GUIVISTA: TRAINING GUI AGENTS FROM TUTORIAL VIDEOS WITH OFFLINE CAPABILITY DIAGNOSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Public software tutorial videos offer scalable supervision for GUI-agent mid-training, but their visual transitions often underdetermine the action that caused them. Even when these interactions are used for training, a failed action does not reveal whether the model chose incorrectly or struggled to express and execute the right action reliably. We present GUIVista, a framework that combines reliable trajectory construction with offline capability diagnosis. GUIVista revisits uncertain action proposals using transition evidence and action-family verification before filtering them into grounded, multi-turn training trajectories. It also evaluates checkpoints by asking them to select a successful action from plausible alternatives under the same GUI context, separating action understanding from open-ended execution. On an action-recovery evaluation, GUIVista improves overall action-type and full-action accuracy over prior video-based systems from 24.75% / 8.80% to 50.42% / 31.29%. Under the shared downstream SFT protocol, GUIVista mid-training improves end-to-end OSWorld AVG@4 from 18.40% for SFT-only to 21.75%. The offline diagnostic tracks post-SFT OSWorld performance across matched checkpoints, with Pearson $r = 0.9254$ and Spearman $\rho = 0.9048$. These results show that GUIVista can turn noisy tutorial videos into useful training supervision and provide an efficient signal for comparing the capabilities acquired during mid-training. We will release all code for data construction and evaluation.

1 INTRODUCTION

Graphical user interface (GUI) agents are designed to execute multi-step tasks across diverse software applications (Xie et al., 2024; Wang et al., 2025b). Training these agents requires demonstrations of how to complete tasks in different applications. Human demonstrations provide explicit action records, but extending their coverage requires collecting additional interactions across applications and tasks (Rawles et al., 2023; Wang et al., 2025b). In contrast, public software tutorial videos already document interface operations and multi-step procedures across applications, offering a source of demonstrations without requiring additional human interaction collection (Zhang et al., 2026; Song et al., 2026). Converting these videos into observation–action trajectories therefore offers a way to expand GUI training data beyond directly recorded demonstrations.

Recent work has used video-derived trajectories for GUI mid-training, exposing models to diverse applications and workflows before supervised fine-tuning (SFT) on curated interaction data (Lu et al., 2026). However, tutorial videos provide uneven supervision over the GUI action space. Common click interactions dominate the data, while rare but task-critical operations such as double-clicks, right-clicks, drags, and keyboard actions form a long tail. These actions are difficult to recover from visual transitions because their types and arguments are often only weakly identified by the observed state change. Simply filtering uncertain transitions can therefore remove useful supervision instead of correcting the underlying action proposal. Because these actions are sparse and difficult to recover, the same long-tail structure also complicates capability assessment during mid-training. Mid-training exposes models to GUI states and action distinctions, but before SFT does not ensure a stable, environment-specific action interface. Online evaluation after SFT remains the standard way to measure downstream benefit, but running it for every checkpoint or data configuration is costly. Moreover, its end-to-end outcome can still entangle whether the model selected the correct action with whether it could express and execute that action reliably, providing limited diagnostic feedback about the capability acquired during mid-training.

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.

To this end, we propose **GUIVista** to improve supervision for these long-tail actions while supporting efficient capability assessment. For trajectory construction, GUIVista treats uncertain video transitions as potentially recoverable supervision. It revisits the initial action hypothesis using the transition and its surrounding context, applying action-specific verification to correct recoverable errors before deciding whether to discard the interaction. This correction-before-rejection design aims to retain difficult operations while limiting unsupported labels. We construct GUIVista-1.1M from 25,450 software-tutorial videos spanning 198 applications, yielding 1,265,219 retained interaction steps organized into 29,391 task-level multi-turn trajectories with grounded rationales (median length 41 steps; 42.9% exceed 50 steps).

Beyond trajectory construction, GUIVista also provides a discriminative evaluation inspired by RuDE (Li et al., 2026) that probes action understanding while reducing the confounding effect of unreliable open-ended execution. Each item pairs a reference action from a successful trajectory with state-conditioned counterfactuals under the same context, and asks the model to select the action consistent with task progress. This tests whether a model can recognize the appropriate action before SFT, without requiring it to generate an environment-compatible action or execute it online. The resulting aggregate score provides an offline diagnostic signal for comparing mid-training checkpoints and data configurations, helping identify promising candidates for subsequent SFT and online evaluation. Diagnostic subsets further reveal weaknesses across action types and task contexts, including low-frequency actions.

Experiments show the effectiveness of GUIVista in long-tail action recovery, downstream training, and capability assessment. On the action-level evaluation set under the AgentNetBench protocol, GUIVista achieves higher overall action-type and full-action accuracy than prior video-based IDM baselines, improving from 24.75% / 8.80% for the strongest baseline to 50.42% / 31.29%. The resulting trajectories also provide useful downstream supervision. With the same downstream SFT recipe, models first mid-trained on GUIVista exceed the SFT-only baseline from 200K examples onward and reach 21.75% OSWorld AVG@4 at 800K, compared with 18.40% for SFT-only in end-to-end evaluation. We also find that the aggregate diagnostic score is consistent with post-SFT OSWorld AVG@4, with Pearson $r = 0.9254$ and Spearman $\rho = 0.9048$ across eight matched GUIVista checkpoints, and $r = 0.9254$ and $\rho = 0.9052$ across the broader panel of established GUI models. The diagnostic profiles further expose model weaknesses across action types and task contexts, providing an interpretable view of the capabilities acquired from video-derived supervision.

2 Related Work

2.1 GUI Trajectory Construction

GUI trajectory construction has largely followed two routes: directly recording interactions and recovering them from existing demonstrations. Direct collection pairs interface observations with explicit action records, providing supervision for GUI-agent fine-tuning. Android in the Wild collects mobile interactions (Rawles et al., 2023), Mind2Web and WebLINX provide demonstrations for web navigation (Deng et al., 2023; Lu et al., 2024), and OpenCUA expands collection across desktop applications and operating systems (Wang et al., 2025b). However, extending these datasets to additional applications and workflows requires further demonstration and validation effort. This cost motivates using broader data sources during mid-training to complement curated fine-tuning trajectories (Wang et al., 2025a; Zhang et al., 2025). Public tutorials are a promising source because they already document diverse software operations and multi-step procedures. TongUI constructs trajectories from videos and articles (Zhang et al., 2026), while VideoAgentTrek explores computer-use pretraining from unlabeled videos (Lu et al., 2026). To recover the actions underlying these demonstrations, Watch & Learn uses inverse dynamics over consecutive observations (Song et al., 2026), and GUIDE combines web-video retrieval with annotation to mitigate domain bias (Xie et al., 2026). These approaches reduce dependence on newly collected demonstrations, but inferred action labels can still misrepresent the observed interactions, especially for sparse and task-critical action types. Filtering uncertain transitions can remove useful supervision without correcting the underlying type or argument errors. GUIVista builds on this line of work with correction-before-rejection: it revisits uncertain proposals using transition evidence and action-specific verification before deciding whether to retain them for mid-training.

