarrow \mathcal{T}$. Processing (V_i, F_{i+1}, a_i), Ψ outputs a concise description capturing the target's visual characteristics and the action's intent, bridging low-level features with high-level objectives.

Integrating both references provides a comprehensive dual-mode guide. This approach enables the agent to robustly associate visual features with task goals, maintaining performance despite UI variations like layout shifts, color changes, or element repositioning, thereby significantly enhancing cross-environment generalization.

4 Methodology

YOSO-Agent operates as a closed-loop system leveraging demonstration data $\mathcal{D}$ to ensure reliable task execution in dynamic GUI environments. Its step correspondence mechanism relies on three specialized components. The Task Manager Agent evaluates alignment via a reward function R to determine whether the current observation matches the intended demonstration step. Upon successful alignment ($\mathcal{R} = 1$), the Action Agent follows the demonstrations guidance to output precise actions, performing high-accuracy localization using ref-

Algorithm 1: YOSO-Agent's Execution Workflow

Input: Demonstration $\mathcal{D} = \{(V_i, T_i, a_i)\}_{i=1}^{N}$, Initial observation O_1
Data: V_i: Visual references for step i;
T_i: Text references for step i;
a_i: Action demonstrated at step i;
a'_i: Executable action for step i;
O_i: Current observation at step i

```
i ← 0;
while i < N do
    ifmatch ← TaskManagerAgent(O_{i+1}, V_{i+1}, T_i, a_i);
    if ifmatch = 1 then
        i ← i + 1;
        a'_i ← ActionAgent(O_i, V_i, T_i, a_i);
        Execute a'_i and capture new observation O_{i+1};
    else
        if a_i = SCROLL then
            a'_i ← ActionAgent(O_i, V_i, T_i, a_i);
            Execute a'_i and capture new observation O_{i+1};
        else
            need_rollback ← ReflectionAgent(O_i, V_i, a'_i);
            if need_rollback then
                i ← i − 1;
            end
            Capture new observation O_i;
        end
    end
end
Task_success ← TaskManagerAgent(O_{n+1}, V_{n+1}, T_{n+1});
```

erence data. Conversely, if misalignment occurs ($\mathcal{R} = 0$), the Reflection Agent analyzes the discrepancy between the demonstration and current state, verifying execution results to guide corrections. This integrated design ensures the agent's execution flow remains strictly aligned with the demonstration across diverse real-world scenarios.

4.1 Task Manager Agent

The Task Manager Agent ensures execution strictly follows the logical progression of the provided demonstration $\mathcal{D}$. It must verify whether each real-time observation O_i aligns with the corresponding demonstration milestone. However, comparing O_{i+1} globally against all demonstration key frames is computationally inefficient and noisy. Instead, the agent tracks execution progress statefully, focusing exclusively on reference data relevant to the active step. Specifically, it evaluates alignment using the quadruple $(a_i, O_{i+1}, V_{i+1}, T_i)$, where a_i denotes the current action type, V_{i+1} serves as the visual goal-condition, and T_i provides the semantic instruction. A fine-tuned vision-language model verifies whether the new screen state O_{i+1} successfully reaches the intended milestone V_{i+1}. Upon validation, relevant reference data is dispatched to the Action Agent.

A critical refinement involves scroll operations. Due to device-specific UI variations, exact pixel displacements are unpredictable, and intermediate scroll states rarely match V_{i+1}. Consequently, a scroll misalignment does not immediately trigger the Reflection Agent. Instead, the Action Agent iteratively scrolls until O_{i+1} aligns with V_{i+1}. For other action types, persistent misalignment triggers Reflection Agent to determine corrective measures.

Action Type	Definition
CLICK$[x, y]$	Click at (x, y).
DOUBLE_CLICK$[x, y]$	Double click at (x, y).
RIGHT_CLICK$[x, y]$	Right click at (x, y).
SCROLL[dir, h]	Scroll dir by h pixels.
TYPE$[x, y,$ text$]$	Click (x, y) and type text.
PRESS_KEY[key]	Press key or shortcut.

Table 1: Definition of Atomic Actions in the Action Space $\mathcal{A}$.

4.2 Action Agent

The Action Agent synthesizes a complete executable action specification by integrating real-time observation O_i with reference data (V_i, T_i). Upon alignment confirmation, it inherits action type a_i and corresponding references to determine execution (see Table 1). Coordinate-free actions (e.g., `SCROLL`, `PRESS_KEY`) are directly replicated from a_i. For positioning-dependent interactions like `CLICK` or `TYPE`, grounding is required. Processing (a_i, O_i, V_i, T_i) aligns the observation with the action type, visual reference, and instruction, yielding executable specification a'_i. For instance, abstract intent a_i becomes `CLICK` at normalized coordinates (x, y) in a'_i, ensuring cross-device consistency.

To maintain reliability across environments, we train the grounding model on diverse visual styles. Rather than relying on pixel-level matching, it recognizes the underlying function and semantics of UI elements, accurately locating targets despite style variations. After execution, a new observation is captured and returned to the Task Manager for subsequent step verification.

4.3 Reflection Agent

The Reflection Agent acts as a diagnostic layer, activating upon spatial action misalignment. It in-

Table 2: Grounding accuracy performance (%) on ScreenSpot-v2-Reference and ScreenSpot-Pro-Reference under different conditions with appearance differences between Observation and Reference data. We evaluate on original images, color-inverted images, blurred images, rainbow-colormapped images, and desktop backgrounds. YOSO maintains robust performance across all conditions.

Dataset	Model(size)	Origin	Inverse	Blurred	Colormap	Background
ScreenSpot-v2	UI-TARS (7B)	51.59%	51.09%	52.35%	52.77%	17.45%
	Qwen3-VL (4B)	83.47%	81.71%	81.88%	80.29%	45.81%
	Qwen3.5 (4B)	98.74%	97.20%	96.00%	94.88%	55.20%
	Qwen3.5 (122B-A10B)	99.50%	99.08%	99.24%	98.57 %	84.38%
	gemini3-flash	93.84%	94.63%	92.45%	92.36%	84.82%
	UGround (7B)	89.77%	89.51%	91.44%	89.26%	43.79%
	YOSO-Model (4B)	**99.83%**	**99.74%**	**99.83%**	**99.74%**	**99.83%**
ScreenSpot-Pro	UI-TARS (7B)	29.60%	30.68%	29.98%	29.54%	1.77%
	Qwen3-VL (4B)	53.89%	51.61%	50.79%	49.65%	3.29%
	Qwen3.5 (4B)	68.2%	57.00%	60.27%	63.27%	13.6%
	Qwen3.5 (122B-A10B)	89.59%	86.36%	83.43%	82.65%	16.34%
	gemini3-flash	86.10%	85.00%	85.02%	84.60%	7.63%
	UGround (7B)	29.35%	27.77%	27.96%	26.19%	2.59%
	YOSO-Model (4B)	**95.12%**	**95.12%**	**85.20%**	**92.76%**	**92.68%**

spects the previous interaction point to determine whether failure stems from inaccurate grounding or delayed UI response. To diagnose, it analyzes triplet (O_i, V_i, a'_i) from step i, where a'_i is the executed specification. It renders executed coordinates from a'_i onto past observation O_i and visually compares this point against the target area in V_i.

Based on this comparison, it initiates one of two corrective paths. If coordinates deviate from the target box in V_i, a localization error is identified. The agent generates an error log triggering an automatic retry. Conversely, if coordinates align correctly, the interface is likely transitioning (e.g., rendering or loading). The system re-captures the screen to re-evaluate alignment. This prevents execution deadlocks and ensures adaptive recovery.

5 Experiments

We evaluate YOSO-Agent across varying UI complexities with two objectives: (1) measuring grounding precision under visual perturbations simulating real-world distribution shifts, and (2) assessing end-to-end capability in long-horizon tasks. Accordingly, we benchmark against existing GUI agents for task execution, and against both open-source and proprietary vision-language models for grounding accuracy.

