ADAPGUIDE: TURNING TUTORIAL KNOWLEDGE INTO EVOLVING GUIDANCE FOR GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

GUI agents rely on external tutorials to acquire procedural knowledge for unfamiliar software tasks. Existing methods retrieve relevant tutorials and convert them into action guidance. Yet tutorial guidance remains tied to its source interface and example, so it may not fit the current task and can become invalid when actions produce unexpected effects. Fixed guidance then propagates local errors through subsequent stages of task execution and may mistake an incorrect state for progress. To bridge this gap, we introduce **AdapGuide**, a training-free framework that reconstructs reusable operation–effect relations and revises task-specific guidance throughout execution. Each relation links a configured operation to its applicable conditions and expected interface change, enabling **AdapGuide** to bind tutorial knowledge to the objects and settings required by the current instruction and interface. During execution, mismatches between observed and expected effects trigger revisions to object bindings, operation settings, and subsequent stages. Experimental results across three benchmarks demonstrate the effectiveness and generalizability of **AdapGuide**, with average task score gains of **8.08–15.86 pp** across five backbones on OSWorld-Verified, **14.73 pp** on Windows Agent Arena, and **12.47 pp** on DeskCraft.

1 INTRODUCTION

GUI agents increasingly interact with software through visual observations and natural-language instructions. Their ability to operate unfamiliar applications depends on procedural knowledge that may not be available from the current interface alone. External tutorials provide procedural knowledge for unfamiliar tasks at inference time without updating agent parameters. Recent methods have advanced both the acquisition and use of this knowledge. GUIDE extracts planning and grounding information from instructional videos (Xie et al., 2026), while RAG-GUI generates tutorial guidance conditioned on the task, current state, and action history (Xu et al., 2025a). Together, these methods turn human demonstrations into guidance that GUI agents can use during execution.

Despite these advances, transferring a tutorial to a new task presents two connected challenges. First, a tutorial binds its workflow to the interface conditions, objects, values, and settings of one demonstration. Directly reusing these choices can make an operation incompatible with the current interface or produce the right operation with the wrong task content. Consider a request for a chart comparing Q1 sales in East and West. A retrieved tutorial instead uses Product categories and Sales values; preserving these bindings produces a well-formed chart with the wrong content. Figure 1 illustrates this tutorial-to-task mismatch.

Second, even well-adapted initial guidance can become invalid during interaction. An action may fail or produce an unexpected interface change, leaving subsequent instructions grounded in a state the agent never reached. Fixed guidance then propagates local errors through later actions and may mistake an incorrect state for progress.

These challenges call for procedural knowledge that supports both task-specific adaptation and execution-time revision. Our key insight is to transfer **operation–effect relations**: each relation connects a configured operation with its applicable conditions and expected interface change. In Figure 1, selected labels become chart categories and selected values become a data series. Binding these roles to East/West and their Q1 values (200/160) adapts the tutorial workflow to the requested chart. The expected chart change then provides a concrete reference for evaluating execution.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

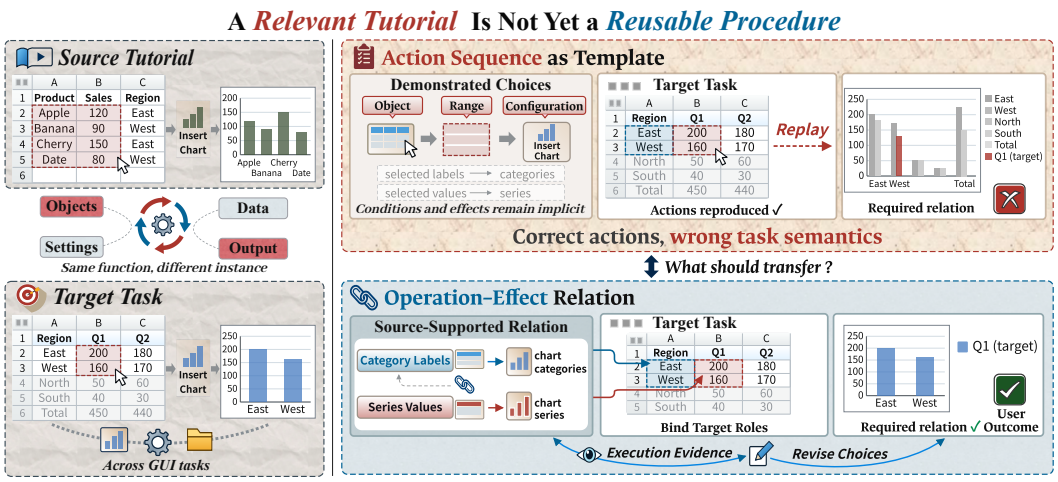

Figure 1: Motivation for transferring operation–effect relations. Replaying tutorial actions yields the wrong chart. Binding categories and values to the target task produces the requested result.

To realize this insight, we introduce **AdapGuide**, a training-free framework that reconstructs source-supported operation–effect relations from retrieved tutorials and turns them into evolving task guidance. AdapGuide first binds relation objects and settings to the current instruction and interface, producing an executable procedure for the requested outcome. During execution, it compares observed interface changes with expected effects. Effect mismatches trigger revisions to object bindings, operation configurations, subsequent stages, and completion judgments. The tutorial-supported relations retain the reusable procedural knowledge, while their task-specific application evolves with interaction. This shared representation connects tutorial reconstruction, task planning, and execution feedback through the conditions and expected effects of the same operations.