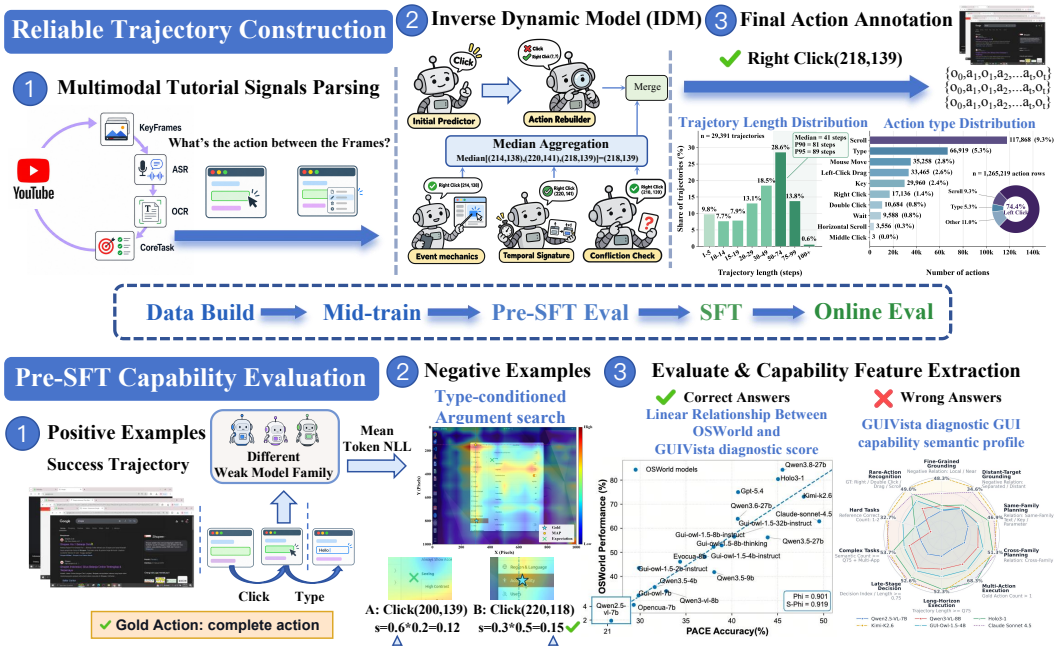


Figure 1: **Overview of the GUIVista framework.** Top: GUIVista converts tutorial videos into verified GUI trajectories through tutorial processing, inverse-dynamics action annotation, evidence-based verification, temporal filtering, and grounded rationale annotation. Bottom: GUIVista's diagnostic constructs a successful action y^+ and state-conditioned counterfactual actions y^- from a weak GUI model family on verified OSWorld trajectories, randomizes their display order, and evaluates checkpoints through multi-choice action discrimination instead of costly online execution.

2.2 GUI Capability Evaluation

GUI-agent evaluation typically measures task completion through execution in interactive environments. WebArena and VisualWebArena assess web navigation (Zhou et al., 2024; Koh et al., 2024), while OSWorld and AndroidWorld evaluate desktop and mobile agents (Xie et al., 2024; Rawles et al., 2025). These benchmarks jointly test perception, action selection, and sequential execution, providing direct evidence of practical task-solving ability. For mid-training checkpoints, however, end-to-end success provides limited diagnostic feedback. Mid-training exposes models to GUI states and action distinctions but does not necessarily establish a stable, environment-specific action interface before SFT, so a failure may reflect either incorrect action selection or unreliable action expression and execution. The standard alternative is to apply SFT and then run online evaluation to measure downstream benefit (Wang et al., 2025a; Lu et al., 2026), but repeating this process for every checkpoint or data configuration is costly and still entangles these failure modes. RuDE offers a complementary approach, using discrimination between valid and controlled counterfactual responses to assess post-training potential without first adapting the model (Li et al., 2026). GUIVista incorporates this principle into its pre-SFT diagnostic, using state-conditioned action alternatives to compare checkpoints and expose action-specific weaknesses.

3 Method

GUIVista combines trajectory construction with capability assessment in video-based GUI mid-training. As shown in Figure 1, it first reconstructs and verifies uncertain action proposals before retaining them as training trajectories. After mid-training, each checkpoint is evaluated by selecting the action consistent with task progress under a fixed GUI context, providing an offline signal before SFT and online execution.

3.1 RELIABLE TRAJECTORY CONSTRUCTION

GUIVista converts each tutorial into a task-conditioned sequence of verified interactions. The construction follows a single progression: we first identify task and state context, then propose an action for each visual transition, rebuild uncertain proposals before filtering, and finally assemble the accepted steps into a multi-turn trajectory. Following prior GUI trajectory formulations (Zhang et al., 2026; Xie et al., 2026), we represent a retained trajectory as

$$\tau = \left(q, \{(o_t, r_t, a_t)\}_{t=1}^{T}\right), \tag{1}$$

where q describes the task, o_t is the observation before an action a_t, and r_t is its grounded rationale.

Task and transition context. We retrieve YouTube tutorials for an application whitelist that combines prior inventories with manually curated additions (Song et al., 2026; Zhang et al., 2026). Titles and transcripts provide task descriptions, while keyframes selected by foreground changes define candidate state transitions.

For each candidate transition (o_t, o_{t+1}), timestamped ASR subtitles provide local procedural context c_t, and an OCR/UI parser supplies element representations (E_t, E_{t+1}) containing text, element types, and locations. Together, these signals connect the observed state change to the task and identify possible interaction targets.

Action proposal. The visual transition is informative but does not always identify the operation that caused it, particularly for sparse action types. An inverse-dynamics annotator uses this multimodal context to propose an action:

$$f_{\text{IDM}} : (o_t, E_t, o_{t+1}, E_{t+1}, q, c_t) \to \tilde{a}_t, \tag{2}$$

where $\tilde{a}_t$, termed the *Base Action*, specifies an initial action type and arguments. The Base Action is only a proposal: it must be reconstructed and verified before entering a training trajectory.

Correction before rejection. Uncertainty can arise from incomplete video evidence or from errors in interpreting the available context. Discarding every uncertain proposal would remove recoverable interactions, including infrequent operations whose visual signatures are ambiguous. GUIVista therefore applies *Action Rebuild* before filtering. Conditioned on $\tilde{a}_t$ and the transition context, specialized text, pointer, and scroll experts inspect the action type and its arguments. Their judgments are combined through role-based voting: acceptance requires agreement on the action type and satisfaction of a shared confidence criterion. For coordinate-based actions, the accepted coordinate proposals are aggregated by taking the median along each axis. Proposals that do not meet the acceptance conditions are discarded. This correction-before-rejection rule retains interactions with recoverable labels while excluding proposals that remain unsupported; its accuracy–coverage trade-off is evaluated in Appendix B.

Trajectory assembly and grounded supervision. The accepted local actions must form coherent sequential supervision. Following the action-compression principles of OpenCUA (Wang et al., 2025b), we absorb cursor movements into clicks or drags, merge consecutive scrolling and typing events, and consolidate multi-step gestures. Ambiguous or non-state-changing events are removed, and sequences are split at discontinuities in step or frame continuity to form task-level trajectories. Once an action is verified, it is held fixed while a rationale is generated from the current observation, task context, and OCR/UI evidence, grounding the explanation in the accepted interaction. At training time, the model predicts (r_t, a_t) from the task q, current observation o_t, and preceding interaction context H_t. This representation supplies task progress alongside each action and yields the multi-turn supervision used for GUI mid-training before downstream SFT.

