GUIDE: Resolving Domain Bias in GUI Agents through Real-Time Web Video Retrieval and Plug-and-Play Annotation

Rui Xie[1,2], Zhi Gao[2,3]✉, Chenrui Shi[2,3], Zirui Shang[2,3], Lu Chen[1]✉, Qing Li[2]✉

[1] Shanghai Jiao Tong University, Shanghai, China
[2] State Key Laboratory for General Artificial Intelligence, BIGAI, Beijing, China
[3] Beijing Institute of Technology, Beijing, China

Abstract. Large vision-language models have endowed GUI agents with strong general capabilities for interface understanding and interaction. However, due to insufficient exposure to domain-specific software operation data during training, these agents exhibit significant *domain bias*—they lack familiarity with the specific operation workflows (*planning*) and UI element layouts (*grounding*) of particular applications, limiting their real-world task performance. In this paper, we present **GUIDE** (**G**UI **U**nbiasing via **I**nstructional-Video **D**riven **E**xpertise), a training-free, plug-and-play framework that resolves GUI agent domain bias by autonomously acquiring domain-specific expertise from web tutorial videos through a retrieval-augmented automated annotation pipeline. **Guide** introduces two key innovations. First, a *subtitle-driven Video-RAG pipeline* unlocks video semantics through subtitle analysis, performing progressive three-stage retrieval—domain classification, topic extraction, and relevance matching—to identify task-relevant tutorial videos. Second, a *fully automated VLM pairwise video annotation pipeline* feeds consecutive keyframes enhanced with UI element detection into VLMs, inferring the required *planning* and *grounding* knowledge that are injected into the agent's corresponding modules to address both manifestations of domain bias. Extensive experiments on OSWorld demonstrate **Guide**'s generality as a plug-and-play component for both multi-agent systems and single-model agents. It yields **+4.47–+7.48 percentage-point** improvements and reduces execution steps—without modifying any model parameters or architecture—validating **Guide** as an architecture-agnostic enhancement to bridge GUI agent domain bias.

Keywords: GUI Agent · Domain Bias · Video Retrieval-Augmented Generation · Autonomous Learning · VLM Pairwise Video Annotation

✉ Corresponding authors.

† Code: https://github.com/sharryXR/GUIDE. Dataset: https://huggingface.co/datasets/sharryXR/GUIDE-dataset.

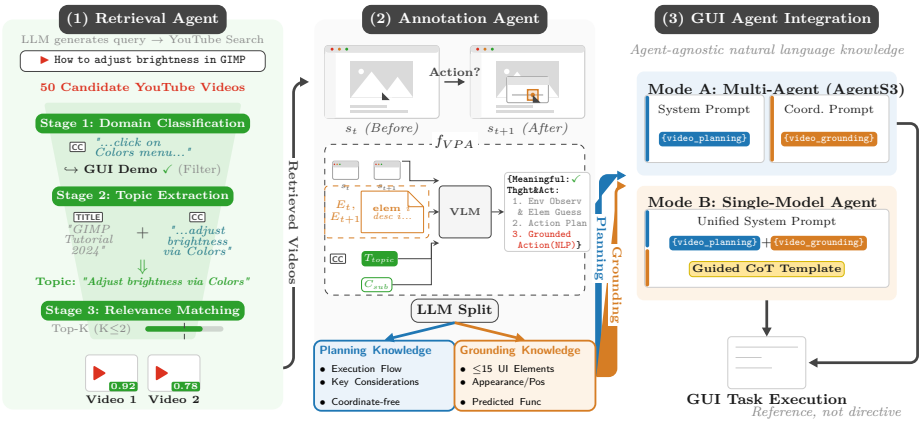

Fig. 1: Overview of GUIDE. (1) A *Retrieval Agent* filters YouTube candidates via three subtitle-driven stages to select top-K videos. **(2)** An *Annotation Agent* applies f_{VPA} (Eq. 1) on keyframe pairs s_t/s_{t+1} with UI element graphs E_t/E_{t+1}, topic T_{topic}, and subtitle context C_{sub}, producing planning and grounding knowledge. **(3)** Knowledge is injected into the *GUI Agent* in a plug-and-play manner—supporting multi-agent (Mode A) and single-model (Mode B) architectures.

1 Introduction

The rapid advancement of large vision-language models (VLMs) [3,4,19,20] has enabled a new generation of GUI agents [1,8,14,24,32] capable of autonomously operating desktop and mobile applications through visual perception and natural language reasoning. These agents can interpret screenshots, understand task instructions, and generate executable actions such as clicks, keystrokes, and scrolls, demonstrating impressive *general* capabilities in interface understanding, complex reasoning, and instruction following. However, when confronted with specific software tasks in real-world scenarios, current GUI agents exhibit significant *domain bias*: they lack familiarity with the **planning-level** and **grounding-level** domain knowledge.

- **Planning-level domain bias.** The agent is unfamiliar with the specific operation workflows of a given application—it cannot accurately determine which concrete steps are needed, their sequential order, or which menus and panels to navigate. For instance, an agent may understand the semantics of "adjust image brightness" but not know that in GIMP the correct path is `Colors` → `Brightness-Contrast` rather than the `Image` → `Adjustments` menu familiar from other editors (*e.g.*, Adobe Photoshop).
- **Grounding-level domain bias.** The agent is unfamiliar with the UI layouts and control styles of a specific application—it struggles to locate task-relevant interactive elements. For example, an agent recognizes the general concept of a "menu" yet cannot locate the `Colors` entry in the GIMP menu bar or identify the brightness slider within its dialog.

This domain bias is fundamentally an *alignment gap* between general model capabilities and domain-specific task requirements, *i.e.*, the model does not lack capability but lacks domain knowledge. Conventional solutions such as manual dataset annotation, expert rule authoring, or domain-specific fine-tuning are costly [27], narrow in coverage, and unable to keep pace with continuously evolving software interfaces.

In this paper we present **GUIDE** (**G**UI **U**nbiasing via **I**nstructional-Video **D**riven **E**xpertise), a novel plug-and-play framework whose core objective is to enable *autonomous learning* for GUI agents—leveraging the vast, continuously growing repository of GUI tutorial videos on the internet to bridge the domain bias, aligning general model capabilities to domain-specific task proficiency.