5.1 Experiment Setup

5.1.1 Implementation Details

Our framework is implemented in PyTorch and trained on 8 NVIDIA A800 GPUs. YOSO-Agent comprises three functional modulesTask Manager Agent, Action Agent, and Reflection Agentall instantiated from a single fine-tuned model. The training procedure emphasizes the development of task decomposition and action execution capabilities.

For the Qwen3-VL-4B backbone, we employ LoRA with rank 8 and alpha 32. Training uses a cosine learning rate scheduler with initial rate 1×10^{-5}, batch size 64, and gradient accumulation over 2 steps. Visual augmentations including pixel-level noise and color transformations are applied to prevent overfitting.

5.1.2 Baselines

For comprehensive grounding evaluation, we compare YOSO-Agent's grounding module against a set of vision-language models, including UI-TARS-7B, Qwen3-VL-4B, Qwen3.5-4B, Qwen3.5-122B-A10B, and UGround-7B. Among these, several models possess substantially larger parameter counts than our 4B backbone, providing a particularly rigorous test of grounding precision under compact model constraints.

For end-to-end evaluation, we compare the full interactive pipeline of YOSO-Agent against existing GUI agents, including SeeClick, Fuyu-GUI, UIX(Qwen2VL), OS-ATLAS, UIPro(Qwen2VL), and ShowUI-Aloha. This comparison isolates the advantages of our agent design beyond standalone grounding capabilities.

5.1.3 Evaluation Metrics

Grounding evaluation is conducted on augmented dataset of ScreenSpot-Pro and ScreenSpot-v2,

Table 3: Performance comparison on Multimodal Mind2Web across three generalization scenarios. † denotes step-level accuracy reported in UIPro (Li et al., 2025a). Comparative evaluation with demonstrations between ShowUI Aloha and our YOSO-Agent.

Agent	Use Demo	Cross-Domain		Cross-Website		Cross-Task	
		Step SR	Full Task SR	Step SR	Full Task SR	Step SR	Full Task SR
SeeClick† (Cheng et al., 2024)	×	25.5	–	16.4	–	20.8	–
Fuyu-GUI† (Nong et al., 2024)	×	15.6	–	12.2	–	11.7	–
UIX† (Liu et al., 2024)	×	38.2	–	31.0	–	34.9	–
OS-ATLAS† (Wu et al., 2024)	×	36.7	–	35.7	–	37.2	–
UIPro† (Li et al., 2025a)	×	48.4	–	43.6	–	45.5	–
ShowUI Aloha (Zhang et al., 2026b)	✓	68.74	21.67	73.11	14.14	72.28	26.98
YOSO-Agent (Ours)	✓	**98.57**	**93.04**	**97.17**	**86.87**	**98.36**	**92.86**

Table 4: Ablation study of training data on ScreenSpot-v2-Reference and Multimodal Mind2Web. Grounding accuracy (%) on ScreenSpot-v2-Reference under five visual conditions. Step SR and Full Task SR reported for Multimodal Mind2Web. Four training configuration : (1) baseline; (2) Visual Reference; (3) adding augmented observations; (4) combining both.

Base Model	with Visual Reference	with Augment Data	ScreenSpot-v2					Multimodal Mind2Web	
			Origin	Inverse	Blurred	Colormap	Background	Step SR	Full Task SR
Qwen3-VL-4B	×	×	83.47%	81.71%	81.88%	80.29%	45.81%	45.20%	7.83%
	✓	×	99.33%	99.33%	99.33%	98.99%	14.93%	61.45%	23.49%
	×	✓	96.98%	95.39%	95.72%	94.88%	27.10%	67.67%	28.44%
	✓	✓	**99.83%**	**99.74%**	**99.83%**	**99.74%**	**99.83%**	**98.32%**	**92.17%**

where a red square centered at each ground-truth bounding box is overlaid on the screenshot to serve as the "Visual Reference". These modified benchmarks are denoted as ScreenSpot-v2-Reference and ScreenSpot-Pro-Reference in subsequent sections.

We adopt **Grounding Accuracy** for offline grounding tasks, requiring predicted coordinates to fall within the target element's ground-truth bounding box. For multi-step scenarios, we measure **Full Task Success Rate (SR)**, defined as the percentage of tasks where agent completes without errors. This dual-metric approach evaluates both grounding precision and high-level reasoning.

5.2 Main Results

5.2.1 Grounding Evaluation under Visual Variants

Table 2 presents grounding performance on ScreenSpot-v2-Reference and ScreenSpot-Pro-Reference under five visual variants. On ScreenSpot-v2-Reference, YOSO-Model achieves near-perfect accuracy across all variants. On the more challenging ScreenSpot-Pro-Reference, YOSO-Model maintains a clear lead with 95.12% on *Origin* and 92.68% on *Background*, while the strongest baseline reaches only 68.2% and 13.6%.

Substantial gaps emerge between YOSO-Model and both small-scale VLMs and specialized grounding architectures. Qwen3-VL-4B and Qwen3.5-VL-4B suffer severe degradation under perturbations, particularly *Background* (45.81% and 13.6%). Specialized models like UI-TARS-7B and UGround-7B show limited utilization of visual references, struggling on ScreenSpot-Pro-Reference (29.60% and 29.35% on *Origin*). Against Qwen3.5-VL-122B-A10B and closed-source Gemini-3-Flash, YOSO-Model maintains consistent superiority despite smaller scale, with pronounced gaps on ScreenSpot-Pro-Reference (95.12% vs. 86.10% on *Origin*; 92.68% vs. 7.63% on *Background*).

5.2.2 End-to-End Task Execution Evaluation

To assess complete agent performance, we evaluate end-to-end task completion on Multimodal-Mind2Web across three scenarios. As shown in Table 3, existing models without demonstration guidance exhibit limited Step SR (11.7%–48.4%) and omit full-task metrics due to poor state coherence.

When leveraging demonstrations, a sharp divergence emerges: ShowUI Aloha achieves respectable Step SR (72%–77%) yet its Full Task SR collapses to 14.14%–26.98% due to error compounding. Conversely, YOSO-Agent sustains high performance across both metrics (Step SR

Table 5: Ablation study of reference data on ScreenSpot-v2-Reference and ScreenSpot-Pro-Reference. We compare three reference types to ground with: Visual Reference (visual demonstrations), Text Reference (textual instructions), and Visual Reference + Text Reference (both). Grounding accuracy (%) reported for each configuration.

Dataset	Model(size)	Visual Reference	Text Reference	Visual Reference + Text Reference
ScreenSpot-v2	UI-TARS (7B)	50.42%	48.07%	51.59%
	Qwen3-VL (4B)	83.81%	84.78%	83.47%
	Qwen3.5 (4B)	98.32%	84.48%	98.74%
	Qwen3.5 (122B-A10B)	99.50%	**92.01**%	99.50%
	UGround (7B)	51.76%	85.91%	89.77%
	YOSO-Model (4B)	**99.83**%	87.04%	**99.92**%
ScreenSpot-Pro	UI-TARS (7B)	30.17%	21.57%	29.60%
	Qwen3-VL (4B)	56.93%	50.49%	53.89%
	Qwen3.5 (4B)	65.7%	24.86%	68.2%
	Qwen3.5 (122B-A10B)	89.81%	**66.01**%	89.59%
	UGround (7B)	7.65%	26.69%	29.35%
	YOSO-Model (4B)	**92.98**%	41.74%	**92.66**%

Table 6: Ablation study of Reflection Agent on Multimodal Mind2Web. Full Task Success Rate reported. Comparison between YOSO-Agent with and without Reflection Agent validates the contribution of Reflection Agent.