3.2 DATASET CHARACTERISTICS

Our dataset, GUIVista-1.1M, contains 1,265,219 retained interaction steps from 25,450 public software-tutorial videos, organized into 29,391 multi-turn task-level trajectories. It scales video-derived supervision across 198 applications and 15 categories, including productivity, design, and programming tools; the application distribution is shown in Figure 2. This breadth exposes the model to diverse interface layouts, software concepts, and workflows without requiring a new human demonstration for each application or task.

The construction preserves the temporal structure for multi-turn GUI behavior. Trajectories have a median length of 41 steps, a mean length of 43.0, and 90th and 95th percentiles of 81 and 89 steps, respectively; 42.9% exceed 50 steps. These long sequences connect earlier interactions to later decisions, supplying workflow context beyond isolated state–action pairs.

Our action inventory retains supervision beyond dominant left-click interactions (74.4%): scroll accounts for 9.3% of steps and type for 5.3%, while double-click, right-click, drag, and keyboard actions form a less frequent long tail. Preserving this tail is important for the paper's goal of improving supervision for difficult, task-critical operations.

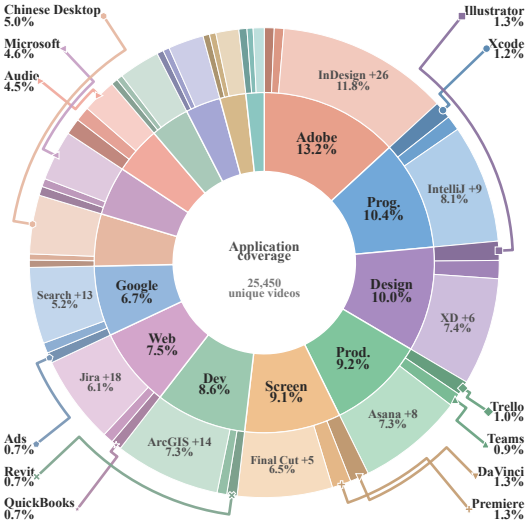


Figure 2: **Application coverage of GUIVista-1.1M.** The training corpus spans 198 applications in 15 categories, including productivity, design, and programming tools.

3.3 GUI Action-Understanding Diagnostic

The resulting trajectories expose models to GUI states and action distinctions, but a mid-trained checkpoint may not yet express those distinctions through a stable, environment-specific action interface. We therefore evaluate action understanding separately from full-task execution. Inspired by RuDE (Li et al., 2026), GUIVista presents a fixed decision context with candidate actions and asks the model to select the action consistent with the demonstrated task progress. This formulation targets action selection directly, including the long-tail action distinctions that are difficult to diagnose from end-to-end success alone.

The evaluation contexts come from successful OSWorld-Verified Ubuntu trajectories (Xie et al., 2024), separately from the public tutorials used to construct the training corpus. We retain complete runs that pass the task evaluator, drawing on the ten highest-scoring public trajectory sources. Each retained run supplies decision contexts $c_i = (q_i, H_i, o_i)$, comprising the task instruction, recent interaction history, and current pre-action screenshot. The executed action y_i^+ serves as the reference for that context. Post-action screenshots are reserved for auditing and withheld from both the models generating alternatives and the checkpoint being evaluated.

Useful alternatives must represent plausible decision errors under the same context. We therefore obtain state-conditioned candidates from a family of weaker GUI models without additional task-specific training, using a frozen deterministic generation configuration. Candidate generation considers several likely action types and predicts their arguments autoregressively. To rank the resulting candidates, we score each action a by its mean-token negative log-likelihood $\ell(a)$, normalizing the sequence score by token count. Raw scores from different generators need not be comparable, so we use an empirical rank within each generator g and first action type κ:

$$r(a) = \frac{1}{|\mathcal{S}_{g,\kappa}|} \sum_{a' \in \mathcal{S}_{g,\kappa}} \mathbf{1}[\ell(a') \le \ell(a)]\,, \tag{3}$$

where $\mathcal{S}_{g,\kappa}$ is the corresponding pool of scored candidates. This rank expresses how likely a candidate is relative to others in the same pool.

Candidates that cannot be parsed, match the reference after normalization, or repeat a recent historical action are removed. We rank the survivors using $r(a)$ and select three alternatives from distinct generators, reducing dependence on any one model's error patterns. Contexts with insufficient surviving alternatives are discarded. The reference and three alternatives form a four-option item; their positions are shuffled with a balanced deterministic permutation, and the resulting evaluation set is frozen across checkpoints. Only the task, recent history, current screenshot, and canonicalized options are presented to the evaluated model; reference identities, generator identities, and construction scores are withheld.

Table 1: **IDM action accuracy ablation (%).** This table shows our IDM's great performance across all action categories. Each action column reports category correctness in the computer-use action space. "Overall" reports two scores under the AgentNetBench protocol (Wang et al., 2025b): action-type accuracy (Type) and action-type-plus-argument accuracy (Type+Arg). We compare TongUI-Crawler from the original public code and VideoAgentTrek-IDM-s2-7B as IDM reproductions (Zhang et al., 2026; Lu et al., 2026). Adding Rebuilder substantially raises Type accuracy; adding experts further improves both Type and Type+Arg and leads these prior IDMs. Best results are highlighted in bold.

Method	Click			Drag	Scroll	Type	Hotkey	Overall
	Left	Double	Right					Type / Type+Arg
TongUI	86.67	7.67	8.00	9.00	8.67	6.33	6.33	16.58(−24.00) / 4.48(−21.02)
VideoAgentTrek	73.67	7.67	18.67	1.00	49.33	9.00	**38.67**	24.75(−15.83) / 8.80(−16.70)
Base Action	**95.33**	16.00	64.00	21.33	34.67	59.67	14.67	40.58 / 25.50
+ Meaningful@0.8	50.00	14.00	61.00	16.00	23.33	46.67	13.67	28.29(−12.29) / 19.15(−6.35)
+ Rebuilder	90.67	22.67	**95.00**	18.00	34.33	59.00	14.33	42.88(+2.30) / 24.87(−0.63)
+ All Experts	93.33	**23.67**	91.00	**24.33**	**54.33**	**84.00**	16.33	**50.42**(+9.84) / **31.29**(+5.79)

For checkpoint m, we define the GUIVista diagnostic score as the unweighted accuracy over the frozen set of N items:

$$\mathcal{D}_m = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[\hat{y}_{m,i} = y_i], \tag{4}$$

where $\mathcal{D}_m$ denotes the diagnostic score for checkpoint m, y_i is the reference option's index, and $\hat{y}_{m,i}$ is the model's selection. Every item contributes equally, with no application-, action-family-, or cluster-specific weighting. Diagnostic subsets examine specific confusions: a spatial alternative with the same action type but a different target probes grounding, while an alternative with a different action type or semantic argument probes action selection. Further breakdowns by action family and task context characterize where reference-action discrimination is weak.

This evaluation provides a common offline signal for comparing checkpoints and reconstruction choices before SFT or online execution. We test its relevance to downstream task performance separately by applying a shared SFT recipe and comparing diagnostic scores with end-to-end OSWorld results. The evaluation-set composition, model configurations, and correlation analysis are reported in Section 4.

4 Experiments

Our experiments are organized into four parts. First, we evaluate whether correction-before-rejection improves the recovery of GUI actions, especially for sparse and task-critical action types. Second, we test whether the resulting trajectories support an effective mid-training-to-SFT pipeline using OSWorld and ScreenSpot-Pro. Third, we use the GUIVista diagnostic to examine what intermediate checkpoints learn, including action-specific improvements. Finally, we test whether the aggregate diagnostic score tracks post-SFT performance across checkpoints and GUI models.