Our main contributions are as follows:

(i) We formulate tutorial-to-task transfer through operation–effect relations, which associate configured operations with their applicable conditions and expected interface changes. This representation separates reusable procedural knowledge from the states, objects, and settings of a tutorial example.

(ii) We introduce **AdapGuide**, a training-free framework that reconstructs source-supported relations, binds them to the current task, and continuously revises object bindings, operation configurations, stages, and completion judgments from observed effects.

(iii) We evaluate AdapGuide across five GUI agent backbones on OSWorld-Verified, where it improves average task scores by 8.08–15.86 pp, and extend the evaluation to Windows Agent Arena and DeskCraft. Controlled ablations attribute 4.71 pp to explicit operation–effect relations and 11.36 pp to online guidance revision.

2 Related Work

GUI Agents for General Computer Use. Advances in vision-language models (VLMs) have enabled GUI agents to perceive interfaces and operate computers through visual observations. SeeClick and CogAgent established GUI grounding and fine-grained interface understanding for screenshot-based interaction (Cheng et al., 2024; Hong et al., 2024). UI-TARS, Aguvis, and Qwen-UI-Agent integrate perception, reasoning, planning, and action generation into unified agents spanning desktop, mobile, and web environments (Qin et al., 2025; Xu et al., 2025b; Zhou et al., 2026). Computer use is also becoming integral to frontier general-purpose multimodal models, with GPT and Claude demonstrating strong capabilities on realistic tasks (OpenAI, 2026a; Anthropic, 2026). Benchmarks such as OSWorld, Windows Agent Arena, and AndroidWorld highlight the remaining gap between these increasingly general capabilities and reliable long-horizon execution across diverse applications (Xie et al., 2024; Bonatti et al., 2025; Rawles et al., 2025).

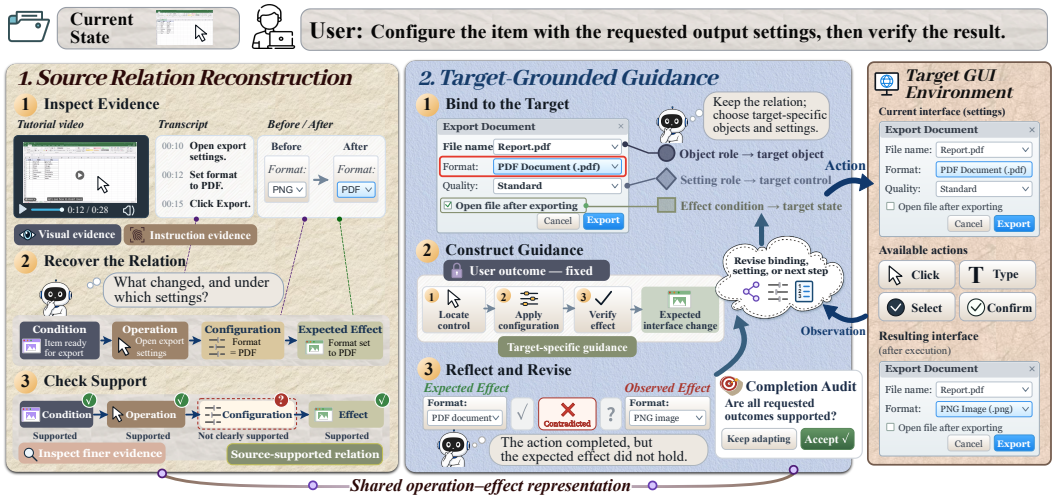

Figure 2: Overview of the **AdapGuide** framework: reconstructing operation–effect relations from tutorials, binding them to the target task, and revising guidance through execution and verification.

Knowledge-Augmented GUI Agents. General computer-use capabilities do not ensure familiarity with application-specific workflows, motivating augmentation with external procedural knowledge. AutoGuide selects applicable guidelines distilled from offline experience (Fu et al., 2024), while Agent S combines external knowledge search and experience retrieval for hierarchical planning (Agashe et al., 2025a). GUIDE converts retrieved instructional videos into planning and grounding knowledge (Xie et al., 2026). Using such guidance during interaction also requires matching it to the evolving task: RAG-GUI assesses tutorial relevance and generates guidance from the current state and action history (Xu et al., 2025a), while WebLearner uses hierarchical memory to organize tutorials, observations, and task progress for multimodal guidance on unseen websites (Long et al., 2026). These efforts move beyond supplying external information toward context-aware construction, selection, and use of procedural guidance.