Configuration	Cross Domain	Cross Website	Cross Task
w.o. Reflection Agent	89.66%	88.10%	82.83%
with Reflection Agent	93.04%	92.86%	86.87%

97.17%–98.57%, Full Task SR 86.87%–93.06%), surpassing ShowUI Aloha by 66.37%, 72.73%, and 65.88% across Cross Domain, Website, and Task settings. This confirms our pipeline mitigates error compounding through robust step-wise alignment.

5.3 Ablation Study

5.3.1 Ablation Study of Training Data

Table 4 shows ablation results across four training configurations. Visual Reference dramatically improves grounding accuracy on ScreenSpot-v2-Reference, from 83.47% to 99.33% on *Origin* and from 45.81% to 99.83% on *Background*, while boosting Step SR from 45.20% to 61.45% and Full Task SR from 7.83% to 23.49% on Multimodal Mind2Web. Stylized augmentation further enhances robustness, particularly on *Background* where accuracy jumps from 14.93% to 99.83% when combined. This confirms that both strategies yielding complementary benefits.

5.3.2 Ablation Study of Reference Input

Table 5 analyzes reference modality contributions. Combining visual and text reference consistently outperforms single-modality settings. However, a significant asymmetry emerges: models maintain robust performance with visual reference alone, whereas textual reference alone leads to substantial degradation. This indicates visual information serves as a more reliable signal for precise localization, while textual descriptions struggle to capture spatial positions accurately.

5.3.3 Ablation Study of Reflection Agent

To validate the self-correction mechanism, we compare YOSO-Agent with and without the Reflection Agent on Multimodal Mind2Web across three generalization scenarios. As summarized in Table 6, the complete system consistently outperforms the variant without Reflection. The Reflection Agent enables iterative self-verification to identify and correct erroneous actions, thereby improving task completion rates in challenging scenarios with different applications or task types.

6 Conclusion

We present YOSO-Agent, a visual grounding framework enabling precise action execution across diverse UI scenarios. Experiments demonstrate near-perfect grounding accuracy under substantial visual perturbations, outperforming strong baselines. End-to-end evaluations confirm precise grounding is the bottleneck for task completion, which our design addresses by bridging high-level reasoning with low-level execution, as validated by ablations showing that our training curriculum and multi-modal reference integration jointly drive the robustness. Overall, YOSO-Agent establishes a new standard for reliable, vision-centric agent grounding in dynamic real-world environments.

Limitations

While YOSO-Agent demonstrates strong performance in demonstration-based GUI automation, several limitations remain to be addressed in future work. First, our framework requires an initial human demonstration for each novel task, which may introduce a non-trivial upfront cost for users compared to fully zero-shot methods. Although a single demonstration suffices for reliable execution, extending the framework to leverage historical demonstrations across tasks could further reduce this overhead. Second, the collaborative multi-agent architecture, while effective for robustness, incurs additional latency due to inter-agent communication and sequential validation steps; optimizing this pipeline for real-time responsiveness is a promising direction. Third, despite the diversity of our constructed dataset, it does not exhaustively cover all UI variations across every operating system and application version, particularly emerging or highly specialized enterprise software. Finally, as with all vision-language models, our fine-tuned agents may occasionally exhibit hallucination or grounding errors under extreme visual perturbations; while the Reflection Agent mitigates such failures, perfect recovery is not guaranteed for arbitrarily long task sequences.

Ethics Statement

This research was conducted with strict adherence to ethical guidelines for AI systems interacting with digital interfaces. All demonstrations and screenshots used for dataset construction and evaluation were obtained from publicly available applications or synthetic environments, with no access to private user data or sensitive personal information. The proposed agent framework is intended to augment human productivity through supervised automation; we emphasize that deployment in real-world settings should require explicit user consent, transparent logging of executed actions, and appropriate access controls to prevent unauthorized operations. We acknowledge the potential risks of automated GUI agents, including unintended interface modifications or execution of sensitive transactions, and advocate for human-in-the-loop oversight for high-stakes tasks. All experiments were designed to minimize environmental impact, and model training was performed using efficient fine-tuning strategies to reduce computational carbon footprint.

References

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. 2025. Agent s2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*.

Minghao Chen, Xinyi Hu, Zhou Yu, and Yufei Yin. 2026. Autorpa: Efficient gui automation through llm-driven code synthesis from interactions. *Preprint*, arXiv:2605.21082.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332.

Horia Cristescu, Charles Park, Trong Canh Nguyen, Sergiu Talmacel, Alexandru-Gabriel Ilie, and Stefan Adam. 2025. Ui-cube: Enterprise-grade computer use agent benchmarking beyond task accuracy to operational reliability. *arXiv preprint arXiv:2511.17131*.

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

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*.

Ziyi Guan, Jason Chun Lok Li, Zhijian Hou, Pingping Zhang, Donglai Xu, Yuzhi Zhao, Mengyang Wu, Jinpeng Chen, Thanh-Toan Nguyen, Pengfei Xian, Wenao Ma, Shengchao Qin, Graziano Chesi, and Ngai Wong. 2025. KG-RAG: Enhancing GUI agent decision-making via knowledge graph-driven retrieval-augmented generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5396–5405, Suzhou, China. Association for Computational Linguistics.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and 1 others. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14281–14290.

Yunseok Jang, Yeda Song, Sungryull Sohn, Lajanugen Logeswaran, Tiange Luo, Dong-Ki Kim, Kyunghoon Bae, and Honglak Lee. 2025. Scalable video-to-dataset generation for cross-platform mobile agents.

In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8604–8614.

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

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and 1 others. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306.

Hongxin Li, Jingran Su, Jingfan Chen, Zheng Ju, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2025a. Uipro: Unleashing superior interaction capability for gui agents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1613–1623.

Jinchao Li, Yunxin Li, Chenrui Zhao, Zhenran Xu, Baotian Hu, and Min Zhang. 2026. Windowsworld: A process-centric benchmark of autonomous gui agents in professional cross-application environments. *Preprint*, arXiv:2604.27776.

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

Yanda Li, Chi Zhang, Wenjia Jiang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*.

Yinheng Li, Hailey Hultquist, Justin Wagle, and Kazuhito Koishida. 2025c. Instruction agent: Enhancing agent with expert demonstration. *arXiv preprint arXiv:2509.07098*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Qinchen Wu, Mingyi Yan, Zhengyuan Yang, Lijuan Wang, and Mike Zheng Shou. 2024. Videogui: A benchmark for gui automation from instructional videos. In *NeurIPS*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2025. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19498–19508.

Tony Lindeberg. 2012. Scale invariant feature transform.

Guangyi Liu, Pengxiang Zhao, Liang Liu, Zhiming Chen, Yuxiang Chai, Shuai Ren, Hao Wang, Shibo He, and Wenchao Meng. 2025. Learnact: Few-shot mobile gui agent with a unified demonstration benchmark. *arXiv preprint arXiv:2504.13805*.

Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. 2024. Harnessing webpage uis for text-rich visual understanding. *arXiv preprint arXiv:2410.13824*.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. 2025. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. *arXiv preprint arXiv:2508.09736*.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, and 1 others. 2026. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 21, pages 17608–17616.

Tiange Luo, Lajanugen Logeswaran, Justin Johnson, and Honglak Lee. 2025. Visual test-time scaling for gui agent grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19989–19998.

Himangi Mittal, Gaurav Mittal, Nelson Daniel Troncoso, and Yu Hu. 2026. See, point, refine: Multi-turn approach to gui grounding with visual feedback. *Preprint*, arXiv:2604.13019.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, and 1 others. 2025. Gui agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22522–22538.

Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. Mobileflow: A multimodal llm for mobile gui agent. *arXiv preprint arXiv:2407.04346*.