4.1 Trajectory Construction Component Ablation

Evaluation Method. We construct a frozen action-recovery evaluation set from AgentNet (Wang et al., 2025b) execution trajectories, comprising 2,400 step-verified transitions balanced across eight canonical action types, with 300 examples per type. Each example consists of the pre-action frame, the executed action recorded in the original trajectory, and the subsequent frame; we retain only transitions whose next step is marked correct.

We first test whether GUIVista's correction-before-rejection design improves the quality of recovered action supervision. We apply the same frozen action-recovery evaluation protocol to every variant. The *Base Action* condition uses direct multimodal proposals from Holo-3.1-35B-A3B, following GUIDE-style video annotation (Xie et al., 2026; Wang et al., 2025a). We then apply the

Table 2: **Mid-training effectiveness across complementary benchmarks.** ScreenSpot-Pro reports Overall, Text, and Icon scores; OSWorld reports end-to-end AVG@4. The first block reports the confirmed SFT-only and GUIVista two-stage conditions, and the second compares prior video-based systems with the strongest GUIVista checkpoint. Qwen2.5-VL-3B (Qwen Team, 2025) is the baseline for both benchmarks; parenthetical deltas are versus this baseline. TongUI's OSWorld score is our Watch & Learn reproduction trained on Qwen2.5-VL-7B (Song et al., 2026; Qwen Team, 2025). Best scores are in bold.

Method	ScreenSpot-Pro			OSWorld AVG@4
	Overall	Text	Icon	
Matched training conditions				
SFT-only-3B	15.31% (+8.41)	22.62% (+13.52)	3.48% (+0.18)	18.40% (+16.50)
GUIVista Mid Training+SFT-3B	18.98% (+12.08)	27.94% (+18.84)	4.47% (+1.17)	**21.75%** (+19.85)
External model comparison				
Qwen2.5-VL-3B (Qwen Team, 2025)	6.90%	9.10%	3.30%	1.90%
VideoAgentTrek (Lu et al., 2026)	22.01% (+15.11)	30.81% (+21.71)	7.78% (+4.48)	15.78% (+13.88)
TongUI-7B Song et al. (2026)	18.00% (+11.10)	26.40% (+17.30)	4.10% (+0.80)	5.40% (+3.50)
GUIVista Mid Training 3B	**29.22%** (+22.32)	**42.17%** (+33.07)	**8.28%** (+4.98)	10.48% (+8.58)

construction stages incrementally: *Meaningful@0.8* filters proposals by an action-level meaning score, *Rebuilder* reconstructs uncertain proposals into executable actions, and the text, pointer, and scroll experts perform action-family-specific verification.

Table 1 compares both the construction variants and prior video-based action-recovery systems. TongUI and VideoAgentTrek obtain only 16.58% / 4.48% and 24.75% / 8.80% overall action-type / full-action accuracy, respectively, whereas the unfiltered Base Action proposals reach 40.58% / 25.50%. Filtering with Meaningful@0.8 reduces these scores to 28.29% / 19.15%, showing that confidence-based rejection alone discards useful supervision. Rebuilder recovers the action-type score to 42.88%, and the full pipeline with action-family experts achieves the best overall results at 50.42% / 31.29%. The action-family columns show where the additional verification matters most: the final system improves double-click, drag, scroll, and type over the Base Action condition, while right-click changes only slightly from 95.00% to 91.00%. Together, the overall and action-family results support reconstructing uncertain proposals before filtering as a way to improve both the breadth and the reliability of recovered action supervision.

4.2 Mid-Training Effectiveness and Downstream Transfer

After validating the trajectory-construction components, we next test whether the resulting GUIVista trajectories provide useful training supervision. We evaluate matched training conditions on ScreenSpot-Pro and OSWorld, and compare the strongest GUIVista checkpoint with existing video-based GUI models. ScreenSpot-Pro measures intermediate GUI grounding, whereas OSWorld measures end-to-end task completion.

Evaluation Protocol. All matched conditions start from Qwen2.5-VL-3B-Instruct (Qwen Team, 2025). The main reported GUIVista condition uses the 800K checkpoint from the frozen ordering of the 1.32M-prefix mid-training mixture, followed by the independent 150K-prefix ProCUA-SFT release. The mixture contains 1.10M YouTube-derived GUIVista prefixes and 220K prefixes sampled from the ProCUA data source; the independent 150K-prefix ProCUA-SFT release is applied after mid-training. We evaluate ScreenSpot-Pro (Li et al., 2025) under a fixed protocol and run all OSWorld tasks with four rollouts, reporting AVG@4 on OSWorld (Xie et al., 2024). The observation interface, action space, decoding configuration, environment setup, and step budget are held fixed across comparisons. Complete data composition, optimization settings, hardware configuration, and compute statistics are provided in Appendix E.

Table 3: **Diagnostic slices across mid-training scale (%).** Overall is the weighted GUIVista diagnostic score. The remaining columns are Rare-Action Recognition and the four slices with the largest 100K-to-800K gains: Complex Tasks (+11.19), Long-Horizon Execution (+4.34), Fine-Grained Grounding (+4.04), and Same-Family Planning (+3.51). Parenthetical deltas are versus the 100K checkpoint. Best scores are in bold. Base / SFT-only slice scores are not available.

Checkpoint	Overall	Largest 100K–800K gains				Rare-Action Recognition
		Complex Tasks	Long-Horizon Execution	Fine-Grained Grounding	Same-Family Planning	
Base	23.27	24.96	22.95	24.60	27.65	16.41
100K	26.39(+3.12)	21.99(−2.97)	26.94(+3.99)	30.32(+5.72)	25.88(−1.77)	19.99(+3.58)
200K	28.79(+5.52)	25.05(+0.09)	28.72(+5.77)	32.46(+7.86)	29.78(+2.13)	20.53(+4.12)
400K	**29.67**(+6.40)	30.95(+5.99)	**32.07**(+9.12)	31.16(+6.56)	**31.37**(+3.72)	21.09(+4.68)
600K	29.11(+5.84)	29.78(+4.82)	31.66(+8.71)	34.24(+9.64)	29.21(+1.56)	21.80(+5.39)
800K	29.25(+5.98)	**33.18**(+8.22)	31.28(+8.33)	**34.36**(+9.76)	29.39(+1.74)	**22.31**(+5.90)

Results Table 2 shows that GUIVista mid-training provides gains both before and after downstream SFT. Starting from the same Qwen2.5-VL-3B backbone, mid-training raises ScreenSpot-Pro from 6.90% to 29.22% overall, with corresponding gains from 9.10% to 42.17% on text targets and from 3.30% to 8.28% on icon targets, indicating substantially stronger GUI grounding from the additional trajectory supervision. Under the matched SFT setting, adding GUIVista mid-training further improves ScreenSpot-Pro from 15.31% to 18.98% and OSWorld AVG@4 from 18.40% to 21.75%, showing that the learned capability transfers to end-to-end execution rather than being absorbed by SFT alone. The mid-trained 3B model also exceeds the 7B VideoAgentTrek and TongUI baselines across all three ScreenSpot-Pro metrics, despite its smaller model size. After the shared SFT stage, it reaches 21.75% on OSWorld, above VideoAgentTrek at 15.78% and TongUI at 5.40%. Together, these results show that GUIVista mid-training improves both fine-grained GUI grounding and downstream task execution, with gains that persist against larger video-trained baselines.