Adaptive Planning and Execution Feedback. Long-horizon GUI interaction requires decisions to adapt to evolving interface states and execution outcomes. Agent S2 revises hierarchical plans as observations change (Agashe et al., 2025b), while GTA1 uses a judge to select among sampled action proposals (Yang et al., 2026). VeriSafe Agent checks actions against formalized user intent before execution, providing corrective feedback or blocking violations (Lee et al., 2025). These approaches address execution reliability through planning, action selection, and verification. An unexpected transition can also change which objects are available, which settings remain active, and whether the conditions for subsequent operations still hold. **AdapGuide** uses execution evidence to revisit how relations reconstructed from tutorials are applied to the target task. When observed transitions no longer support the current instantiation, it revises object bindings, operation configurations, subsequent stages, and completion judgments while retaining the underlying operation–effect relations supported by source evidence.

3 METHOD

Overview. Figure 2 shows relation reconstruction from tutorials (left), guidance adapted to the target task (center), and GUI interaction (right). Given an initial context $\mathcal{C}_0$ comprising the user instruction, initial GUI observation, and task constraints, **AdapGuide** reconstructs operation–effect relations from tutorial evidence (Section 3.1) and binds them to target objects and settings (Section 3.2). The resulting procedure pairs operations with the relationships they are expected to establish. Observed transitions are assessed against these expectations to update progress and revise guidance; completion is audited against the original user requirements (Section 3.3). All processing occurs at inference time without updating model parameters.

3.1 KNOWLEDGE RECONSTRUCTION

AdapGuide retrieves web and video tutorials using $\mathcal{C}_0$ to reconstruct candidate operation-effect relations $\widetilde{\mathcal{K}}_0$ from passages, transcripts, and interface frames. Each relation specifies how an operation affects object relationships under particular conditions and configurations, retaining object roles and supporting evidence. Procedures validated across tasks also supply prior knowledge. Relations are checked against source evidence: in Figure 2 (left), selecting PDF changes the format setting from PNG to PDF, while uncertain configuration evidence prompts finer inspection. Missing dependencies guide targeted retrieval, and conflicting effects are examined under their demonstrated conditions and configurations to refine applicability. Unresolved assumptions remain explicit. We retain

$$\mathcal{K}_0 = \left\{ k \in \widetilde{\mathcal{K}}_0 \;\middle|\; \mathrm{Compat}(k; \mathcal{C}_0) = 1 \right\}, \tag{1}$$

where Compat denotes the backbone's qualitative applicability judgment based on application identity, visible interface cues, task constraints, and operational assumptions. Under explicit criteria, relations requiring disallowed environment changes or conflicting with task requirements are excluded. Retained relations are subsequently bound to target objects and settings, keeping source values distinct from target parameters; execution prerequisites need not already hold. Details appear in Appendix A.1.

3.2 RELATION APPLICATION AND REVISION

AdapGuide uses $\mathcal{K}_0$ to derive task-specific operations and configurations with expected relational effects. At step t, the agent maintains

$$\mathcal{L}_t = (\mathcal{U}, \mathcal{P}_t, \mathbf{z}_t), \tag{2}$$

with fixed user requirements $\mathcal{U}$, revisable guidance $\mathcal{P}_t$, and progress assessments $\mathbf{z}_t$. Planning binds $\mathcal{K}_0$ to target objects and controls, producing stages described in natural language that specify operations, configurations, conditions, and expected effects. The agent acts from the latest guidance without revisiting tutorials. In Figure 2, observing PNG instead of the requested PDF triggers guidance revision to achieve the target setting.

Before/after observations, execution results, and recent interaction records form transition evidence $\mathcal{E}_t$. After each action in OSWorld and WAA, or action batch in DeskCraft, the backbone compares this evidence with expected effects and continuation conditions to produce Δ_t:

$$\mathcal{L}_{t+1} = \mathcal{L}_t \oplus \Delta_t, \qquad \Delta_t|_{\mathcal{U}} = 0. \tag{3}$$

The update changes bindings, configurations, stage membership or order, and progress while preserving user requirements and a valid procedure. When an action fails, the framework can revise how the relation is applied to the target task without discarding the relation itself. With guidance fixed, verification can identify unmet conditions, redirect actions, and reject completion requests, while status tracking records progress. Guidance revision additionally rewrites the prescribed bindings, configurations, and stages; each version replaces the previous guidance. In Figure 4, disabling row labels corrects the configuration; applying the revised sort produces the required column order.

Let $\mathcal{D}_t$ be completed stage identifiers and $\mathcal{I}_{t+1}$ all updated stage identifiers. With revoked completions $\mathcal{R}_t \subseteq \mathcal{D}_t \cap \mathcal{I}_{t+1}$ and newly completed stages $\mathcal{A}_t \subseteq \mathcal{I}_{t+1}$,

$$\mathcal{D}_{t+1} = \left[(\mathcal{D}_t \cap \mathcal{I}_{t+1}) \setminus \mathcal{R}_t \right] \cup \mathcal{A}_t. \tag{4}$$

Evidence of the expected effect establishes completion; contradictory evidence or changed stage requirements revoke it. Earlier completion is retained when its effect merely leaves view. Unresolved effects remain pending, while intermediate progress supports continuation without marking the stage complete. The agent selects its next action using the updated guidance and progress assessments.

3.3 OPERATION–EFFECT VERIFICATION

Before execution, active rules check required conditions, such as which columns participate in sorting, and suspend actions with corrective feedback when conditions are contradicted or required

evidence is missing. After execution, verification combines progress assessments, errors, and interface changes to guide the next action and revision. Repeated equivalent attempts without progress are constrained; new progress clears the failure state.