As illustrated in Fig. 1, **Guide** orchestrates three collaborating agents:
(1) Retrieval Agent (Sec. 3.2). Given a task instruction, the retrieval agent searches YouTube for candidate tutorial videos and applies a progressive three-stage subtitle-driven filtering pipeline—domain classification, topic extraction, and relevance matching—to precisely identify the most task-relevant GUI demonstration videos.
(2) Annotation Agent (Sec. 3.3). The annotation agent processes selected videos through a fully automated VLM pairwise video annotation pipeline: extracting keyframes, detecting UI elements, and inferring transferable annotation information that is then decomposed into structured **Planning** and **Grounding** knowledge.
(3) GUI Agent integration (Sec. 3.4). **Planning** knowledge is injected into the GUI agent's task planning module, and **Grounding** knowledge is injected into the coordinate generation module, precisely targeting the two manifestations of domain bias without modifying the agent's architecture or model weights.

Contributions.

(i) We propose an *autonomous learning* paradigm for GUI agents that leverages internet video resources to bridge *domain bias*, aligning general capabilities to domain-specific tasks without manual annotation or model fine-tuning.
(ii) We design a *subtitle-driven Video-RAG pipeline* that exploits the unique subtitle modality of videos, achieving progressive content-level retrieval precision far beyond title-based keyword matching.
(iii) We construct a fully automated VLM pairwise video annotation pipeline with a *transferable annotation format*, producing the **Planning** and **Grounding** knowledge that targets the two manifestations of GUI agent domain bias.
(iv) We show **+4.47–+7.48 percentage-point** OSWorld [27] gains across three agent architectures, plus WindowsAgentArena (WAA) [5] transfer and a same-backbone Watch & Learn-style raw-trajectory control, validating **Guide** as a scalable plug-and-play solution.

2 Related Work

GUI Agents. Vision-language models have enabled GUI agents to operate across desktop [1, 2, 14, 16], web [13], and mobile [8] environments via direct visual

perception and action generation. However, OSWorld [27]—a benchmark of 369 real-world tasks spanning diverse applications—reveals that even state-of-the-art agents suffer severe performance drops on out-of-distribution software, exposing a persistent *domain bias* that generalist training alone cannot overcome.

Retrieval-Augmented Generation for GUI Agents. RAG [15] augments language models with external knowledge at inference time; ReAct [30] further integrates reasoning with acting. Recent works extend this paradigm to GUI agents: RAG-GUI [28] retrieves web tutorials, KG-RAG [12] constructs knowledge graphs from UI transitions for plan retrieval, and OS-Symphony [29] synthesizes visually grounded tutorials on demand. A complementary line injects video-based knowledge without automated retrieval: Mobile-Agent-V [25] and ShowUI-Aloha [33] feed pre-assigned demonstration videos into agents, but rely on *manually curated, task-specific* assignments that do not scale. None of these retrieves and processes *raw tutorial videos from the open web* at test time—the core capability of **Guide**.

Video Materials for GUI Agents. Screen recordings and tutorial videos are increasingly exploited as data sources for GUI agents. Large-scale datasets such as AITW [22] and GUI-World [7] provide annotated interaction episodes, while AssistGUI [10] uses screen recordings for desktop task planning. TongUI [31] and VideoAgentTrek [17] further mine YouTube tutorials at scale to synthesize GUI trajectories for model fine-tuning—the former via VLM zero-shot annotation, the latter via an inverse dynamics module with LLM-generated inner monologues. Watch and Learn [23], the most closely related work, trains a lightweight inverse dynamics model on consecutive frames, generates LLM reasoning traces, and supports both offline fine-tuning and inference-time retrieval as in-context exemplars. **Guide** differs in three respects: *(i)* **live web retrieval**—**Guide** searches YouTube in real time for each task, whereas Watch and Learn retrieves from a pre-built offline corpus; *(ii)* **subtitle-driven Video-RAG**—progressive semantic filtering (domain classification → topic extraction → relevance matching) via the subtitle modality, achieving content-level precision beyond keyword search; *(iii)* **richer annotation**—a VLM annotator jointly consumes keyframe pairs with UI element graphs, video topic, and subtitle context to produce structured knowledge (strategic reasoning, visual element descriptions, coordinate-free workflow), rather than labeling every frame pair with a lightweight IDM.

3 Method

3.1 Overview

Given a task instruction $\mathcal{T}$ (*e.g.*, "set conditional formatting for column B in LibreOffice Calc") and the associated application name, **Guide** automatically retrieves relevant GUI tutorial videos from the web, extracts transferable domain knowledge, and injects it into a GUI agent to bridge its domain bias for the specific task. Architecturally, **Guide** is a *multi-agent collaborative system*

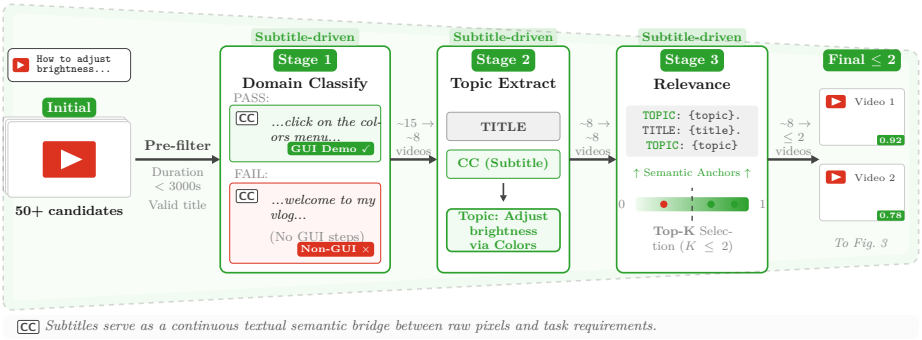

Fig. 2: Subtitle-driven Video-RAG pipeline. From 50+ YouTube candidates, a metadata pre-filter removes outliers, then three subtitle-driven stages progressively narrow results: **(a)** domain classification filters non-GUI content, **(b)** topic extraction (title + subtitle → semantic descriptor), and **(c)** dual-anchored relevance matching (topic as primary anchor, score 0–1). Final top-K ($K \leq 2$) videos proceed to annotation (Fig. 3).

comprising three specialized agents (Fig. 1): a *Retrieval Agent* that identifies task-relevant tutorial videos via subtitle-driven Video-RAG (Sec. 3.2), an *Annotation Agent* that extracts transferable knowledge through pairwise video annotation (Sec. 3.3), and the downstream *GUI Agent* into which the extracted knowledge is injected in a plug-and-play manner (Sec. 3.4).