4.3 Checkpoint Diagnostic and Action-Specific Findings

GUIVista Diagnostic Construction. We construct the GUIVista diagnostic from successful OSWorld-Verified Ubuntu trajectories, separately from the public tutorial videos used for mid-training. For each decision context, we use an off-the-shelf weak-model family without additional task-specific training: Qwen3-VL-8B-Instruct (Bai et al., 2025), Qwen2.5-VL-7B-Instruct (Qwen Team, 2025), UI-TARS-1.5-7B (Wang et al., 2025a)., and EvoCUA-8B-20260105 (Xue et al., 2026). Each item presents a fixed task context, recent interaction history, current screenshot, one executed reference action, and state-conditioned counterfactual actions generated by these models. After parsing and validity filtering, the frozen panel contains 2,849 four-option decision items; candidate provenance, construction scores, and post-action screenshots are withheld from the evaluated checkpoint. The detailed action-family and application composition of the panel is reported in the appendix (Table 7).

Checkpoint Analysis Table 3 shows that the GUIVista diagnostic captures systematic capability gains as mid-training progresses. The overall score increases from 26.39% at 100K to 29.25% at 800K, reaching 29.67% at 400K, especially Rare-Action Recognition improves consistently from 19.99% to 22.31%. The largest gains also appear on challenging subsets targeted by GUIVista, including Complex Tasks (+11.19), Long-Horizon Execution (+4.34), Fine-Grained Grounding (+4.04), and Same-Family Planning (+3.51), indicating that the diagnostic is sensitive to improvements beyond common actions. We also observe that several subsets improve at the earlier checkpoints but then decline at the later stages of mid-training. We hypothesize that these fluctuations stem from the absence of synchronized reasoning-oriented supervision during mid-training, which may lead action recognition to improve more steadily than higher-level planning and reasoning capabilities.

4.4 Diagnostic Validity and Post-SFT Findings

The action-specific analysis above describes what intermediate checkpoints have learned. We next test whether the aggregate diagnostic tracks downstream behavior after SFT.

Using the same frozen $N = 2{,}849$-item panel and canonicalized candidates, we evaluate 19 established GUI models and eight GUIVista configurations formed by four training scales (100K, 200K, 600K, and 800K), each measured before and after the shared SFT recipe; post-action frames, gold labels, candidate provenance, and construction scores are withheld. For Midtrain+SFT, OSWorld AVG@4 increases from 15.91% at 100K to 19.46% at 200K, 20.04% at 600K, and 21.75% at 800K, compared with the 18.40% SFT-only baseline; the corresponding mid-training-only endpoints are 4.38% at 100K and 10.48% at 800K. The diagnostic correlates with online OSWorld at Pearson $r = 0.9254$ and Spearman $\rho = 0.9052$ on the 19-model panel, and at $r = 0.9254$ and $\rho = 0.9048$ across the eight configurations at the four matched training scales. The 19-model result is an in-sample calibration because the same panel is used for fitting and reporting.

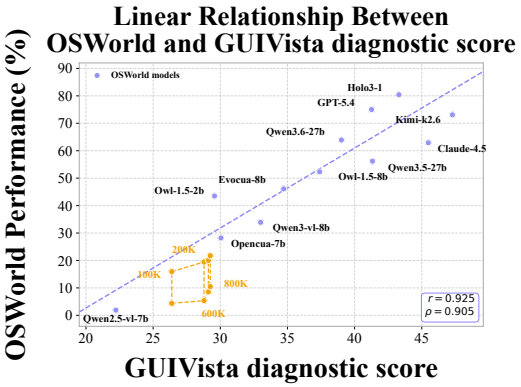


Figure 3: Relationship between recorded OSWorld scores and the GUIVista diagnostic. Twelve representative model points are shown; the dashed fit and the in-sample Pearson $r = 0.9254$ and Spearman $\rho = 0.9052$ use all 19 established GUI models. Orange segments mark four matched training scales (100K, 200K, 600K, and 800K): the lower point is mid-training only and the upper point is Midtrain+SFT.

5 Conclusion

We presented GUIVista, a framework for converting public software tutorials into verified GUI trajectories and analyzing the capabilities acquired during mid-training. Its correction-before-rejection pipeline improves recovered action supervision by preserving interactions that can be repaired while filtering unsupported proposals. Under the evaluated two-stage training protocol, GUIVista supervision provides measurable downstream gains across complementary GUI benchmarks. GUIVista's offline action-selection diagnostic further exposes checkpoint-level and action-specific capability differences before SFT or online execution, and its aggregate scores closely track post-SFT OSWorld performance across matched checkpoints. Together, these results support a practical training and evaluation loop for improving GUI agents from public video supervision.

Limitations

The GUIVista diagnostic is an offline signal for action understanding, not a replacement for online evaluation. Its scores can diverge from environment success, and the current item set is built mainly from successful OSWorld-Verified Ubuntu trajectories. It therefore does not cover other evaluation environments, unsuccessful rollouts, or a broader range of data sources and task distributions.

The production inverse-dynamics annotator infers actions from a pair of consecutive keyframes. In tutorial videos, however, the visible effect of an action may span several keyframes, and some operations, such as certain hotkeys, leave little or no change on the current screen. Restricting inverse dynamics to adjacent frames can miss or mislabel these cases, so the annotator remains limited in temporal span and for actions without a local visual signature.

AI USE STATEMENT

In this work, we used generative AI tools as components of the GUIVista data-construction pipeline, including inverse-dynamics action proposals, verification judgments, and grounded rationale annotation, as described in the method section. We additionally used generative AI tools to help implement and debug data-processing and plotting scripts and to prepare scientific figures. These tools also assisted with language editing, formatting, and manuscript organization. The authors evaluated the feedback, filtered model-produced annotations under the stated procedures, checked analysis outputs against source data and experiment logs, and verified citations and scientific claims. The authors retain full responsibility for the research, the final manuscript, and all reported results.

ETHICS STATEMENT

This work uses publicly available software-tutorial videos and previously released GUI trajectories. Dataset construction applies filtering for privacy, licensing, and sensitive content. Released training prefixes retain screenshots and action labels rather than raw video streams. GUI automation can be misused; the methods target research on computer-use agents and do not provide tools designed to bypass authentication or access private accounts. The authors take responsibility for the data-processing procedures described in the paper and appendix.

REPRODUCIBILITY STATEMENT

Prompt construction and the GUIVista pipeline are described in Section 3. Training, OSWorld, action-recovery, and GUIVista diagnostic protocols are given in Section 4. The desktop action space and compute measurements appear in Appendices A and D. Frozen data-construction settings, diagnostic item selection, and prompt templates are recorded in Appendices E to G. An anonymous supplementary archive provides implementation and evaluation code, including video-pipeline sources, diagnostic construction and scoring scripts, and the recorded training YAML files. We will release the complete codebase for data construction, training, and evaluation.

REFERENCES

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, 2025. URL https://arxiv.org/abs/2511.21631.

H Company. Holo3.1: Fast local computer use agents, 2026. URL https://huggingface.co/Hcompany/Holo-3.1-35B-A3B.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. Advances in Neural Information Processing Systems, 2023.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024.