Niall OMahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. 2019. Deep learning vs. traditional computer vision. In *Science and information conference*, pages 128–144. Springer.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Kean Shi, Zihang Li, Tianyi Ma, Zengji Tu, Jialong Wu, Xinbo Xu, Qingyao Yang, Ruoyu Wu, Weichu Xie, Ming Wu, Jason Zeng, Michael Heinrich, Elvis Zhang, Liang Chen, Kuan Li, and Baobao Chang. 2026. Saas-bench: Can computer-use agents leverage real-world saas to solve professional workflows? *Preprint*, arXiv:2605.15777.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, and 1 others. 2025. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems*, 37:2686–2710.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024b. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. 2025. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*.

Zheng Wu, Heyuan Huang, Yanjia Yang, Yuanyi Song, Xingyu Lou, Weiwen Liu, Weinan Zhang, Jun Wang, and Zhuosheng Zhang. 2025. Quick on the uptake: Eliciting implicit intents from human demonstrations for personalized mobile-use agents. *arXiv preprint arXiv:2508.08645*.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and 1 others. 2024. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2025. Aria-ui: Visual grounding for gui instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22418–22433.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, and 1 others. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*.

Borui Zhang, Bo Zhang, Bo Wang, Wenzhao Zheng, Yuhao Cheng, Liang Tang, Yiqiang Yan, Jie Zhou, and Jiwen Lu. 2026a. Bami: Training-free bias mitigation in gui grounding. *Preprint*, arXiv:2605.06664.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

Yichun Zhang, Xiangwu Guo, Yauhong Goh, Jessica Hu, Zhiheng Chen, Xin Wang, Difei Gao, and Mike Zheng Shou. 2026b. Showui-aloha: Human-taught gui agent. *arXiv preprint arXiv:2601.07181*.

Yuzhe Zhang, Xianwei Xue, Xingyong Wu, Mengke Chen, Chen Liu, Xinran He, Run Shao, Feiran Liu, Huanmin Xu, Qiutong Pan, and Haiwei Wang. 2026c. Don't act blindly: Robust gui automation via action-effect verification and self-correction. *Preprint*, arXiv:2604.05477.

Haoren Zhao, Tianyi Chen, and Zhen Wang. 2026a. Windesktop: A benchmark for robust gui grounding in complex multi-window desktop environments. *Preprint*, arXiv:2605.16402.

Zirui Zhao, Jun Hao Liew, Yan Yang, Wenzhuo Yang, Ziyang Luo, Doyen Sahoo, Silvio Savarese, and Junnan Li. 2026b. Gpa: Learning gui process automation from demonstrations. *Preprint*, arXiv:2604.01676.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*.

A Implementation Details of YOSO-Agent

A.1 Model Architecture

The proposed YOSO model is built upon the Qwen3-VL multimodal foundation, incorporating task-specific adaptations tailored for visual grounding. The core architectural configurations are detailed below.

Vision Encoder. The visual backbone comprises a 24-layer Vision Transformer with a hidden dimension of 1,024, an intermediate feed-forward size of 4,096, and 16 attention heads. Input imagery is tokenized using 16×16 patch embeddings, augmented with a spatial merge factor of 2 and a temporal patch size of 2 to support video understanding. Visual features are linearly projected to a dimensionality of 2,560 to align with the language model's hidden state space. During training, images undergo dynamic resizing constrained by a maximum pixel budget of 1,440,000 (capped at a 1600×900 resolution), and the model operates within a context window of 8,192 tokens.

Parameter-Efficient Adaptation. We employ Low-Rank Adaptation (LoRA) for efficient model tuning, configuring it with rank $r = 8$ and scaling factor $\alpha = 32$, yielding an effective scaling ratio of $\alpha/r = 4$. A dropout probability of 0.05 is applied to the low-rank matrices. Following the default target module configuration in the Swift training framework for Qwen3-VL, LoRA adapters are injected exclusively into the query (W_q) and value (W_v) projection layers of the Transformer's self-attention blocks, while the vision encoder and cross-modal projection layers remain frozen.

Multimodal Feature Alignment. Rather than introducing an external cross-modal projector, we leverage the native multimodal bridge inherent to Qwen3-VL. The visual encoder's output dimension (2,560) natively matches the hidden size of the language model, enabling seamless visual-textual feature fusion without introducing additional trainable parameters.

Text-Based Coordinate Decoding. Our model formulates visual grounding as a unified text generation task. Spatial coordinates are serialized as zero-padded strings in the fixed format `[xxx.x, yyy.y]`, where leading zeros are applied to integer positions to ensure consistent tokenization and parsing. These formatted sequences are generated autoregressively by the language model alongside natural language responses. To enhance localization precision, we employ a digit-weighted cross-entropy loss that assigns higher penalties to numeric tokens within the coordinate strings. This design intrinsically unifies the grounding objective with the standard autoregressive language modeling loss, obviating the need for specialized decoding architectures.

A.2 Training Data Curation Details

A.2.1 Dataset Selection and Characteristics

We construct a heterogeneous training corpus comprising four complementary datasets to ensure comprehensive coverage of GUI environments:

- **Multimodal Mind2Web** (Deng et al., 2023): 7,775 long-scroll web screenshots with sequential action annotations. Selected for its rich task dependencies and diverse web interfaces requiring scrolling interactions.
- **OmniAct** (Kapoor et al., 2024): 5,362 desktop application screenshots (mean resolution 2584×1563) with precise bounding box annotations, providing high-fidelity professional software interfaces distinct from web DOM structures.
- **WebDirect (U-Ground)**(Gou et al., 2024): 148,416 point-annotated web grounding samples. Leveraged to enhance visual diversity across website designs, with annotations concentrated on interactive elements.
- **GroundCUA**: 157,832 bounding box annotations from 2,194 desktop application screenshots (7-Zip, Natron, RStudio). Distinguished by its focus on **compact UI elements** (mean dimensions 87.5×29.7 pixels), enabling fine-grained localization of small controls prevalent in professional software.

A.2.2 Initial Data Processing and Augmentation

Stage 1: Coordinate Grounding Pre-training. Each source image from Mind2Web and OmniAct undergoes systematic augmentation to prevent positional shortcut learning. We apply four transformations: color inversion, colormap conversion, resolution reduction ($0.7\times$), and identity mapping, yielding 52,548 total samples. These perturbations force the model to rely on structural rather than chromatic cues. Processed images are embedded onto synthetic desktop backgrounds with random

Configuration	Stage 1 (Grounding)	Stage 2 (Matching)
Optimization Objective	Supervised coordinate regression	Supervised classification + grounding
Training Samples	52,548	240,000
Model Initialization	Qwen3-VL-4B / Qwen3.5-4B	Stage 1 checkpoint
Learning Rate	3×10^{-5}	3×10^{-5}
LR Scheduler	Cosine	Cosine
Warmup Steps	0	0
Training Epochs	1	1
Global Batch Size	40	40
Per-device Batch Size	5	5
Gradient Accumulation Steps	1	1
Maximum Sequence Length	32,768	32,768
Precision	BF16	BF16
Trainable Parameters	LoRA	LoRA
Hardware Configuration	$8\times$ A800 80GB	$8\times$ A800 80GB

scaling ($0.4\times$–$1.0\times$) and positioning, ensuring target bounding boxes remain within screen bounds while allowing partial window visibility. Reference images display the target region marked by a red square (5-pixel width, side length 2% of image width) centered on the bounding box. For Mind2Web's longitudinal screenshots, we enforce vertical misalignment between observation windows and reference crops to eliminate global positional priors.

Stage 2: Visual Matching and Recognition. This stage employs a curriculum learning approach with stratified difficulty. From WebDirect and GroundCUA (305K base samples), we generate resolution variants ($0.4\times$–$0.8\times$ and $1.25\times$–$2.5\times$) placed on desktop backgrounds, creating 600K augmented instances. Using the Stage 1 model (temperature 0.7, 5 rollouts per sample), we filter for informative cases, retaining 75K challenging positives where the model exhibits uncertainty, supplemented with 45K Mind2Web positives generated by extracting multiple observation windows per long screenshot.