3.2 Subtitle-Driven Video-RAG Pipeline

Extending RAG [15] from text to *videos* poses unique challenges: video titles are noisy, candidate types are heterogeneous (*e.g.*, lectures vs. hands-on tutorials), and the operational knowledge within frames is semantically opaque. **Guide** addresses these challenges through a key insight: *subtitles* (including auto-generated captions) naturally contain narrated operation steps, UI element names, and task-purpose explanations, serving as a textual semantic bridge between video content and task requirements. As illustrated in Fig. 2, the pipeline progressively narrows 50+ candidates to at most $K=2$ relevant videos through initial pre-filtering followed by three subtitle-driven stages.

Initial Retrieval and Pre-filtering. Given $\mathcal{T}$ and the application name, an LLM generates a search-friendly query (*e.g.*, "How to adjust brightness in GIMP"). To broaden recall, a simplified variant is also produced by stripping filler words (*e.g.*, "how to", "tutorial"), yielding up to two query variants. These are executed via yt-dlp [9] YouTube search to collect 50+ candidate URLs, which are deduplicated and subjected to lightweight metadata pre-filtering (duration < 3000 s, valid title characters).

Stage 1: Subtitle-Assisted Domain Classification. For each surviving candidate, the system downloads its subtitle file and performs multi-step cleaning:

removing VTT/SRT markers, timestamps, HTML tags, and duplicate lines, then re-segmenting at sentence boundaries. The cleaned text (truncated to 10 000 characters) is jointly input with the video title to an LLM, which classifies whether the video contains *actual GUI operation demonstrations* rather than theoretical explanations, reviews, or entertainment.

The critical value: *subtitles provide content evidence that titles alone cannot.* A title such as "Excel Tips" is ambiguous; subtitles containing "click on the Format menu" or "select the cell range" unambiguously indicate a hands-on GUI tutorial.

Stage 2: Subtitle-Driven Topic Extraction. For confirmed GUI tutorials, the system extracts a refined *topic*—a 12–30 word text precisely describing the software, task, and key operations. This extraction is based on joint title + subtitle analysis, with the LLM distilling the actual operational semantics from the narration rather than relying on the potentially misleading title. For example, a video titled "Excel Tutorial 2024" whose subtitles actually demonstrate LibreOffice Calc operations might yield the topic "Creating and formatting a data table with conditional formatting in LibreOffice Calc"—correctly reflecting the true content rather than the misleading title.

Stage 3: Topic-Title Joint Relevance Matching. For each candidate, the system constructs a *dual-anchored prompt* that positions the subtitle-derived topic as the primary semantic anchor:

```
TOPIC (higher priority): {topic}. TITLE: {title}. TOPIC: {topic}
```

The topic is repeated at both ends of the context window as a dual-anchored attention signal, ensuring the LLM prioritizes content-level semantics over the potentially noisy title. All candidates are batch-scored for relevance (0.0–1.0) against the task description.

An *adaptive top-K* strategy retains the top-ranked video unconditionally; subsequent videos require relevance ≥ 0.5, up to $K{=}2$, ensuring at least one reference while preventing low-quality videos from degrading performance.

3.3 Fully Automated Annotation Pipeline

The second core innovation of **Guide** is a fully automated pipeline that converts retrieved tutorial videos into structured, transferable knowledge (Fig. 3). The pipeline follows an *inverse-dynamics-style* pairwise annotation paradigm: given two consecutive interface states (keyframes), a VLM infers the actions and intent that likely occurred between them, without training a separate dynamics model.

Keyframe Extraction: State Sampling. The pipeline first transcribes the video audio using Whisper [21] (base model, word-level timestamps), producing VTT subtitle files with sentence-level segment merging. It then extracts

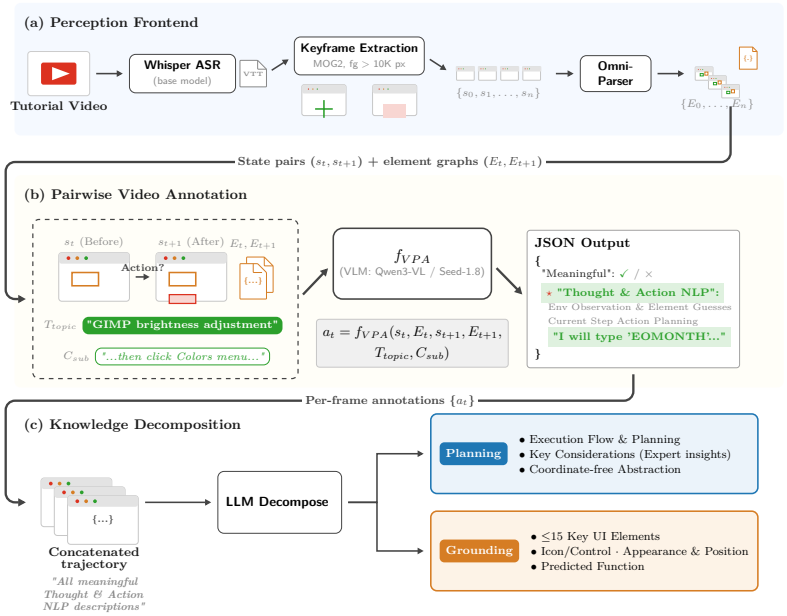

Fig. 3: Fully automated annotation pipeline. Retrieved videos (Fig. 2) are converted into structured knowledge through three phases. **(a)** *Perception Frontend*: Whisper ASR → keyframe extraction (MOG2) → OmniParser [18] UI element graphs E_t. **(b)** *Pairwise Video Annotation*: f_{VPA} (Eq. 1) takes keyframe pairs (s_t, s_{t+1}), element graphs (E_t, E_{t+1}), topic T_{topic}, and subtitle context C_{sub} to produce structured annotations. **(c)** *Knowledge Decomposition*: per-frame annotations are aggregated and decomposed into planning and grounding knowledge.

keyframes using a subtitle-timestamp-aligned background subtraction method: the video is segmented according to subtitle intervals, and within each segment the MOG2 algorithm [34] detects frames with significant visual changes (foreground pixel count $> 10\,000$). Consecutive change sequences are analyzed to extract start and end frames of each transition, yielding a discrete sequence of interface states—the "state pairs" for pairwise video annotation.