Kaixin Li, Meng Ziyang, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: GUI grounding for professional high-resolution computer use. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=XaKNDIAHas.

Xiaoyuan Li, Yubo Ma, Kexin Yang, Moxin Li, Keqin Bao, Wenie Wang, Fuli Feng, and Dayiheng Liu. On predicting the post-training potential of pre-trained LLMs, 2026.

Dunjie Lu, Yiheng Xu, Junli Wang, Haoyuan Wu, Xinyuan Wang, Zekun Wang, Junlin Yang, Hongjin SU, Jixuan Chen, Junda Chen, Yuchen Mao, Junyang Lin, Binyuan Hui, and Tao Yu. Videoagenttrek: Computer-use pretraining from unlabeled videos. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=xxYPqm1qWz.

Xing Han Lu, Zdenek Kasner, and Siva Reddy. WebLINX: Real-world website navigation with multi-turn dialogue. Proceedings of the 41st International Conference on Machine Learning, 2024.

Qwen Team. Computer use with Qwen2.5-VL. Official cookbook, https://github.com/QwenLM/Qwen2.5-VL/blob/465b320d4bc365782513d500940ec76e8e06bfb9/cookbooks/utils/agent_function_call.py, 2025. Accessed: 2026-07-27.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control, 2023. URL https://arxiv.org/abs/2307.10088.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=il5yUQsrjC.

Chan Hee Song, Yiwen Song, Palash Goyal, Yu Su, Oriana Riva, Hamid Palangi, and Tomas Pfister. Watch and learn: Learning to use computers from online videos. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2026.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, et al. UI-TARS-2 technical report: Advancing GUI agent with multi-turn reinforcement learning, 2025a.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, LI PEIHANG, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Hu Jiarui, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Y.Charles, Zhilin Yang, and Tao Yu. Opencua: Open foundations for computer-use agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 139756–139806. Curran Associates, Inc., 2025b. doi: 10.52202/085713-4669. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/cc7ae529e945226b0d52ea4ac478c4f3-Paper-Conference.pdf.

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, 2026. URL https://arxiv.org/abs/2603.26266.

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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5d413e48f84dc61244b6be550f1cd8f5-Paper-Datasets_and_Benchmarks_Track.pdf.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, et al. Evocua: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv:2601.15876*, 2026.

Bofei Zhang, Zirui Shang, Zhi Gao, Wang Zhang, Rui Xie, Xiaojian Ma, Tao Yuan, Xinxiao Wu, Song-Chun Zhu, and Qing Li. TongUI: Internet-scale trajectories from multimodal web tutorials for generalized GUI agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. URL https://arxiv.org/abs/2504.12679.

Junlei Zhang, Zichen Ding, Chang Ma, Zijie Chen, Qiushi Sun, Zhenzhong Lan, and Junxian He. Breaking the data barrier—building GUI agents through task generalization. Conference on Language Modeling, 2025.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. International Conference on Learning Representations, 2024.

Table 4: Complete desktop action space used by the Qwen2.5-VL computer-use cookbook. Only `action` is universally required; the arguments in the second column are conditional on the selected action.

Action	Required argument(s)	Description
`key`	`keys`: array of strings	Press the listed keys in order, then release them in reverse order.
`type`	`text`: string	Type a string using the keyboard.
`mouse_move`	`coordinate`: $[x, y]$	Move the cursor to the specified screen coordinate.
`left_click`	—	Click the left mouse button at the current cursor position.
`left_click_drag`	`coordinate`: $[x, y]$	Hold the left mouse button and drag the cursor from its current position to the specified coordinate.
`right_click`	—	Click the right mouse button at the current cursor position.
`middle_click`	—	Click the middle mouse button at the current cursor position.
`double_click`	—	Double-click the left mouse button at the current cursor position.
`scroll`	`pixels`: number	Scroll by the specified amount; positive values scroll up and negative values scroll down.
`wait`	`time`: number	Wait for the specified number of seconds.
`terminate`	`status`: `success` or `failure`	Terminate the current task and report its completion status.

A Qwen2.5-VL Action Space

For reproducibility, Table 4 reports the complete desktop action space defined by the official Qwen2.5-VL computer-use cookbook (Qwen Team, 2025). Coordinates are pixel locations in the resized display, with the origin at the upper-left corner. Click actions operate at the current cursor position; a target can therefore be selected by issuing `mouse_move` before the corresponding click.

B IDM Ablation

This appendix reports the component ablation of our inverse dynamics model (IDM) on the archived evaluation set, corresponding to the IDM analysis in the main experiments. *Base Action* is direct action prediction with Holo3.1 (Company, 2026), following GUIDE-style video annotation and using multimodal video context as the baseline (Xie et al., 2026; Wang et al., 2025a). *Meaningful@0.8* keeps predictions whose action-level meaning score exceeds 0.8, analogous to video-level scoring in prior work but applied to recovered actions. *Rebuilder* converts action proposals into executable sequences. The text, pointer, and scroll experts then specialize this reconstruction for their corresponding action families. Coverage is the archive retention rate, Parse Acc. is primary-action correctness after reconstruction, and Grounding Acc@50 uses a 50-pixel Euclidean click tolerance.

Table 5 shows that filtering alone cannot replace reconstruction. Meaningful@0.8 drops Coverage from 100.00% to 48.58%. Rebuilder converts proposals into executable actions and recovers Coverage to 78.67% with Parse Acc. 42.88%. Adding all experts further improves parseability and click grounding, reaching 93.38% Coverage, 50.42% Parse Acc., and 28.25% Grounding Acc@50. Relative to Meaningful@0.8, Rebuilder restores Coverage and Parse Acc. by turning action proposals into executable steps; adding all experts further raises Parse Acc. and Grounding Acc@50.

The same variants are compared by ground-truth action type in the main experiments (Table 1). Experts raise the overall score from 39.50% to 49.04%, with the largest gains on scroll (11.33% to 51.33%) and type (55.00% to 76.00%); right-click falls from 95.00% to 91.00%.

Table 5: **GUIVista trajectory-construction component ablation on the action-level evaluation set.** Coverage is the archive retention rate, Parse Acc. is primary-action correctness after reconstruction, and Grounding Acc@50 uses a 50-pixel Euclidean click tolerance. Best in each column in bold. Relative to Meaningful@0.8, Rebuilder restores Coverage and Parse Acc. by turning action proposals into executable steps; adding all experts further raises Parse Acc. and Grounding Acc@50.

Method	Coverage (%)	Parse Acc. (%)	Grounding Acc@50 (%)
Base Action	**100.00**	40.58	23.62
Base Action + Meaningful@0.8	48.58	28.29	18.25
Rebuilder	78.67	42.88	23.38
Rebuilder + Pointer	81.88	44.46	24.75
Rebuilder + Pointer + Scroll	86.08	47.04	26.33
Rebuilder + All Experts	93.38	**50.42**	**28.25**

Table 6: Comparison between online GUI performance and pre-SFT GUIVista diagnostic accuracy across established GUI models and GUIVista training configurations. The first block reports published OSWorld scores for 12 GUI models. For GUIVista checkpoints, we report full-set pass@4 over 369 tasks with four rollouts per task.