For negative sampling, we construct two tiers: (1) Easy negatives (60K) via random cross-pairing of mismatched WebDirect images; (2) Hard negatives (60K from Mind2Web) comprising disjoint regions (observation and reference crops from the same screenshot with non-overlapping vertical spans) and intersecting exclusions (overlapping vertical spans where the target lies within the reference crop but outside the observation window). The final training set comprises 240K balanced samples (120K positive, 120K negative).

A.3 Training Procedure and Hyperparameters

We adopt a progressive two-stage training paradigm that incrementally increases task complexity. The second stage builds upon the previous, transitioning from pure coordinate regression to binary decision-making.

Stage 1: Coordinate Grounding Pre-training. This stage employs supervised fine-tuning (SFT) to establish the model's foundational spatial localization capability. The optimization objective minimizes the L_1 loss between predicted and ground-truth normalized coordinates. We initialize from the base multimodal model and train with full parameter updates. The model learns to interpret visual references (red-square markers) and map them to precise screen coordinates within the desktop background context. Training employs comprehensive data augmentation to ensure robustness against visual perturbations. We use cosine learning rate decay with a linear warmup period, and validate on a held-out 5% subset to monitor overfitting. The training process converges when validation loss plateaus, typically within 3 epochs.

Stage 2: Visual Matching and Recognition. The second stage introduces binary classification capabilities alongside grounding. We continue with full-parameter supervised fine-tuning, but modify the optimization objective to handle both match/no-match decisions and conditional coordinate prediction. This stage implements curriculum learning: initially, the model trains on easy negatives to establish baseline discrimination, then progressively incorporates hard negatives (spatially challenging

cases from Mind2Web). The loss function combines binary cross-entropy for the matching decision and coordinate regression. We maintain consistent global batch sizing but extend training to accommodate the larger dataset (240K samples), using validation accuracy on a stratified validation set to determine convergence.

A.4 Evaluation Protocols

We evaluate our system on two complementary benchmarks that assess distinct yet critical capabilities: fine-grained visual grounding under UI variations and long-horizon task execution in web environments.

A.4.1 Benchmarks and Datasets

Visual Grounding Benchmark. For evaluating reference-based localization robustness, we construct a large-scale evaluation suite based on ScreenSpot-v2 and ScreenSpot-Pro (Li et al., 2025b), comprising 2,773 test instances across diverse platforms and device form factors.

ScreenSpot-v2 provides 1,192 samples spanning mobile (Android/iOS), desktop (Windows/-macOS), and web platforms, offering broad coverage of consumer application interfaces. In contrast, ScreenSpot-Pro contributes 1,581 instances focused exclusively on professional desktop software (macOS/Windows/Linux), featuring significantly more challenging localization targets. As illustrated in Table 7, Pro targets occupy merely 0.065% of screen area on average (median 0.036%) compared to 2.01% (median 0.72%) in v2, presenting a 30× reduction in target size that approximates real-world scenarios involving small control elements like toolbar buttons and menu items.

Task Execution Benchmark. For assessing end-to-end web automation capabilities, we employ the Mind2Web (Deng et al., 2023) test suite, encompassing 728 tasks totaling 4,290 actionable steps across domain, task, and website splits. This benchmark evaluates long-horizon planning with an average task length of 5.89 steps (max 22), requiring agents to handle complex action sequences including navigation (CLICK: 82.12%), text input (TYPE: 13.59%), and selection operations (SELECT: 4.29%).

Visual Reference Construction. To simulate real-world deployment where agents must locate targets from demonstration references, we synthesize query images by overlaying a red square marker at each target's bounding box center. The marker size defaults to 2% of image width, with an adaptive reduction to 1% for extreme aspect ratios (e.g., dual-monitor configurations in ScreenSpot-Pro) to prevent marker occlusion of compact targets.

Evaluation Metric. We adopt the standard accuracy metric: a prediction is considered correct if the predicted coordinate falls within the ground-truth bounding box. This strict criterion ensures that agents achieve pixel-precise localization rather than approximate region identification, which is critical for reliable UI automation.

A.4.2 Task Execution Evaluation Protocol

Hierarchical Observation Processing. Given Mind2Web's longitudinal screenshots (often exceeding 10,000 pixels in height), we implement a sliding-window observation mechanism. Each webpage is preprocessed into sequential blocks of 1280×1000 pixels with a stride of 800 pixels, creating 200-pixel vertical overlaps between adjacent blocks. This overlapping design ensures UI elements spanning block boundaries remain fully visible in at least one observation window.

Interactive Evaluation Loop. The evaluation proceeds as follows: (1) For each step, the agent receives the topmost block as initial observation; (2) If the target element is not visible (determined by ground-truth bbox intersection), the agent executes a scroll action (800px downward) to reveal the next block; (3) This scroll-and-observe cycle continues until the target becomes visible or maximum scroll limits are reached; (4) Once visible, the agent performs the required interaction (click/type/select).

The metric *Step Success Rate* counts a step as successful only if the correct action is performed on the correct target within minimal scrolling (ideally $n = 0$). The composite *Task Success Rate* requires all steps within a task to succeed. This protocol rigorously tests the agent's ability to integrate visual grounding with exploration strategies, as excessive scrolling or misidentification of off-screen elements immediately penalizes performance.

Action-Specific Handling. TYPE and SELECT actions require additional value prediction (present in 17–20% of steps). The evaluation checks both target localization accuracy and correct value generation, with any semantic deviation counting as a failure regardless of coordinate precision.

Table 7: Evaluation Dataset Characteristics

Dataset	Samples	textbfPlatforms	Form Factor	Mean Area Ratio
ScreenSpot-v2	1,192	Android, iOS, Web, macOS, Win	Mobile/Tablet/Desktop	2.01%
ScreenSpot-Pro	1,581	macOS, Windows, Linux	Desktop-only	0.065%
Mind2Web (domain)	2,799 actions (503 tasks)	Web	Desktop	0.955%
Mind2Web (task)	855 actions (126 tasks)	Web	Desktop	0.829%
Mind2Web (website)	636 actions (99 tasks)	Web	Desktop	1.124%

Table 8: Grounding Accuracy (%) under Observation Perturbations. *Origin* denotes unperturbed inputs; single perturbations isolate individual distortions; composite perturbations embed observations on desktop backgrounds with simultaneous stylistic transformations. All **YOSO-Model** variants are **initialized and fine-tuned from** their corresponding backbones.

Dataset	Model (Backbone)	Origin	Blur+Noise	Colormap	Inverse	On Desktop	Desktop +Colormap	Desktop +Inverse
ScreenSpot-v2	YOSO-Model (Qwen3-VL-4B)	99.83%	99.83%	99.74%	99.74%	99.83%	99.58%	99.83%
	YOSO-Model (Qwen3.5-4B)	99.92%	99.50%	99.66%	99.75%	99.92%	99.66%	99.83%
ScreenSpot-Pro	YOSO-Model (Qwen3-VL-4B)	95.12%	85.20%	92.76%	95.12%	92.68%	87.90%	89.87%
	YOSO-Model (Qwen3.5-4B)	94.05%	82.16%	87.92%	93.49%	88.30%	82.48%	87.54%

B Additional Experimental Results

B.1 Extented Backbone Study of Qwen3.5-4B

B.1.1 Grounding Evaluation

Table 8 establishes baseline robustness under single and composite observation perturbations. On ScreenSpot-v2, YOSO-Model(Qwen3.5-4B) achieves marginally superior origin accuracy (99.92% vs 99.83%) while maintaining comparable resilience across all distortion types, with performance remaining above 99.50% even under aggressive perturbations. This demonstrates that the enhanced training paradigm successfully preserves coarse-grained localization stability while upgrading the underlying architecture.