UI Element Detection: Enhancing Perception. Each keyframe is processed by OmniParser [18], which detects interactive UI elements (buttons, text fields, menus, *etc.*) and produces bounding-box annotations with structured JSON descriptions (position, type, text label). This *element-level prior knowledge* substantially improves VLM inference by providing structured UI understanding beyond raw visual perception.

VLM Pairwise Video Annotation. We formalize the annotation step as a pairwise video annotation problem. Let s_t and s_{t+1} denote two consecutive keyframe images, E_t and E_{t+1} the corresponding OmniParser element graphs (bounding boxes, types, and text labels), T_{topic} the video topic extracted during

retrieval, and C_{sub} the local subtitle context (preceding, current, and following sentences at the corresponding timestamp). The annotation agent acts as a VLM pairwise annotation function:

$$a_t \;=\; f_{\text{VPA}}\big(s_t,\, E_t,\, s_{t+1},\, E_{t+1},\, T_{\text{topic}},\, C_{\text{sub}}\big), \tag{1}$$

where f_{VPA} denotes the VLM-based pairwise video annotator and a_t is the structured annotation output. Consecutive keyframe pairs are processed in a sliding-window fashion (frames 0 – 1, 1 – 2, 2 – 3, *etc.*). The annotator visually compares the two frames, detects element-level changes via E_t/E_{t+1} comparison, assesses task-meaningfulness using T_{topic} and C_{sub}, and infers actions with strategic intent.

Transferable Annotation Format. Coordinate-level video labels are not transferable across environments. Instead, a_t contains a `Meaningful` flag and a natural-language `ThoughtAction` field:

Semantic Noise Filtering (`Meaningful`). A boolean indicating whether the frame-pair change is task-relevant. This field leverages the joint semantic judgment of VLM + OmniParser to *complement traditional keyframe extraction*, which detects only pixel-level changes (mouse movement, window flicker) but cannot distinguish meaningful operations from visual noise. Human verification confirms that this mechanism correctly filters out >91% of non-GUI frames and idle no-action frames; the complete evaluation is provided in the supplementary material.

Strategic Narrative (`Thought&ActionNLP`). The core annotation field, written from the first-person executor perspective, integrating step-by-step execution reasoning—not just "what was done" but "why," the decision logic, and contextual judgment. It includes *UI element visual descriptions* (color, shape, position, text labels) and function inferences (*i.e.*, guesses about each element's purpose based on appearance and context), as well as *grounded actions* (click, type, scroll, *etc.*) in natural language form. This strategic knowledge is inherently transferable across interface variations.

Knowledge Decomposition. Per-frame annotations are consolidated into per-video trajectories by concatenating the meaningful natural-language action descriptions. An LLM then decomposes each trajectory into two knowledge types, precisely targeting the two manifestations of domain bias:

Planning knowledge—provides domain-specific operational workflow:
- *Execution Flow & Planning*: a coherent narrative of the task workflow—logical step sequences, stage objectives, and progression.
- *Key Considerations*: high-value insights (1–2 sentences each) distilled from the tutorial creator's expertise, helping avoid common pitfalls.
- *Coordinate-free Abstraction*: deliberately excludes low-level UI coordinates, decoupling workflow logic from specific display resolutions or layout versions.

Grounding knowledge—provides domain-specific UI element descriptions:

- Up to 15 key interactive elements, each with: *Icon/Control* (name), *Appearance & Position* (visual attributes, screen-relative location), and *Predicted Function* (inferred functionality and typical usage).
- Use visual appearance and semantic function rather than absolute coordinates, enabling element identification across interface states.

3.4 Plug-and-Play Agent Integration

A key design principle of **Guide** is *agent-agnosticism*: the extracted **Planning** and **Grounding** knowledge are expressed entirely in natural language and are independent of any specific GUI agent architecture, enabling plug-and-play integration with both single-model agents and multi-agent systems without modifying model parameters or architecture.

General Integration Principle. **Planning** knowledge is injected into the agent's *planning context* (task decomposition and action decisions); **Grounding** knowledge is injected into the *grounding context* (instruction-to-coordinate mapping). In all cases the knowledge is framed as *reference material*, not directives: the agent verifies suggestions against the current screenshot and prioritizes its own observations when conflicts arise (*e.g.*, due to software version shifts or UI layout differences between the tutorial and the target environment), providing *robustness against domain shift*. When $K>1$ videos are retrieved, each video's knowledge is individually labeled and concatenated, providing multi-perspective references.

Mode A: Multi-Agent System (e.g., AgentS3 [11]). We integrate with AgentS3, consisting of a *Worker*, a *Grounding Agent*, and a *Code Agent*. The two knowledge streams are routed to architecturally matched modules: **Planning** knowledge is appended to the Worker's system prompt as a "Video Planning Reference" section alongside the screenshot and interaction history, guiding task decomposition; **Grounding** knowledge is supplied to the Grounding Agent as supplementary context for each action query, aiding coordinate prediction. This separation mirrors the system's native division of labor; full prompts are provided in the supplementary material.

Mode B: Single-Model Agent (e.g., Qwen3-VL [4]). Both **Planning** and **Grounding** knowledge are unified into a single system prompt under an "External Knowledge" section. A *structured Thought template*, acting as a *guided Chain-of-Thought* [26], enforces active knowledge referencing at each step: the model first assesses task similarity against the retrieved demo, then locates its current stage within the workflow, derives the next sub-task from **Planning** knowledge, and matches **Grounding** element descriptions against the current screenshot before selecting an action. When only one channel is available, the template degrades gracefully to a partial CoT.

4 Experiments

We evaluate **Guide** on OSWorld, studying its impact across agent architectures, knowledge channel configurations, and application domains.

4.1 Experimental Setup

Benchmark. We evaluate on OSWorld [27], a benchmark of 369 real-world computer tasks executed inside live Ubuntu virtual machines. Following common practice, we exclude 8 Google-Drive-dependent tasks and report the *average score* (%) on the remaining 361 tasks spanning 10 application domains; full benchmark details are provided in the supplementary material.