OSWorld Models		
Model / Checkpoint	**OSWorld (%)**	**Diagnostic Acc. (%)**
Qwen2.5VL-7B(Baseline)	1.90	41.20
OpenCUA-7B	28.20	46.46
GUI-Owl-7B	32.10	46.22
Qwen3-VL-8B	33.90	47.38
Qwen3.5-4B	35.60	49.23
EvoCUA-8B	46.10	49.18
Qwen3.5-27B	56.20	54.76
GUI-Owl-1.5-32B-Instruct	56.50	55.22
Claude-Sonnet-4.5	62.90	62.77
Kimi-K2.6	73.10	57.49
GPT-5.4	75.00	57.12
Holo3.1	80.40	59.02

C OSWorld 2.0 Transfer Analysis

Figure 5 presents a separate transfer analysis using the available OSWorld 2.0 single-action scores for six API models. This setting is out of domain relative to the OSWorld trajectories used to construct PACE and is therefore treated as complementary evidence rather than being included in the primary OSWorld linear fit. The relationship observed in this setting indicates that PACE retains predictive value under distribution shift.

D Compute Cost of a Data Decision

Completed Qwen2.5-VL-3B runs on 16 NVIDIA A100-80GB GPUs measure mid-training on the frozen 1.32M-prefix release at 493 optimizer steps, 61.49 hours wall-clock, and 983.82 A100-hours, and the subsequent 150K ProCUA-SFT stage at 106 steps, 11.09 hours, and 177.49 A100-hours.

After a mid-trained checkpoint is already in hand, exposing its post-SFT OSWorld return is dominated by this SFT run (177.49 A100-hours), plus online evaluation of all 369 OSWorld tasks with four rollouts. Prorating the measured mid-training cost to a 100K prefix block gives $983.82 \times (100/1320) \approx 74.5$ A100-hours (about 4.7 hours wall-clock on 16 GPUs). A comparison loop that includes 100K mid-training, the shared ProCUA-SFT recipe, and OSWorld evaluation is

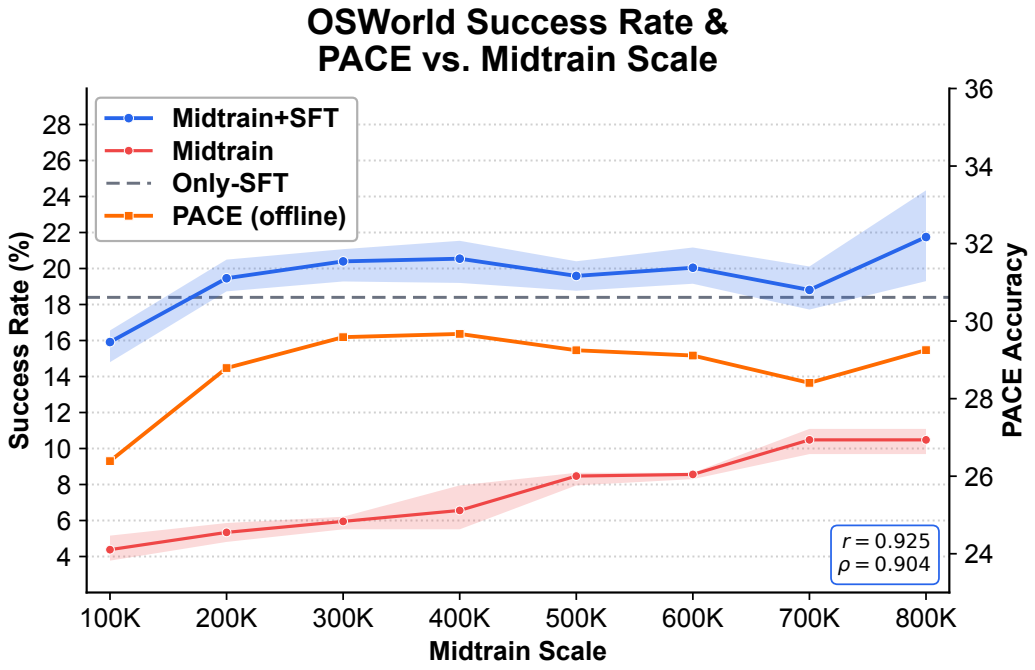


Figure 4: **OSWorld success and pre-SFT diagnostic score versus mid-training scale.** The dashed line is the SFT-only baseline (18.40%). Mid-training-only OSWorld stays in a lower band; the shared SFT recipe reaches its best result at 800K, and the diagnostic tracks the post-SFT trend across the eight checkpoints (Pearson $r = 0.9254$, Spearman $\rho = 0.9048$).

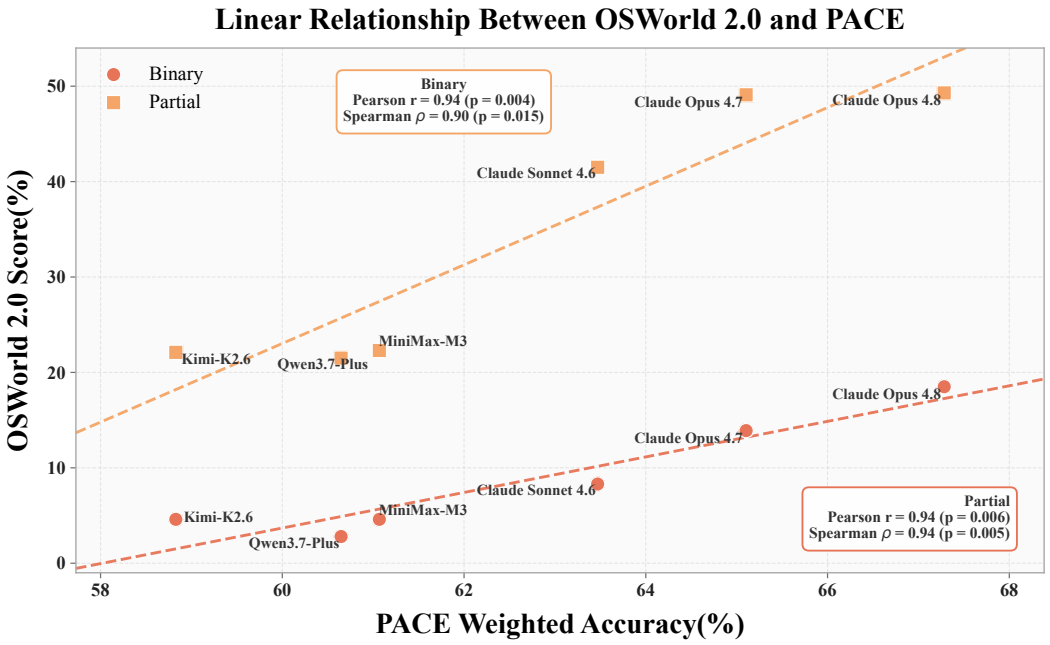


Figure 5: PACE unweighted accuracy versus the available OSWorld 2.0 single-action scores for six API models. Pearson and Spearman correlations are reported for both the binary and partial-binary evaluation settings.

Table 7: Composition of the frozen 2,849-instance GUIVista diagnostic panel. The *Other* application row aggregates the remaining five of the ten application groups.