ScreenSpot-Pro presents a more challenging scenario due to diminutive target elements (mean area ratio 0.065%) in complex professional interfaces. While YOSO-Model(Qwen3.5-4B) shows modest degradation relative to YOSO-Model(Qwen3-VL-4B) under certain perturbations, the absolute performance remains remarkably high given the significantly enhanced fine-grained localization capability. Notably, under blur+noise conditions, the degradation is merely 3.04 percentage points (82.16% vs 85.20%)—a modest cost considering the architectural upgrade from a general-purpose vision encoder to one specifically optimized for high-resolution GUI understanding. The training curriculum's emphasis on chromatic augmentation manifests in the exceptional color inversion performance (93.49%), where the degradation is only 1.63 percentage points from the previous generation despite the enhanced encoder's greater sensitivity to photometric variations.

The composite perturbation results reveal the most demanding test: when observations are embedded on desktop backgrounds and simultaneously subjected to photometric transformations. Even in this adversarial scenario, YOSO-Model(Qwen3.5-4B) maintains respectable performance—82.48% for desktop+colormap and 87.54% for desktop+inverse—representing controlled degradation given the substantial increase in model complexity. The 5.42 percentage point gap on desktop+colormap, while notable, must be contextualized against the architectural leap: the enhanced encoder's fine-grained spatial sensitivity enables superior localization on clean, high-resolution inputs (as evidenced by ScreenSpot-Pro origin performance), and the training paradigm successfully constrains the robustness degradation to single-digit percentages even under compound domain shifts that would catastrophically impact less sophisticated architectures.

Table 9 provides compelling evidence for the efficacy of the training paradigm. On ScreenSpot-Pro, YOSO-Model(Qwen3.5-4B) demonstrates substantial gains under reference-side blur+noise (+2.55%), precisely the scenario where low-quality references (compressed thumbnails, motion-blurred captures) are prevalent in real-world deployments. This improvement is particularly significant given that the enhanced architecture might have been expected to suffer greater degradation under texture-level corruption. The training curriculum's balanced augmentation strategy—combining photometric distortions, resolution scaling, and syn-

Table 9: Grounding Accuracy (%) under Reference Perturbations. Relative improvement of YOSO-Model(Qwen3.5-4B) over YOSO-Model(Qwen3-VL-4B) shown in parentheses. Notable gains on ScreenSpot-Pro under blur+noise (+2.55%) suggest enhanced low-quality reference resilience, offset by sensitivity to context shifts.

Dataset	Model	Blur+Noise	Colormap	Desktop	Inverse
ScreenSpot-v2	YOSO-Model(Qwen3-VL-4B)	97.99%	99.08%	99.16%	98.91%
	YOSO-Model(Qwen3.5-4B)	98.91% (+0.92%)	99.33% (+0.25%)	99.16% (0.00%)	99.25% (+0.34%)
ScreenSpot-Pro	YOSO-Model(Qwen3-VL-4B)	78.98%	95.22%	92.29%	95.29%
	YOSO-Model(Qwen3.5-4B)	81.53% (+2.55%)	93.74% (-1.48%)	89.12% (-3.17%)	94.69% (-0.60%)

Table 10: Performance comparison on Multimodal Mind2Web End-to-End tasks across three generalization scenarios (Domain, Task, Website), reporting Step Success Rate (Step SR) and Full Task Success Rate (Task SR). Comparison between YOSO-Model variants with and without Reflection Agent validates the contribution of reflection-based self-correction. Absolute improvements shown in parentheses.

		Step Success Rate			Full Task Success Rate		
Split	Model	w/o Refl.	w/ Refl.	Improv.	w/o Refl.	w/ Refl.	Improv.
Domain	YOSO-Model(Qwen3-VL-4B)	97.82%	98.57%	+0.75%	89.66%	93.04%	+3.38%
	YOSO-Model(Qwen3.5-4B)	98.64%	98.93%	+0.29%	92.84%	94.43%	+1.59%
Task	YOSO-Model(Qwen3-VL-4B)	97.42%	98.36%	+0.94%	88.09%	92.86%	+4.77%
	YOSO-Model(Qwen3.5-4B)	98.71%	98.95%	+0.24%	92.86%	93.65%	+0.79%
Website	YOSO-Model(Qwen3-VL-4B)	95.91%	97.17%	+1.26%	82.83%	86.87%	+4.04%
	YOSO-Model(Qwen3.5-4B)	96.86%	96.86%	0.00%	88.89%	88.89%	0.00%

thetic noise injection—effectively regularizes the model against reference-side degradation while preserving its fine-grained discrimination capability.

The asymmetric robustness profile (superior handling of blur+noise, modest degradation on context shifts) represents a favorable trade-off for practical GUI grounding systems: real-world failures are more frequently caused by low-quality reference captures than by radical desktop theme transformations. The training paradigm thus achieves its design objective of enhancing practical deployment resilience while accepting controlled degradation on edge-case composite perturbations that remain challenging for the broader research community.

B.1.2 End2End Task Evaluation

Table 10 presents an ablation study of the Reflection Agent on Multimodal Mind2Web end-to-end tasks across three generalization scenarios: Domain (unseen websites within seen domains), Task (unseen task types), and Website (completely unseen websites). The results demonstrate consistent gains from reflection-based self-correction, with the magnitude of improvement varying by model capacity and generalization split.

YOSO-Model(Qwen3-VL-4B) exhibits substantial absolute gains from reflection, particularly on Full Task Success Rate: +3.38% on Domain, +4.77% on Task, and +4.04% on Website splits. This suggests that the base model benefits significantly from iterative self-correction, with the reflection mechanism compensating for initial grounding errors that would otherwise cascade into task failures. The Step Success Rate improvements are more modest (+0.75% to +1.26%), indicating that reflection primarily aids in recovering from critical errors rather than incrementally improving every step.

YOSO-Model(Qwen3.5-4B) shows a different pattern. With higher baseline performance, the absolute gains from reflection are smaller: +1.59% on Domain Task SR, +0.79% on Task Task SR, and zero improvement on Website. Notably, the Website split shows saturated performance (96.86% Step SR, 88.89% Task SR) where reflection yields no additional benefit. This ceiling effect suggests that the enhanced visual encoder in Qwen3.5-VL-4B already captures sufficient grounding cues for reliable single-pass execution in familiar visual contexts, leaving limited headroom for iterative refinement. However, the Domain and Task splits still benefit from reflection, indicating that semantic complexity and task novelty remain challenging even with improved visual representations.

The comparison reveals a trade-off between model capacity and reflection utility: stronger base models reduce the marginal value of self-correction but do not eliminate it entirely in challenging generalization scenarios. This validates the architectural choice of decoupling visual grounding from decision-making, allowing the reflection agent to operate as an orthogonal enhancement that scales with task difficulty rather than model capability.

Table 11: Performance Breakdown: SIFT vs. YOSO-Model (Qwen3-VL-4B)

Visual Condition	ScreenSpot-v2			ScreenSpot-Pro		
	SIFT	YOSO	Δ	SIFT	YOSO	Δ
Original	100.0	99.83	−0.17	100.0	95.12	−4.88
Blurred	99.9	99.83	−0.07	100.0	85.20	−14.80
On Desktop	100.0	99.83	−0.17	100.0	92.68	−7.32
Color-Mapped	43.3	99.74	+56.44	76.4	92.76	+16.36
Color-Inverted	0.3	99.74	+99.44	0.3	95.12	+94.82
Average	68.4	99.79	+31.39	75.3	92.18	+16.84

B.2 Comparison with Traditional Feature Matching Methods

To contextualize the performance of our learned grounding approach, we establish a rigorous baseline using classical computer vision techniques. Specifically, we implement a Scale-Invariant Feature Transform (SIFT) (Lindeberg, 2012) pipeline with nearest-neighbor matching (Lowe's ratio test, threshold 0.8) and RANSAC-based geometric verification (confidence 0.995, maximum reprojection error 3 pixels). This represents the strongest traditional alternative for reference-based localization tasks.