Evaluation Protocol. We evaluate the effectiveness of knowledge injection by comparing *the same agent under different configurations*: **No Annotation** (baseline), **+ Planning** (planning only), and **+ Planning & Grounding** (full dual-channel, default k=7 grounding elements). Three agents are evaluated: (i) Seed-1.8 [6] (Mode B, single-model); (ii) Qwen3-VL-8B [4] (Mode B, single-model); and (iii) AgentS3 [11], a multi-agent system with GPT-5.2 [20] Worker + Seed-1.8 Grounding (Mode A).

Implementation Details. The annotation pipeline uses GPT-5.1 [20] for both frame-pair VLM pairwise annotation (Eq. 1) and planning-grounding decomposition, with OmniParser [18] for UI element detection. The retrieval pipeline uses GPT-4.1 [19] for query generation and GPT-4.1-mini for classification and scoring. Full configuration details are provided in the supplementary material.

Practical Notes. Two factors affect absolute scores: (i) the Seed-1.8 API endpoint employs anti-distillation measures that intermittently produce malformed outputs; we apply lightweight parsing fixes, though the *relative* improvements from **Guide** are unaffected; (ii) intermittent network issues prevent a subset of Chrome tasks from loading target webpages, slightly depressing scores in this domain across all configurations.

4.2 Main Results

Tab. 1 presents the main results on the full OSWorld benchmark. As a plug-and-play component, **Guide** yields consistent and significant improvements across all three agents without modifying any model parameters.

Dual-Channel Analysis. **Planning** knowledge alone produces the dominant share of improvement: +6.79% on Seed-1.8 and +5.03% on Qwen3-VL-8B, accounting for ∼86–91% of the total gain, confirming that the primary bottleneck is *planning-level domain bias*. Adding **Grounding** yields further complementary gains (+0.69–0.80%), strongest in domains with complex UI layouts (GIMP +7.69pp, Calc +6.39pp on Seed-1.8).

Table 1: GUIDE main results on OSWorld (average score, %). Configurations: No Annotation (baseline), + Plan. (planning only), + Plan. & Gnd. (full dual-channel). †Baseline from [4]. ‡GPT-5.2 Worker + Seed-1.8 Grounding (Mode A); k=5. **Bold**: best per column per agent; green: absolute gain over baseline.

Agent	Configuration	Overall (361)	Chrome (46)	GIMP (26)	Calc (47)	Impress (45)	Writer (23)	OS (24)	ThBrd (15)	VLC (17)	VSCode (23)	Multi (93)
Seed-1.8	No Annotation	37.14	36.87	26.92	29.79	43.09	34.77	45.83	66.67	47.06	60.87	26.88
	+ Planning	43.93 +6.79	**47.74**	34.62	42.55	**46.51**	56.38	**50.00**	**73.33**	**52.32**	**65.22**	**27.89**
	+ Plan. & Gnd.	**44.62** +7.48	**47.74**	**42.31**	**48.94**	45.31	**56.51**	**50.00**	**73.33**	**52.32**	**65.22**	25.74
Qwen3-VL-8B	No Annotation†	33.90	–	–	–	–	–	–	–	–	–	–
	+ Planning	38.93 +5.03	**41.22**	**48.00**	27.66	**47.59**	**52.16**	50.00	73.33	40.56	**52.17**	21.71
	+ Plan. & Gnd.	**39.73** +5.83	36.87	46.15	**34.78**	38.48	**52.16**	**54.17**	**80.00**	**51.77**	43.48	**25.98**
AgentS3‡	No Annotation	50.18	41.18	38.46	51.06	44.62	52.17	**70.83**	73.33	**73.91**	73.91	**40.32**
	+ Plan. & Gnd.	**54.65** +4.47	**49.85**	**53.85**	**65.96**	**46.88**	**65.22**	**70.83**	**80.00**	56.25	**82.61**	37.10

Table 2: Cross-benchmark transfer on WindowsAgentArena [5] (154 tasks, average score, %). GUIDE uses the same planning and grounding knowledge format as in OSWorld, without WAA-specific tuning. Task counts: Office 45, Web 30, System 24, Code 24, Media 21, Utility 10. **Bold**: best per backbone group.

Backbone	Configuration	Office	Web	System	Code	Media	Utility	**Overall**	Δ
Agents3+GPT-5.2	No Annotation	47.80	38.60	69.00	62.10	32.30	41.40	49.00	+0.00
	+ Plan. & Gnd.	**57.78**	**46.67**	**83.33**	**75.00**	**38.99**	**50.00**	**59.21**	+10.21
Qwen3-VL-32B	No Annotation	27.10	35.90	53.80	20.90	30.80	14.40	31.70	+0.00
	+ Plan. & Gnd.	**37.78**	**50.00**	**75.00**	**29.17**	**42.89**	**20.00**	**44.16**	+12.46

Architecture-Agnostic Generality. **Guide** also generalizes to multi-agent systems: AgentS3 [11] (GPT-5.2 Worker + Seed-1.8 Grounding) improves from 50.18% to 54.65% (+4.47pp), with the largest gains in GIMP (+15.39pp), Calc (+14.90pp), and Writer (+13.05pp). On WindowsAgentArena (WAA) [5], the same coordinate-free knowledge format transfers to native Windows widgets without tuning, improving Agents3+GPT-5.2 by +10.21pp and Qwen3-VL-32B-Instruct by +12.46pp (Tab. 2). This cross-benchmark evidence confirms that GUIDE's gain is not an OSWorld artifact.

Per-Domain Analysis and Execution Efficiency. Domains with severe domain bias benefit most (Writer +21.74pp, Calc +19.15pp on Seed-1.8). **Grounding** knowledge reduces exploration on successful tasks (VLC −5.0, Calc −3.7 steps) by enabling direct element identification. Knowledge injection increases per-step latency (9.5→11.6 s, +2.1 s), but grounding compensates through fewer steps (15.7 vs. 16.6 on successful tasks); overall, success count grows from 134 to 161 (+20.1%) at modest wall-clock overhead.

4.3 Ablation Studies and Additional Controls

Cross-Benchmark Transfer. Tab. 2 evaluates GUIDE on a different benchmark and OS stack. OSWorld runs in Linux desktop applications, whereas WAA uses native Windows widgets and a different action interface. Despite this shift,

Table 3: Same-backbone OSWorld controls on Qwen3-VL-8B-thinking (361 tasks, average score, %). The Watch & Learn [23]-style row uses the same retrieved videos as GUIDE, but injects raw text trajectories rather than decomposed planning/-grounding knowledge. Domain scores for the published no-annotation baseline are not available in [4].