At termination, the audit checks each original requirement $u_i \in \mathcal{U}$ against available interaction evidence, independently of whether the stages have been marked complete. Its assessment $\widehat{c}_t(u_i)$ is `supported`, `contradicted`, or `unresolved`. Outstanding requirements and the decision on a normal completion request are

$$\begin{aligned} \mathcal{Q}_t &= \{u_i \in \mathcal{U} \mid \widehat{c}_t(u_i) \neq \texttt{supported}\}, \\ d_t &= \texttt{accept} \iff \mathcal{Q}_t = \varnothing. \end{aligned} \tag{5}$$

Outstanding requirements return to execution and trigger guidance revisions when the current procedure does not adequately address them, including save, apply, or export stages. The audit decides completion; revision updates how the remaining requirements will be fulfilled.

4 Experiments

4.1 Experimental Setup

Benchmarks and metrics. Following Xie et al. (2026), our main evaluation comprises 361 local-environment tasks from OSWorld-Verified (Xie et al., 2024). The benchmark's eight Google Drive-dependent tasks require external services. We additionally evaluate GPT-5.5 on 154 Windows Agent Arena (WAA) tasks (Bonatti et al., 2025) and all 147 tasks in the standard L2 subset of DeskCraft (Wang et al., 2026). We select L2 to evaluate procedural adaptation in professional applications at a task complexity comparable to OSWorld-Verified. The main results and ablations report mean evaluator scores (%), including partial credit; Overall averages over tasks, and differences are in percentage points (pp). The revision analysis reports positive-score rates and the distribution of improved tasks.

Implementation details. We evaluate GPT-5.5, GPT-5.6-sol, Claude Opus 5, Qwen3-VL-32B-Instruct, and Gemini 3.6 Flash. Within each **AdapGuide** configuration, relation reconstruction, target-task planning, action generation, procedure revision, and execution verification use the same backbone. GPT-5.5 acquires tutorial materials at task time on all three benchmarks using the pipeline described in Section 3.1. The OSWorld-Verified evaluations for the other backbones share these retrieved materials, fixing the tutorial evidence while each backbone independently reconstructs operation–effect relations and derives task-specific procedures. For tutorial acquisition, Exa MCP retrieves web pages from the task and application context. yt–dlp metadata and subtitles locate video segments; the YouTube IFrame Player API seeks to these timestamps for frame inspection. We combine passages, transcripts, and targeted frames to recover selections, settings, and effects.

The maximum interaction budget is 50 steps on OSWorld-Verified and WAA, and 100 on DeskCraft. For Base and AdapGuide on OSWorld, Claude Opus 5 uses adaptive context management, while the other backbones retain the five most recent complete interaction turns. Each backbone is evaluated in its Base and AdapGuide configurations. For GPT-5.5, we additionally evaluate direct tutorial injection (Direct) and GUIDE (Xie et al., 2026). GUIDE supplies its officially released planning and grounding annotations to the GPT-5.5 backbone under the same agent setting. Its adapter settings are reported in the supplementary appendix.

Table 1 maps the guidance source, guidance form, and online mechanism for the GPT-5.5 comparisons.

4.2 Main Results

Overall gains across backbones. Table 2 shows average task-score gains of **8.08–15.86 pp** across all five backbones. With GPT-5.5, **AdapGuide** achieves 74.37%, exceeding Base (65.86%), direct tutorial injection (64.82%), and GUIDE (68.43%) by 8.51, 9.55, and 5.94 pp, respectively. Compared with Base, Direct decreases the score by 1.04 pp, whereas GUIDE increases it by 2.57 pp. This contrast suggests that organizing external knowledge into procedural guidance can improve its utility for GUI execution, with AdapGuide providing further gains.

Table 1: Configuration map for GPT-5.5 comparisons on OSWorld-Verified. Verify + revise updates target bindings, configurations, and stages from execution evidence.

Configuration	Guidance source	Guidance form	Online mechanism
Base	Backbone knowledge	Backbone-native plan	Native action loop
Direct	Retrieved tutorials	Tutorial context	Native action loop
GUIDE	Released annotations	Planning + grounding	Native action loop
w/o EK	Backbone knowledge	Task-specific procedure	Verify + revise
w/o OER	Retrieved tutorials	Procedural steps	Verify + revise
w/o GR	Retrieved tutorials	Operation–effect guidance	Verify + track
AdapGuide	Retrieved tutorials	Operation–effect guidance	**Verify + revise**

Table 2: OSWorld-Verified scores (%) across five backbones. Parentheses give task counts; signed Overall differences are relative to each backbone's baseline (pp). Bold marks the best score within each backbone and column, including ties; shaded rows identify AdapGuide. Direct denotes direct tutorial injection; GUIDE denotes the transfer of GUIDE's released planning and grounding annotations (Xie et al., 2026) to the GPT-5.5 agent.