The comparison addresses a fundamental research question: Do modern multimodal models provide substantive advantages over hand-crafted feature descriptors for GUI element grounding, or is the task solvable through established geometric correspondence methods? Table 11 reveals that while SIFT performs adequately under pristine conditions, it exhibits systematic failures in scenarios common to GUI automation but rare in natural image matching.

Performance under pristine conditions. Under standard visual conditions (Original, Blurred, On Desktop), SIFT achieves near-perfect accuracy (99.9–100%) on both benchmarks. Our YOSO-Model (4B) achieves 99.83% across all scenarios on ScreenSpot-v2, demonstrating comparable performance to SIFT (-0.07 to -0.17 percentage points). On ScreenSpot-Pro, YOSO achieves 85.20–95.12%, showing larger gaps (-4.88 to -14.80 points) particularly under blur. While SIFT maintains marginal advantages in these idealized scenarios, this apparent parity obscures critical limitations that emerge under photometric stress.

Catastrophic failure under photometric transformations. The fundamental weakness of hand-crafted features emerges when visual conditions deviate from ideal assumptions. On color-mapped images, where false-color transformations destroy natural intensity relationships, SIFT accuracy drops to 43.3% (ScreenSpot-v2) and 76.4% (ScreenSpot-Pro). More dramatically, on color-inverted images, SIFT accuracy plummets to 0.3% on both benchmarks—effectively total failure. This catastrophic performance stems from SIFT's Difference-of-Gaussians detector, which relies on absolute intensity gradients; when pixel values are negated, gradient polarity reverses, causing the detector to miss all keypoints.

In contrast, our YOSO-Model maintains robust performance across all photometric variants: 99.74% on both color-mapped and color-inverted images for ScreenSpot-v2, and 92.76% and 95.12% respectively for ScreenSpot-Pro. This yields dramatic improvements of +56.44 and +99.44 percentage points on ScreenSpot-v2, and +16.36 and +94.82 points on ScreenSpot-Pro. The average robustness improvement (considering only Color and Inverse variants) reaches 77.94 points for ScreenSpot-v2 and 55.59 points for ScreenSpot-Pro.

Fundamental limitations of feature matching. Three inherent constraints explain SIFT's inadequacy for GUI grounding:

1. Texture-less Regions: Modern UI elements frequently employ flat design principles with uniform backgrounds and minimal gradients, providing insufficient keypoints for reliable matching.

2. Photometric Invariance: SIFT's brightness constancy assumption is fundamentally violated by dramatic color variations common in themed applications, dark mode interfaces, and accessibility settings. Our results demonstrate that color inversion—a common user preference—renders SIFT completely ineffective (0.3% accuracy).

3. Semantic Understanding: Hand-crafted descriptors operate at the pixel level without semantic comprehension. While humans effortlessly recognize inverted or color-mapped UI elements based on shape and layout, SIFT cannot bridge the appearance gap without explicit photometric invariance mechanisms.

Implications for GUI automation. The 55–78 percentage point robustness gap demonstrated in Table 11 validates the necessity of semantic-level

Table 12: Latency Comparison: YOSO-Agent vs Qwen3.5-122B. Task: 17 steps on akc.org. All times in milliseconds (ms).

Resolution	Action	YOSO-Agent (Qwen3-VL-4B)				Qwen3.5-122B-A10B			
		n	Mean	Min	Max	n	Mean	Min	Max
640×360	Click	10	307.1	251.4	335.4	10	486.4	435.8	533.2
	Scroll	7	171.2	156.7	195.5	7	344.6	317.6	372.6
	All	**17**	**251.1**	**156.7**	**335.4**	**17**	**428.0**	**317.6**	**533.2**
1280×1000	Click	10	339.2	301.9	393.8	10	558.0	472.9	676.0
	Scroll	7	214.1	175.3	268.8	7	469.0	436.4	495.9
	All	**17**	**287.7**	**175.3**	**393.8**	**17**	**521.3**	**436.4**	**676.0**
1920×1080	Click	10	367.8	322.3	423.8	10	667.0	603.2	733.4
	Scroll	7	281.5	204.0	367.8	7	607.6	570.0	644.9
	All	**17**	**332.2**	**204.0**	**423.8**	**17**	**642.6**	**570.0**	**733.4**

Total Task Latency (640×360 | 1280×1000 | 1920×1080): YOSO-Agent: 4.3s | 4.9s | 5.6s Qwen3.5-122B: 7.3s | 8.9s | 10.9s

Table 13: End-to-End Error Distribution by Split. State verification failures stem from the Task Manager's binary classification deficiency, distinct from geometric grounding errors.

Split	Error Category	Manifestation	Original	Recovered	Residual	Ratio
Domain	Geometric Misalignment	grounding_point_outside_bbox	41	11	30	56.9%
	State Verification Failure	new_step_misjudge	13	7	6	18.1%
		scroll_not_reached	11	0	11	15.3%
		scroll_passed	7	3	4	9.7%
Task	Geometric Misalignment	grounding_point_outside_bbox	17	5	12	65.4%
	State Verification Failure	new_step_misjudge	4	2	2	15.4%
		scroll_not_reached	3	0	3	11.5%
		scroll_passed	2	1	1	7.7%
Website	Geometric Misalignment	grounding_point_outside_bbox	16	5	11	48.5%
	State Verification Failure	new_step_misjudge	10	3	7	30.3%
		scroll_not_reached	7	0	7	21.2%

understanding for reliable GUI automation. Notably, our YOSO-Model achieves >99.7% accuracy across all variants on ScreenSpot-v2, demonstrating that a 4B parameter model can attain near-perfect robustness when trained with appropriate grounding objectives. Real-world deployment scenarios invariably involve visual variations—dark mode, high-contrast themes, color filters for accessibility—that violate the photometric assumptions underlying traditional feature matching. Our approach's ability to maintain >92% accuracy across all tested variants, while SIFT fails catastrophically (0.3–43.3%), establishes that modern multimodal models provide substantive and essential advantages for robust GUI grounding.

Implications for GUI automation. The 52–78 percentage point robustness gap demonstrated in Table 11 validates the necessity of semantic-level understanding for reliable GUI automation. Real-world deployment scenarios invariably involve visual variations—dark mode, high-contrast themes, color filters for accessibility—that violate the photometric assumptions underlying traditional feature matching. Our learned approach's ability to maintain >87% accuracy across all tested variants, while SIFT fails catastrophically (0.3–43.3%), establishes that modern multimodal models provide substantive and essential advantages for robust GUI grounding.

B.3 End-to-End Latency Evaluation

For practical deployment in interactive automation scenarios, system latency is as critical as accuracy. We conduct comprehensive latency profiling to characterize the real-time performance of our multi-agent architecture and identify optimization opportunities.

Our measurement methodology employs high-resolution timing (microsecond precision) injected at each pipeline stage. We report statistics over a representative complex task on akc.org comprising 17 steps (10 clicks, 7 scrolls). All measurements are performed after warmup to eliminate cold-start effects.

Overall Latency Advantage. As shown in Table 12, YOSO-Agent achieves substantial la-

tency reductions across all resolutions compared to Qwen3.5-122B-A10B. At 640×360 resolution, YOSO-Agent completes actions in 251.1 ms on average—1.71× faster than Qwen3.5-122B's 428.0 ms. This gap widens at higher resolutions: at 1920×1080, YOSO-Agent maintains 332.2 ms mean latency versus 642.6 ms (1.93× speedup).

Resolution Scaling. Both systems exhibit latency growth with resolution, but YOSO-Agent scales more gracefully. Increasing resolution from 640×360 to 1920×1080 incurs only 1.32× overhead for YOSO-Agent (251.1→332.2 ms), compared to 1.50× for Qwen3.5-122B (428.0→642.6 ms). This superior scaling stems from YOSO-Agent's efficient 4B parameter visual encoder versus the 122B model's heavier computational demands.