Configuration	**Overall**	Chrome	GIMP	Calc	Impress	Writer	OS	ThBrd	VLC	VSCode	Multi	Δ
No Annotation†	33.90	–	–	–	–	–	–	–	–	–	–	+0.00
W&L raw traj. ICL	31.96	45.56	38.46	17.02	29.58	43.47	45.83	66.67	40.56	30.43	18.94	-1.94
Grounding-only (k=7)	34.31	41.22	46.15	23.40	38.17	47.82	45.83	66.67	28.79	39.13	19.44	+0.41
+ Planning	38.93	**41.22**	**48.00**	27.66	**47.59**	**52.16**	50.00	73.33	40.56	**52.17**	21.71	+5.03
+ Plan. & Gnd.	**39.73**	36.87	46.15	**34.78**	38.48	**52.16**	**54.17**	**80.00**	**51.77**	43.48	**25.98**	+5.83

Table 4: Annotation-pipeline ablation on a balanced OSWorld subset (120 tasks, 12 per domain, Qwen3-VL-8B-thinking, average over three random seeds; score %). Subtitle-only removes the visual annotation chain and injects transcript-derived planning knowledge. No-OmniParser keeps the visual pairwise annotation chain but removes the UI element graph.

Domain	n	Score (%)			
		None	Sub.	w/o OP	**Full**
Chrome	12	58.33	58.33	58.33	**58.33**
GIMP	12	33.33	50.00	**58.33**	50.00
Calc	12	8.33	16.67	**25.00**	16.67
Impress	12	33.33	41.91	**58.57**	**58.57**
Writer	12	66.65	58.31	58.31	**66.65**
Multi-apps	12	**41.67**	33.33	**41.67**	**41.67**
OS	12	25.00	**41.67**	**41.67**	**41.67**
Thunderbird	12	**75.00**	58.33	66.67	66.67
VLC	12	16.67	39.65	33.33	**40.89**
VSCode	12	41.67	25.00	33.33	**50.00**
Average	**120**	40.00	42.32	47.52	**49.11**
Δ vs. No Annotation	–	+0.00	+2.32	+7.52	+9.11

GUIDE improves every WAA domain under both a strong multi-agent backbone and an open 32B VLM backbone. The open backbone receives the larger absolute gain (+12.46pp, 39% relative), indicating that video-derived procedural knowledge remains useful precisely where built-in domain expertise is weaker.

Attribution and Raw-Context Controls. Tab. 3 isolates the knowledge-injection design on the same backbone and split. Planning-only accounts for 5.03/5.83 = 86.3% of the full Qwen3-VL-8B gain, while grounding alone gives only +0.41pp; grounding is therefore complementary, adding +0.80pp on top of an already correct workflow plan. The Watch & Learn-style ICL baseline uses the same 299 retrieved-video tasks, the same GPT-5.1 trajectory source before decomposition, K=2 demos, 12 steps per demo, and a 24K-character cap. It scores 31.96, below both no annotation and GUIDE, showing that raw tutorial-derived trajectories do not explain the gain; the useful signal is the structured conversion into **Planning** and **Grounding** knowledge.

Annotation-Pipeline Components. Tab. 4 separates the sources of improvement under repeated seeds. Subtitle-only already improves the 120-task subset by

Table 5: Number of retrieved video demonstrations on a 50-task OSWorld subset with Qwen3-VL-8B-thinking and full GUIDE knowledge (score %). All settings complete 50/50 tasks; only the number of retrieved demonstrations changes.

Domain	#Tasks	K=1	K=2	K=3
Chrome	7	71.43	**85.71**	57.14
GIMP	8	**50.00**	**50.00**	**50.00**
Calc	6	**16.67**	**16.67**	**16.67**
Impress	6	33.33	50.34	**50.84**
Writer	5	20.00	**40.00**	20.00
Multi-apps	6	50.00	33.33	**66.67**
OS	1	**100.00**	**100.00**	**100.00**
Thunderbird	3	33.33	**66.67**	**66.67**
VLC	5	**58.13**	20.00	40.00
VSCode	3	**0.00**	**0.00**	**0.00**
Overall	**50**	41.81	44.04	**44.10**

Table 6: Retrieval matching controls under the same 300-video, 3-annotator protocol as the paper's retrieval audit. All modes choose from the same per-task `yt-dlp` candidate pool; scores use the 1.0/0.5/0.0 direct/partial/failed scale.

Retrieval mode	#Videos	Mean	Acc.$\geq$ 0.5	1.0 / 0.5 / 0.0
Full GUIDE	300	**0.867**	**96.00%**	**77.33** / 18.67 / 4.00
Title-only	300	0.782	88.67%	67.67 / 21.00 / 11.33
Random	300	0.628	80.33%	45.33 / 35.00 / 19.67

+2.32pp, confirming that retrieved ASR transcripts contain useful procedural cues. Restoring visual pairwise annotation without OmniParser raises the gain to +7.52pp, showing that adjacent-frame visual reasoning contributes substantially beyond transcripts. Full GUIDE reaches +9.11pp; the additional +1.59pp from OmniParser is positive but modest under a strong GPT-5.2 annotator, so we treat it as complementary rather than decisive in this setting. The design motivation remains robustness: when the annotator is weaker, the explicit UI graph provides the structural prior needed to produce reliable coordinate-free element descriptions.

Retrieved-Video Count. Tab. 5 supports the default K=2 design. Adding a second relevant video improves the 50-task score by +2.23pp over K=1, while a third video adds only +0.06pp. The per-domain pattern is non-monotonic because additional videos can introduce alternative workflows or UI variants that are not helpful for a particular task; overall, K=2 provides the best accuracy/-context tradeoff.

Retrieval Controls. Tab. 6 evaluates whether the retrieval pipeline adds value beyond the candidate pool itself. The random baseline is already reasonably high because the initial task-derived `yt-dlp` query filters for basic relevance. Title-only ranking adds a substantial improvement over random, but Full GUIDE further improves both mean match and acceptable rate by using subtitle-derived topics. Most importantly, the strictly direct-match share rises from 45.33% (random) and 67.67% (title-only) to 77.33%, isolating the subtitle/topic signal as the source of retrieval quality rather than merely the candidate pool.