Backbone	Config.	Overall (361)	Chrome (46)	GIMP (26)	Calc (47)	Impress (47)	Writer (23)	OS (24)	ThBrd (15)	VLC (17)	VSCode (23)	Multi (93)
GPT-5.5	Baseline	65.86	67.26	61.54	82.98	72.13	69.55	83.33	80.00	69.93	78.26	43.04
	+ Direct	64.82 *–1.04*	71.60	65.38	82.98	57.24	78.25	87.50	73.33	**81.18**	65.22	42.31
	+ GUIDE	68.43 *+2.57*	67.26	**69.23**	65.96	72.13	**82.20**	87.50	**86.67**	75.92	73.91	54.15
	+ AdapGuide	**74.37** *+8.51*	**78.13**	**69.23**	**85.11**	**78.52**	78.12	**91.67**	**86.67**	75.85	**86.96**	**55.66**
GPT-5.6 Sol	Baseline	66.63	58.56	61.54	78.72	84.90	**78.25**	75.00	80.00	64.71	78.26	46.98
	+ AdapGuide	**74.71** *+8.08*	**75.95**	**76.92**	**85.11**	**86.32**	78.12	**87.50**	**86.67**	**81.18**	**82.61**	**53.15**
Claude Opus 5	Baseline	68.23	65.08	73.08	59.57	74.26	69.56	83.33	**86.67**	69.53	78.26	59.85
	+ AdapGuide	**77.78** *+9.55*	**80.30**	**84.62**	**82.98**	**86.18**	**73.90**	**87.50**	**86.67**	**82.35**	**82.61**	**62.75**
Qwen3-VL 32B-Instruct	Baseline	33.15	45.52	38.46	17.02	31.78	39.12	41.67	60.00	52.94	**60.87**	15.90
	+ AdapGuide	**43.17** *+10.02*	**56.39**	**46.15**	**42.55**	**33.84**	**43.47**	**70.83**	**86.67**	**69.97**	47.83	**20.55**
Gemini 3.6 Flash	Baseline	53.27	56.39	57.69	27.66	40.22	**73.90**	83.33	80.00	**87.28**	**86.96**	38.30
	+ AdapGuide	**69.13** *+15.86*	**71.60**	**84.62**	**78.72**	**59.37**	73.77	**87.50**	**86.67**	75.49	69.57	**53.67**

Backbone sources: GPT-5.5 (OpenAI, 2026a); GPT-5.6 Sol (OpenAI, 2026b); Claude Opus 5 (Anthropic, 2026); Qwen3-VL-32B-Instruct (Bai et al., 2025); Gemini 3.6 Flash (Google, 2026).

Gains across application domains. Table 2 reports application-level scores, with ThBrd denoting Thunderbird and Multi denoting multi-application tasks. The gains extend across most application domains. Chrome, GIMP, Calc, Impress, OS, and Multi improve for all five backbones; GPT-5.5 gains 12.62 pp on Multi (43.04% to 55.66%). The remaining domain-level variation identifies applications where further adaptation is useful: Qwen3-VL-32B-Instruct declines on VS Code, Gemini 3.6 Flash on Writer, VLC, and VS Code, and GPT-5.6-sol shows a small Writer decrease.

Gains across benchmarks. Table 3 extends the GPT-5.5 evaluation to WAA and DeskCraft. **AdapGuide** improves scores from 51.86% to 66.59% on WAA and from 47.17% to 59.64% on DeskCraft, yielding gains of 14.73 and 12.47 pp. These results extend the observed benefits to Windows tasks and the professional application tasks covered by DeskCraft L2.

4.3 Acquiring and Applying Operation–Effect Relations

Tutorial evidence acquisition. **AdapGuide** acquires evidence for operation–effect reconstruction with 6.22 model calls and 73.99K tokens per task, **69.1% fewer calls** and **56.0–65.4% fewer tokens** than GUIDE's estimated annotation demand (Xie et al., 2026) (Table 4). Costs are normalized over 361 tasks, with each method using its own tutorial collection. The reported call and token totals

Table 3: GPT-5.5 scores (%) across three benchmarks. Δ (pp) is AdapGuide's gain over the corresponding baseline in percentage points.

Benchmark	Tasks	Max steps	Baseline	**AdapGuide**	Δ
OSWorld-Verified	361	50	65.86	**74.37**	***+8.51***
WAA	154	50	51.86	**66.59**	***+14.73***
DeskCraft standard L2	147	100	47.17	**59.64**	***+12.47***

Table 4: Tutorial acquisition and processing costs for 361 OSWorld-Verified tasks with GPT-5.5. GUIDE uses annotation estimates; AdapGuide uses recorded usage. Reductions are relative to GUIDE estimates. K/M denote thousands/millions of tokens; NR marks unavailable values.

Metric	GUIDE	**AdapGuide**
Per-task acquisition and processing		
Model calls	≈ 20.11	6.22 ***↓69.1%***
Tokens (K)	≈ 168.09–214.10	73.99 ***↓56.0–65.4%***
Acquisition latency (s; mean ± SD)	NR	294.96 ± 149.10
Retained-video processing totals		
Processing totals for retained videos	≈ 427	473
Model calls for retained videos	≈ 7,259	946
Tokens for retained videos (M)	60.68–77.29	10.64

cover successful logical model calls with recorded usage across all acquisition stages; the lower block reports processing costs for the retained videos. Acquisition takes 294.96 ± 149.10 seconds per task (mean ± sample SD), measured from the initialized environment through completed tutorial processing.