Action Type Analysis. Clicks consistently dominate latency for both systems due to their two-stage processing (Task Manager verification + Action Agent coordinate prediction) versus single-stage scroll classification. Notably, YOSO-Agent's click latency (307.1–367.8 ms) remains competitive with Qwen3.5-122B's scroll latency (344.6–607.6 ms), underscoring the efficiency of our lightweight grounding architecture.

End-to-End Task Performance. The complete 17-step task completes in 4.3 s, 4.9 s, and 5.6 s for YOSO-Agent across the three resolutions, compared to 7.3 s, 8.9 s, and 10.9 s for Qwen3.5-122B—representing 1.7× to 1.9× end-to-end speedups. Variance remains moderate for both systems (max/mean ≈ 1.3–1.4×), primarily attributable to variable generation lengths for coordinate outputs.

C Error Analysis

Beyond quantitative metrics, we analyze failure modes to guide future architectural improvements. We conduct two complementary analyses: (1) resolution-aware grounding error analysis on ScreenSpot-v2, examining how extreme scales affect localization precision; and (2) end-to-end task error analysis on Mind2Web spanning 4,290 steps across domain, task, and website splits.

Grounding Error Analysis by Resolution Tier. Analysis of 15 failure cases reveals a bimodal error distribution across resolution tiers (Table 14). Nearly half (46.7%) concentrate in low-resolution inputs (short edge <500px), where target elements approach sub-pixel scales and high-frequency texture cues are lost during downsampling. Conversely, one-third (33.3%) occur in high-resolution canvases (2560×1440 and mobile 1080×2400), where the fine-grained encoder's attention mechanism appears to dilute across large pixel spaces, missing diminutive targets with area ratios below 0.02%. This suggests the enhanced architecture excels at moderate resolutions but faces dual challenges at extremes: insufficient detail in low-res, and attention fragmentation in high-res.

End-to-End Error Taxonomy. Table 13 presents the error distribution across three splits. We categorize failures into two primary sources: **geometric misalignment** (coordinate predictions outside target bounding boxes) and **state verification failure** (erroneous binary classification by the Task Manager). The latter encompasses three manifestations: `new_step_misjudge` (element-level state misclassification), `scroll_not_reached` (insufficient scroll depth), and `scroll_passed` (premature termination including `scroll_passed_bottom` and `chosen_block_mismatch`).

Geometric misalignment dominates the Task split (65.4%), while state verification failures concentrate in the Website split (51.5%). Notably, `scroll_not_reached` shows 0% recovery across all splits, indicating fundamental limitations in the Task Manager's viewport state classification rather than grounding inaccuracies.

D Prompt Templates

For reproducibility, we provide the complete prompt templates employed by our progressive three-stage training paradigm. The prompts are designed to enforce strict output formatting while maintaining sufficient contextual information for spatial reasoning and decision-making. All Markdown syntax and special formatting symbols have been stripped to ensure raw text compatibility during model ingestion.

Reflection Agent (Self-Correction). this prompt enforces a rigid JSON output schema for reliable reward calculation. It evaluates predicted actions against visual feedback and historical trajectories.

```
System: You are a rigorous GUI
    interaction localization evaluator.
    Your sole output must be a valid
    JSON object.
```

Table 14: Grounding Error Analysis on Screenspot-v2-Reference dataset

Resolution Tier	Count	Typical Dimensions
Low-Resolution (Short edge <500px)	7 (46.7%)	362×313, 399×454, 480×531
Mid-Resolution (Short edge 500–1000px)	3 (20.0%)	960×540
High-Resolution (Short edge >1000px)	5 (33.3%)	2560×1440, 1080×2400

```
User: The input images are provided in
    fixed order: Reference Image,
    followed by Observation (where the
    predicted click position is marked
    with an asterisk). The predicted
    click coordinates are ({pred_x:.1f},
     {pred_y:.1f}). {history_text}
    Evaluate whether the current click
    position is incorrect. If incorrect,
     determine the approximate direction
     of the click point relative to the
    target bounding box in the Reference
     Image. Strictly output only the
    following JSON format: {"is_wrong":
    true/false, "direction": "left/right
    /up/down/upper_left/upper_right/
    lower_left /lower_right/overlap/
    unknown", "reason": "concise
    explanation"}
```

Action Agent (Matching and Grounding). A streamlined variant that merges task management and grounding functions into a binary matching decision with conditional coordinate regression.

```
System: You are an intelligent GUI
    Grounding Agent. You will receive
    two images in fixed order: Current
    Screenshot and Reference Image. The
    Reference Image contains a red
    bounding box marking the target
    element. If the Current Screenshot
    fully contains the identical element
    , output exactly "Yes, [xxx.x, yyy.y
    ]", where coordinates are normalized
     to a 000.0-999.9 scale and
    correspond to the element center in
    the Current Screenshot. Otherwise,
    output exactly "No".


User: The Current Screenshot and
    Reference Image are provided.
    Determine whether the Current
    Screenshot fully contains the red-
    boxed target from the Reference. If
    yes, output "Yes, [xxx.x, yyy.y]".
    If no, or if only partially visible,
     output "No".

[Dynamic History Injection After
    Reflection]
Previous interaction logs: Round {i}:
    Predicted ({x:.1f}, {y:.1f}), Error:
     {is_wrong}, Direction: {direction},
     Reason: {reason}
```

Action Agent (Coordinate Grounding Only). This prompt corresponds to the supervised fine-tuning phase, establishing foundational spatial localization. It guides the model to map visual references to precise normalized screen coordinates without generating extraneous text.

```
System: You are an intelligent GUI
    Grounding Agent. Your task is to
    analyze a computer screen and
    determine precise pixel coordinates
    for GUI interactions based on a
    visual reference.
Objective: Locate the target UI element
    in the Screenshot that corresponds
    to the red bounding box in the
    Reference Image, considering the
    User Task.
Input Data:
1. Screenshot: The current GUI state.
2. Reference Image: Contains a red
    bounding box marking the target.
3. User Task: Context of the goal.
4. Predicted Action: Interaction type.
Coordinate System Specifications:
- Normalization: Coordinates are
    normalized to a 000.0-999.9 scale.
- Origin: (000.0, 000.0) is the Top-Left
     corner.
- X-axis: 000.0 (Left edge) to 999.9 (
    Right edge).
- Y-axis: 000.0 (Top edge) to 999.9 (
    Bottom edge).
- Example: [500.0, 500.0] represents the
     exact center of the screen.
Grounding Strategy:
1. Analyze: Identify the target in the
    Reference Image and understand the
    User Task.
2. Locate: Search for the corresponding
    element in the Screenshot. If
    multiple identical elements exist,
    use the User Task to select the
    correct one.
3. Decide: If the element is found,
    calculate the coordinates ON THE
    CURRENT SCREENSHOT based on the
    Action Type. The coordinates MUST be
     calculated from the Current
    Screenshot, NOT from the Reference
    Image. The Reference Image only
    shows WHAT to find; the coordinates
    are WHERE it is in the Current
    Screenshot.
Output Formats (Strict Adherence
    Required):
Output ONLY the coordinate string. Do
    NOT provide explanations.
```

```
- Point Clicks: [xxx.x, yyy.y] -> Must
    be the Visual Center of the element.
- Padding Rule: Each number MUST be
    exactly 5 characters (3 digits +
    period + 1 digit).
Constraints:
- Output must be pure text: no markdown,
     no sentences.
- All coordinate values must be between
    000.0 and 999.9.
- Do not approximate; be precise with
    the decimal place.

User:
Current Screenshot: [Image]
Reference Image: [Image]
User Task: [Task Description]
Predicted Action: [Action Type]
Based on the Reference Image and User
    Task, locate the target in the
    Current Screenshot. If the target
    element is clearly visible, output
    the coordinates for the action;
    otherwise, output "none".
```