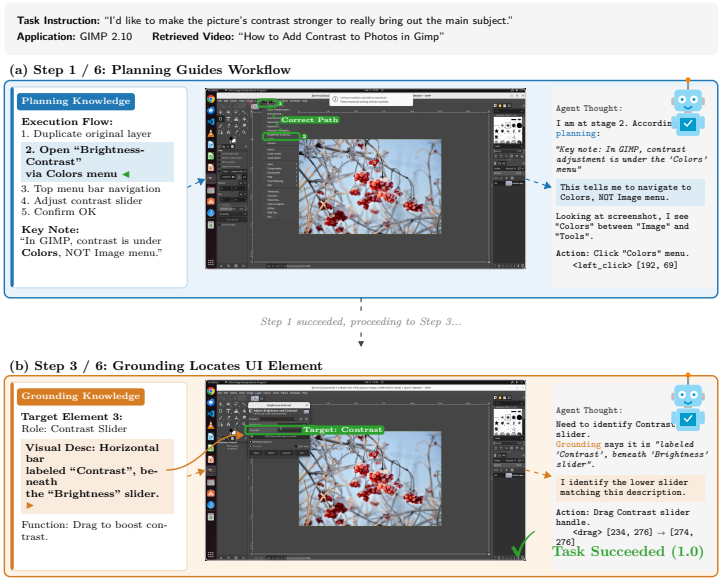

Fig. 4: Qualitative example on a GIMP contrast task. (a) Planning redirects the agent from the conventional "Image" menu to GIMP's correct "Colors" menu. **(b)** Grounding identifies the Contrast slider among visually similar controls.

4.4 Qualitative Analysis

Fig. 4 presents a representative example from the GIMP domain to illustrate how **Guide**'s dual-channel knowledge injection resolves domain bias at both the planning and grounding levels during task execution.

Planning Knowledge Resolves Workflow Ambiguity. In Fig. 4(a), a general-purpose agent would choose the "Image" menu, the conventional location in other editors. GIMP instead places contrast controls under "Colors", creating a typical *planning-level bias*. The retrieved planning knowledge states this distinction, letting the agent select the correct menu path without trial-and-error.

Grounding Knowledge Enables Precise Element Identification. In Fig. 4(b), the dialog presents multiple visually similar sliders. Grounding knowledge provides a discriminative description—"horizontal bar labeled 'Contrast', beneath the 'Brightness' slider"—enabling the agent to identify and manipulate the correct control. Together, planning addresses *what to do* and grounding addresses *where to act*—neither alone suffices, consistent with the ablation results in Sec. 4.3.

Failure Cases. Injected knowledge can also slightly degrade performance when (i) the retrieved video is procedurally mismatched to the target task, misleading the agent with an inapplicable workflow, or (ii) the tutorial's UI layout differs substantially from the evaluation environment, invalidating grounding descriptions. Further examples are provided in the supplementary material.

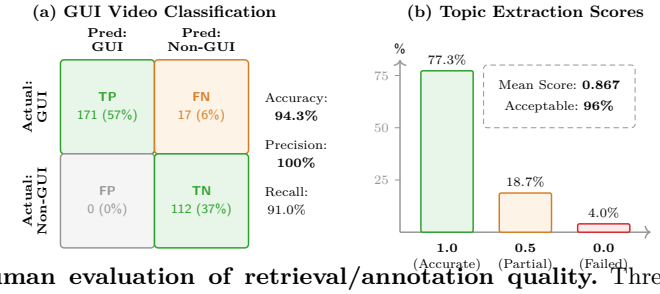

Fig. 5: Human evaluation of retrieval/annotation quality. Three annotators score 300 videos per stage: **(a)** GUI classification keeps 100% precision; **(b)** topic extraction reaches mean 0.867 and 96% acceptable.

4.5 Annotation Pipeline Quality

We audit retrieval coverage and two pipeline stages with human labels.

Retrieval Coverage. Of 361 OSWorld tasks, 299 (82.8%) retrieve at least one relevant tutorial video, and 42.7% of covered tasks retrieve two videos. All covered tasks complete download, keyframe extraction, annotation, and knowledge decomposition; the remaining 62 tasks fall back to baseline behavior without injected knowledge.

GUI Domain Classification (Stage 1). Three annotators judge 300 candidate videos for actual GUI demonstrations (Fig. 5a). The subtitle-assisted classifier reaches **100%** precision and 90.96% recall (F1 = 95.27%), preventing non-GUI videos from entering annotation; false negatives are mostly visual-only tutorials whose narration lacks GUI interaction terms.

Topic Extraction (Stage 2). On 300 confirmed GUI videos, annotators score extracted topics as 1.0/0.5/0.0 for accurate/partial/failed matching. The mean score is 0.867 with 96.00% acceptable topics (Fig. 5b); only 4.00% fail, mainly due to poor auto-generated subtitles, confirming that subtitle-derived topics provide reliable anchors for relevance matching (Sec. 3.2).

5 Conclusion

We presented **Guide**, a training-free framework that mitigates GUI-agent domain bias by acquiring procedural expertise from web tutorial videos. Its subtitle-driven Video-RAG and VLM pairwise video annotation produce transferable **Planning** and **Grounding** knowledge without changing model weights. OSWorld gains of **+4.47–+7.48 percentage points**, WindowsAgentArena transfer [5], and controlled ablations show that structured video knowledge is a scalable resource for GUI agents.

Acknowledgments. We thank the anonymous reviewers, area chairs, and human annotators for their constructive feedback and evaluation support.