Effect-guided adaptation during execution. Table 5 provides a descriptive trajectory analysis that groups **AdapGuide** runs by accepted structural revisions after initialization and pairs Base outcomes on the same tasks. Structural revisions change target bindings, operation configurations, or procedure stages; progress tracking records status updates. The Base and AdapGuide columns report the percentage of tasks receiving a positive score (including partial credit), whereas Table 2 reports mean task scores. The revision group starts from a lower Base positive-score rate (54.55% versus 73.36%) but gains **17.42 pp**, compared with 3.49 pp in the stable-guidance group. Full-credit recovery among tasks with Base scores below full credit reaches **40.32%** (25/62), versus 19.72% (14/71), with 27 improvements against three regressions in the revision group. Structural revisions occur in 36.57% of tasks and account for **61.36% of score improvements** and **64.10% of new full-credit successes**. The w/o GR ablation provides the controlled comparison for revision, while this trajectory analysis shows that accepted structural revisions concentrate among recovered tasks that challenge the base agent.

4.4 Ablation Study

Table 6 and Figure 3 compare six GPT-5.5 configurations on the same 361 tasks: Base, direct tutorial injection (Direct), removal of external knowledge (w/o EK), removal of operation–effect relations (w/o OER), removal of online guidance revision (w/o GR), and AdapGuide. Base, Direct, and AdapGuide reuse the results in Table 2.

Direct Tutorial. Direct places retrieved tutorial material in the task context and uses backbone-native planning. It scores 64.82%, 9.55 pp below AdapGuide's explicit, tutorial-derived procedure.

w/o External Knowledge. This variant initializes and revises a task-specific procedure from the backbone's knowledge, user instruction, and initial GUI observation. It scores 68.55% (+2.69 pp over Base); tutorial-derived knowledge in AdapGuide provides a further 5.82 pp gain.

w/o Operation–Effect Relations. This variant uses the same tutorial materials and model-call budget to construct, verify, and revise procedural steps. It scores 69.66% (+3.80 pp over Base); explicit operation–effect relations in AdapGuide provide a further 4.71 pp gain.

Table 5: Structural guidance revision and task outcomes for GPT-5.5 on 361 OSWorld-Verified tasks. **(A)** Task-level positive-score rates (%; score > 0, including partial credit), with gains computed from unrounded rates. **(B)** Paired score changes relative to Base and full-credit recovery k/m (%), where m counts tasks with Base scores below full credit.

A. Outcomes by structural-revision status

Task group	n	Base	AdapGuide	Δ
Revision group	132	54.55	**71.97**	**+17.42**
Stable-guidance group	229	73.36	76.86	+3.49
All tasks	361	66.48	75.07	+8.59

B. Paired score changes and full-credit recovery

Task group	E / H / R	Full-credit recovery
Revision group	**27 / 102 / 3**	**25/62 (40.32)**
Stable-guidance group	17 / 203 / 9	14/71 (19.72)
All tasks	44 / 305 / 12	39/133 (29.32)

E/H/R: improved / unchanged / regressed. Full-credit recovery reports k/m, where k counts tasks receiving full credit under AdapGuide.

Table 6: GPT-5.5 ablations on 361 OSWorld-Verified tasks (50-step limit). Overall is the mean evaluator score (%, including partial credit); Δ is relative to Base (pp). Bold marks the best results.

Configuration	Overall	Δ	E / H / R
Reference configurations			
Base	65.86	–	–
Direct	64.82	−1.04	29 / 297 / 35
Component ablations			
w/o EK	68.55	+2.69	19 / 334 / 8
w/o OER	69.66	+3.80	40 / 296 / 25
w/o GR	63.01	−2.85	53 / 246 / 62
Proposed method			
AdapGuide	**74.37**	**+8.51**	44 / 305 / 12

E/H/R: improved / unchanged / regressed. Counts and Δ are relative to Base.

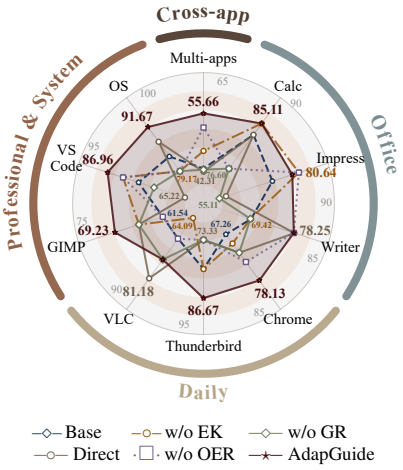

Figure 3: Per-domain scores for Table 6. Axes use separate linear scales; labels mark extrema (%).

w/o Guidance Revision. This variant maintains the tutorial-derived operations, configurations, conditions, and expected effects as fixed guidance ($\mathcal{P}_t = \mathcal{P}_0$), while execution verification updates completion judgments. AdapGuide uses observed effects to revise bindings, configurations, and stages, achieving 74.37% compared with 63.01%.