References

1. Agashe, S., Jimenez, C.E., Yang, B., Kim, N., Yang, J.: Agent S: An open agentic framework that uses computers like a human. ICLR (2025)
2. Agashe, S., Wong, K., Tu, V., Yang, J., Li, A., Wang, X.E.: Agent S2: A compositional generalist-specialist framework for computer use agents. COLM (2025)
3. Anthropic: Claude Opus 4.6. https://www.anthropic.com/news/claude-opus-4-6 (2025), accessed 24 Jun 2026
4. Bai, S., Cai, Y., Chen, R., Chen, K., Cheng, Z., Ge, W., Hui, B., Li, K., Lin, J., Lu, D., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
5. Bonatti, R., Zhao, D., Bonacci, F., Dupont, D., Abdali, S., Li, Y., Lu, Y., Wagle, J., Koishida, K., Bucker, A., Jang, L., Hui, Z.: Windows Agent Arena: Evaluating multi-modal OS agents at scale. arXiv preprint arXiv:2409.08264 (2024), available at https://arxiv.org/abs/2409.08264. Accessed 24 Jun 2026
6. Bytedance Seed Team: Seed-1.8 model card. Tech. rep., Bytedance (2025), available at https://lf3-static.bytednsdoc.com/obj/eden-cn/lapzild-tss/ljhwZthlaukjlkulzlp/research/Seed-1.8-Modelcard.pdf. Accessed 24 Jun 2026
7. Chen, D., Huang, Y., Wu, S., Tang, J., Chen, L., Bai, Y., He, Z., Wang, C., Zhou, H., Li, Y., Zhou, T., Yu, Y., Gao, C., Zhang, Q., Gui, Y., Li, Z., Wan, Y., Zhou, P., Gao, J., Sun, L.: GUI-World: A video benchmark and dataset for multimodal GUI-oriented understanding. ICLR (2025)
8. Cheng, K., Sun, Q., Chu, Y., Xu, F., Li, Y., Zhang, J., Wu, Z.: SeeClick: Harnessing GUI grounding for advanced visual GUI agents. ACL (2024)
9. yt-dlp contributors: yt-dlp: A feature-rich command-line audio/video downloader. https://github.com/yt-dlp/yt-dlp (2024), accessed 24 Jun 2026
10. Gao, D., Ji, L., Bai, Z., Ouyang, M., Li, P., Mao, D., Wu, Q., Zhang, W., Wang, P., Guo, X., Wang, H., Zhou, L., Shou, M.Z.: ASSISTGUI: Task-oriented desktop graphical user interface automation. CVPR (2024)
11. Gonzalez-Pumariega, G., Tu, V., Lee, C.L., Yang, J., Li, A., Wang, X.E.: Scaling agents for computer use. arXiv preprint arXiv:2510.02250 (2025)
12. Guan, Z., Li, J.C.L., Hou, Z., Zhang, P., Xu, D., Zhao, Y., Wu, M., Chen, J., Nguyen, T.T., Xian, P., Ma, W., Qin, S., Chesi, G., Wong, N.: KG-RAG: Enhancing GUI agent decision-making via knowledge graph-driven retrieval-augmented generation. EMNLP (2025)
13. Gur, I., Furuta, H., Huang, A., Safdari, M., Matsuo, Y., Eck, D., Faust, A.: A real-world WebAgent with planning, long context understanding, and program synthesis. ICLR (2024)
14. Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Zhang, Y., Li, J., et al.: CogAgent: A visual language model for GUI agents. CVPR (2024)
15. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. NeurIPS (2020)
16. Lin, K.Q., Xu, L., Guo, D., Zhong, Y., Liu, J., et al.: ShowUI: One vision-language-action model for GUI visual agent. arXiv preprint arXiv:2411.17465 (2024)
17. Lu, D., Xu, Y., Wang, J., Wu, H., Wang, X., Wang, Z., Yang, J., Su, H., Chen, J., Chen, J., Mao, Y., Zhou, J., Lin, J., Hui, B., Yu, T.: VideoAgentTrek: Computer use pretraining from unlabeled videos. arXiv preprint arXiv:2510.19488 (2025)
18. Lu, Y., Yang, J., Shen, Y., Awadallah, A.: OmniParser for pure vision based GUI agent. arXiv preprint arXiv:2408.00203 (2024)

19. OpenAI: GPT-4o system card. arXiv preprint arXiv:2410.21276 (2024)
20. OpenAI: GPT-5 system card. arXiv preprint arXiv:2601.03267 (2025)
21. Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Robust speech recognition via large-scale weak supervision. ICML (2023)
22. Rawles, C., Li, A., Rodriguez, D., Riva, O., Lillicrap, T.: Android in the wild: A large-scale dataset for android device control. NeurIPS (2023)
23. Song, C.H., Song, Y., Goyal, P., Su, Y., Riva, O., Palangi, H., Pfister, T.: Watch and learn: Learning to use computers from online videos. arXiv preprint arXiv:2510.04673 (2025)
24. Tan, W., Zhang, W., Xu, X., Xia, H., Ding, Z., Li, B., Zhou, B., Yue, J., Jiang, J., Li, Y., et al.: Cradle: Empowering foundation agents towards general computer control. NeurIPS (2024)
25. Wang, J., Xu, H., Zhang, X., Yan, M., Zhang, J., Huang, F., Sang, J.: Mobile-Agent-V: A video-guided approach for effortless and efficient operational knowledge injection in mobile automation. arXiv preprint arXiv:2502.17110 (2025)
26. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. NeurIPS (2022)
27. Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T.J., Cheng, Z., Gao, D., Lu, S., et al.: OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. NeurIPS (2024)
28. Xu, R., Ma, K., Yu, W., Zhang, H., Ho, J.C., Yang, C., Yu, D.: Retrieval-augmented GUI agents with generative guidelines. EMNLP (2025)
29. Yang, B., Jin, K., Wu, Z., Liu, Z., Sun, Q., Li, Z., Xie, J., Liu, Z., Xu, F., Cheng, K., Li, Q., Wang, Y., Qiao, Y., Wang, Z., Ding, Z.: OS-Symphony: A holistic framework for robust and generalist computer-using agent. arXiv preprint arXiv:2601.07779 (2026)
30. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.: ReAct: Synergizing reasoning and acting in language models. ICLR (2023)
31. Zhang, B., Shang, Z., Gao, Z., Zhang, W., Xie, R., Ma, X., Yuan, T., Wu, X., Zhu, S.C., Li, Q.: TongUI: Internet-scale trajectories from multimodal web tutorials for generalized GUI agents. AAAI (2026)
32. Zhang, C., Li, L., He, S., Zhang, X., Qiao, B., Qin, S., Ma, M., Kang, Y., Lin, Q., Rajmohan, S., Zhang, D., Zhang, Q.: UFO: A UI-focused agent for Windows OS interaction. arXiv preprint arXiv:2402.07939 (2024)
33. Zhang, Y., Guo, X., Goh, Y., Hu, J., Chen, Z., Wang, X., Gao, D., Shou, M.Z.: ShowUI-Aloha: Human-taught GUI agent. arXiv preprint arXiv:2601.07181 (2026)
34. Zivkovic, Z.: Improved adaptive Gaussian mixture model for background subtraction. ICPR (2004)