Together, the ablations support external procedural knowledge, explicit operation–effect relations, and online guidance revision. Relative to Base, AdapGuide records 44 improvements and 12 regressions, compared with 53 and 62 for w/o GR. GUIDE records 42 improvements and 32 regressions (Appendix C.2). Application results further show that **AdapGuide** and w/o EK match on Calc, while w/o EK scores higher on Impress (80.64% versus 78.52%).

4.5 Discussion

Operation–effect relations as a basis for adaptation. The controlled ablations attribute 4.71 pp to operation–effect relations and 11.36 pp to structural guidance revision. Table 5 locates accepted revisions and their recovered outcomes. Grouping tasks by execution trajectory provides a descriptive

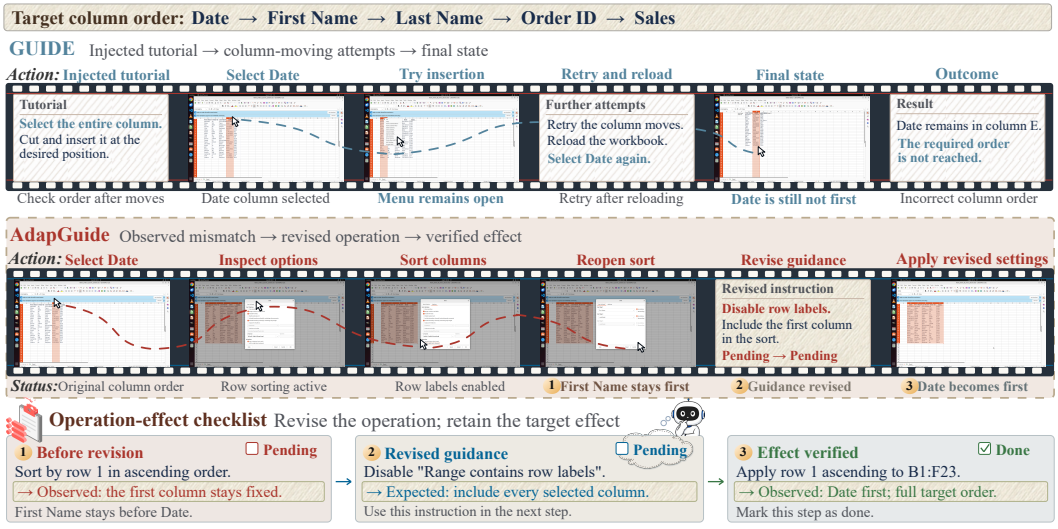

Figure 4: Execution trajectories for column reordering in Calc. GUIDE leaves the order incorrect; **AdapGuide** revises the sort settings, verifies the required column order, and completes the task.

analysis rather than an estimate of the causal effect of revision. Expected effects provide a reference for execution feedback, while revision converts that feedback into changes to task bindings, configurations, and stages.

From observed mismatch to guidance revision. Figure 4 shows how guidance revision achieves the required Calc column order: Date, First Name, Last Name, Order ID, and Sales. GUIDE's attempts leave Date in column E. In the AdapGuide trajectory, sorting initially keeps First Name fixed because the row-label option excludes the first selected column. AdapGuide revises the guidance to disable `Range contains row labels`. Applying the revised sort to `B1:F23` produces the target order and marks the stage `Done`.

Evidence-based completion judgments. Completion auditing evaluates the relationships and outcomes in the original request against interaction evidence. It assigns supported, contradicted, and unresolved judgments from visible effects and interaction records. These judgments reflect the agent's available evidence, not an independent guarantee of task success. Comparing these judgments with benchmark outcomes and aligned trajectories identifies revisions that resolve mismatches and execution errors that persist.

5 Conclusion

In this paper, we presented **AdapGuide**, a training-free framework for transferring tutorial knowledge to GUI tasks through operation–effect relations and revising their task-specific instantiations using execution evidence. On OSWorld-Verified, AdapGuide improves average task scores by 8.08–15.86 pp across five backbones, with further gains of 14.73 and 12.47 pp for GPT-5.5 on Windows Agent Arena and DeskCraft, respectively. Ablations support combining external procedural knowledge with online guidance revision rather than relying on fixed initial guidance with verification and completion tracking alone. Overall, the results support using observed execution outcomes not only to assess progress, but also to revise how source-supported relations are instantiated as task guidance for the current instruction and interface.

Limitations and future work. AdapGuide relies on the relevance of retrieved tutorials and the execution evidence available through the interface. In applications with limited documentation or outcomes that are difficult to observe, reconstructing applicable relations and assessing completion can remain uncertain. Future work should improve the calibration of progress judgments and allocate evidence acquisition and verification effort according to task needs, reducing overhead during inference while preserving the evidence needed for reliable guidance revision and completion decisions throughout the execution of GUI tasks.

AI USE STATEMENT

We used generative AI tools to assist with writing portions of the code, translating and polishing manuscript text, and identifying potentially relevant literature on GUI agents and tutorial-based knowledge acquisition. These tools were not used to originate the research ideas, design the experiments, or generate or alter the reported experimental results. The use of language and vision-language models within the evaluated agent systems is described separately in Sections 3 and 4. The authors reviewed and revised the AI-assisted manuscript text and checked technical statements against the underlying work and original sources. All final decisions were made by the authors, who take full responsibility for the code, results, and final manuscript.

REPRODUCIBILITY STATEMENT

We will publicly release the AdapGuide framework and provide configuration files and usage instructions to support reproducibility and further research. The release will include the pipeline for retrieving web pages and video tutorials at task time, together with the implementation of knowledge reconstruction, guidance adaptation and revision, and execution verification. Appendices A and B provide additional method details, benchmark protocols, and runtime configurations, while Appendix E presents the prompt templates and execution instructions. Together, these materials document the configurations and procedures used in this work and will accompany the planned release.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Wang. Agent S: An open agentic framework that uses computers like a human. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/394c7c30ea87b5c3521b4d9e9d419071-Abstract-Conference.html.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S2: A compositional generalist-specialist framework for computer use agents. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=zg5is4GJ3R.

Anthropic. Claude Opus 5 system card. Technical report, Anthropic, 2026. URL https://anthropic.com/claude-opus-5-system-card.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. Windows Agent Arena: Evaluating multi-modal OS agents at scale. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 4874–4910. PMLR, 2025. URL https://proceedings.mlr.press/v267/bonatti25a.html.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.505. URL https://aclanthology.org/2024.acl-long.505/.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 119919–119948, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d8efbb5dd415974eb095c3f06bff1f48-Abstract-Conference.html.

Google. Gemini 3.6 Flash. Gemini API documentation, 2026. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.6-flash. Accessed: 2026-09-15.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Hong_CogAgent_A_Visual_Language_Model_for_GUI_Agents_CVPR_2024_paper.html.

Jungjae Lee, Dongjae Lee, Chihun Choi, Youngmin Im, Jaeyoung Wi, Kihong Heo, Sangeun Oh, Sunjae Lee, and Insik Shin. VeriSafe Agent: Safeguarding mobile GUI agent via logic-based action verification. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*, pp. 817–831. Association for Computing Machinery, 2025. doi: 10.1145/3680207.3765248. URL https://doi.org/10.1145/3680207.3765248.

Xinwei Long, Kai Tian, Peng Xu, Weibo Gao, Yihua Shao, Guoli Jia, Haozhe Geng, Sa Yang, Jingxuan Li, Huayong Hu, Kaiyan Zhang, Jiaqi Wang, and Bowen Zhou. Guiding the blind: Generalizing GUI agents to unseen websites via multimodal tutorials. In *Computer Vision – ECCV 2026*, volume 17031 of *Lecture Notes in Computer Science*, pp. 373–391. Springer, 2026. doi: 10.1007/978-3-032-37490-5_21. URL https://doi.org/10.1007/978-3-032-37490-5_21.

OpenAI. GPT-5.5 system card. Technical report, OpenAI, 2026a. URL https://deploymentsafety.openai.com/gpt-5-5.

OpenAI. GPT-5.6 Sol model. OpenAI API documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol. Accessed: 2026-09-15.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering automated GUI interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025. URL https://arxiv.org/abs/2501.12326.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/01a83bc2f2732a58e6aa731e659e7101-Abstract-Conference.html.

Wenkai Wang, Tao Xiong, Jingchen Ni, Yunpeng Bao, Xiyun Li, Tianqi Liu, Hongcan Guo, Zilong Huang, and Shengyu Zhang. DeskCraft: Benchmarking desktop agents on professional workflows and human-in-the-loop collaboration. *arXiv preprint arXiv:2606.03103*, 2026. URL https://arxiv.org/abs/2606.03103.

Rui Xie, Zhi Gao, Chenrui Shi, Zirui Shang, Lu Chen, and Qing Li. GUIDE: Resolving domain bias in GUI agents through real-time web video retrieval and plug-and-play annotation. In *Computer Vision – ECCV 2026*, volume 17008 of *Lecture Notes in Computer Science*, pp. 581–598, Cham, 2026. Springer Nature Switzerland. doi: 10.1007/978-3-032-37026-6_32. URL https://doi.org/10.1007/978-3-032-37026-6_32.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Ran Xu, Kaixin Ma, Wenhao Yu, Hongming Zhang, Joyce C. Ho, Carl Yang, and Dong Yu. Retrieval-augmented GUI agents with generative guidelines. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17866–17875. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.emnlp-main.902. URL https://aclanthology.org/2025.emnlp-main.902/.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous GUI interaction. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 69772–69805. PMLR, 2025b. URL https://proceedings.mlr.press/v267/xu25ae.html.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, Ran Xu, Liyuan Pan, Caiming Xiong, and Junnan Li. GTA1: GUI test-time scaling agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10011639.

Hanzhang Zhou, Panrong Tong, Xu Zhang, Quyu Kong, Chenglin Cai, Tianyu Xia, Gongjie Zhang, Jianan Zhang, Long Li, Long Chen, Lei Wang, Gaole Dai, Pengxiang Li, Liangyu Chen, Yue Wang, and Steven Hoi. Qwen-UI-Agent technical report: Toward next-generation real-world centric foundation GUI agents. *arXiv preprint arXiv:2607.28227*, 2026. URL https://arxiv.org/abs/2607.28227.