(a) Gold-action family

Action Family	Count	Share
Point	1,378	48.4%
Keyboard	812	28.5%
Other	318	11.2%
Scroll	134	4.7%
Multi-action	113	4.0%
Drag	94	3.3%
Total	2,849	100.0%

(b) Application composition

Application Group	Count	Share
Multi-app	771	27.1%
LibreOffice Calc	384	13.5%
LibreOffice Impress	368	12.9%
Chrome	236	8.3%
GIMP	232	8.1%
Other	858	30.1%
Total	2,849	100.0%

therefore at least $74.5 + 177.49 = 252$ A100-hours before counting environment rollouts. These figures replace earlier 8-GPU SFT estimates and are distinct from the full 1.32M mid-training cost.

E Data Construction and Training Protocol

This section records the frozen GUIVista mid-training and ProCUA-SFT protocol used in the experiments. A prefix predicts one final action; prefix counts are neither video counts, trajectory counts, nor packed-sequence counts. The current mid-training release contains 1,320,000 prefixes: 1,100,000 from YouTube and 220,000 from AgentNet, selected at a 5:1 ratio and globally ordered with seed 42. ProCUA is not mixed into this stage. An independent 150,000-prefix ProCUA-SFT release is applied after the selected GUIVista checkpoint. An earlier 739,220+140,000=879,220 YouTube–ProCUA mixture is superseded and is not the training corpus reported in Section 4.

YouTube construction enumerates 19,003 successful video records in the frozen preprocessing manifest. ASR uses faster-whisper `tiny`; keyframes are selected from uniformly spaced timestamps with a 400-point cap and a grayscale mean-absolute-difference gate; OCR uses PP-OCRv6 medium detection and recognition. The production IDM is Holo-3.1-35B-A3B with a title-first instruction. Each proposal uses the before and after frames. Contrastive-C votes over eight action classes with abstention allowed; a mutually exclusive text, pointer, or scroll specialist then applies three roles, requiring unanimous support with each evidence strength at least 0.8. The builder validates 1,109,797 eligible YouTube and 239,456 AgentNet candidates, then selects 1,100,000 and 220,000 without replacement. YouTube uses a deterministic uniform rank; AgentNet uses platform/application/action-family buckets with coverage-first allocation and remaining mass proportional to the square root of bucket size.

Training uses Qwen2.5-VL-3B-Instruct with full-parameter fine-tuning, a 32,768-token cutoff, packed sequences, history masking, per-device batch size 1, gradient accumulation 32 (global batch 512), AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$), learning rate 1×10^{-5}, a constant schedule with 30 warmup steps, weight decay 0.1, and one epoch, on 16 NVIDIA A100-80GB GPUs. The completed mid-training run used 493 optimizer steps (7.795B input tokens, 983.82 A100-hours); ProCUA-SFT used 106 steps (1.632B tokens, 177.49 A100-hours). The trainer-side logical-prefix counter of 1,320,069 is pack-accounting and does not revise the 1,320,000-row release. TongUI-3B YAML files are included in the anonymous code archive as configured branches and were not completed at the audit cutoff.

F PACE Construction, Evaluation, and Code Supplement

PACE items are built from successful Ubuntu trajectories in ten public OSWorld-Verified archives. Gold is the current decision from a successful trajectory, not a vote among generators. Each anchor requires two representable historical decisions, the current screenshot, and a subsequent screenshot reserved for audit and withheld from evaluation. Four generators produce state-conditioned negatives: Qwen3-VL-8B-Instruct (Bai et al., 2025), Qwen2.5-VL-7B-Instruct (Qwen Team, 2025), UI-TARS-1.5-7B (Wang et al., 2025a), and EvoCUA-8B-20260105 (Xue et al., 2026). The four-model

branch keeps at most one negative per generator and requires three eligible negatives. Quality filtering of $4{,}000$ constructed items leaves $3{,}952$ eligible examples and a $3{,}139$-item parent set; final selection yields $2{,}849$ frozen four-option questions with Gold-position counts $713/712/712/712$.

The evaluation request supplies the task, two historical screenshot–action pairs, the current screenshot, and four canonical candidates. Gold identity, candidate provenance, and the subsequent image are withheld. Local decoding uses temperature 0 and greedy search; API budgets are heterogeneous and are recorded in the supplementary decoding table. For model m, PACE is unweighted accuracy over the $N = 2{,}849$ items. Pearson r and Spearman ρ in the main text are computed on the frozen 19-model panel and the eight GUIVista checkpoints; because the 19-model panel participated in item selection, those correlations are in-sample calibrations rather than held-out tests. The appendix audit recomputes a selection-gate correlation on the same matrix; we report the main-text values $r = 0.9254$ and $\rho = 0.9052$ for the 19-model panel (Figure 3).

The anonymous supplementary archive contains inspectable core code rather than a self-contained weight or full-data release: video-pipeline sources under `src/video_pipeline/`, PACE construction and scoring under `src/gui_benchmark/`, recorded training YAML files under `config/training/`, and prompt catalogs under `docs/`. Historical GPU launchers are references and are not commands to run without configured inputs, model weights, and image archives.

G Prompt Templates

The following boxes show the production-path templates used for training-format supervision, H0 proposal, Contrastive-C, routed specialists, and PACE evaluation. Dynamic builders, CoT recovery variants, and action-cleaning few-shots remain in the supplementary prompt catalog.

GUIVista training system prompt

[System]
You are a GUI agent. Use the task instruction, screenshots, and previous interactions to perform the next action toward completing the task.

Coordinates are absolute pixel positions on the corresponding processed screenshot, measured from its top-left corner. The current screenshot is {`width`} pixels wide and {`height`} pixels high.

[Response format]
`## Action:`
A concise description of the next action and its target, without numeric coordinates, executable code, or reasoning.
`## Code:`
A Python fence containing executable PyAutoGUI code (multiple lines allowed).

H0 inverse-dynamics proposal

[System]
You are Holo3 performing GUI inverse dynamics annotation. Given two consecutive screenshots, infer the user action that caused Frame 0 (before) to become Frame 1 (after). Do not continue the task and do not plan the next action.

Choose exactly one native tool call. Coordinates are integers in $[0, 1000]$ relative to the screenshot, origin at the top-left. Use click, double_click, right_click, drag, write, keypress, scroll, wait, or answer. Drag is for held-button motion; write when text appeared; keypress for shortcuts; wait when no clear user input can be identified.

[User]
`<task>` {`title-first instruction`} `</task>`
Infer the action that already happened between Frame 0 and Frame 1.
Optional ASR context and OCR element lists for Frame 0 and Frame 1.
Frame 0 image, then Frame 1 image.

Contrastive-C closed-set verification

[System]
You are a strict GUI transition verifier. Return only schema-valid JSON.

[User]
Compare Frame 0 (before) and Frame 1 (after) and identify the visible GUI primitive. Task context may resolve semantics but cannot replace visible transition evidence. Choose exactly one candidate only when supported; otherwise set `selected_action=null` and `abstain=true`.

Role instruction (one of visual_delta, causal_primitive, contradiction).
`Task:` {topic}. `H0 proposed action:` {proposal} (a proposal, not evidence).
Candidates are a closed set among left click, double click, right click, left-click drag, scroll, key, type, and mouse move.
Frame 0 image, then Frame 1 image.

[Schema]
`selected_action` (string or null), `evidence_strength` $\in [0, 1]$, `observed_effect`, `abstain`.

Routed specialist (text / pointer / scroll)

[System]
You are a {text | pointer | scroll} GUI transition specialist. Return only schema-valid JSON.

[User]
Infer the exact already-completed GUI action between Frame 0 and Frame 1. The frozen C/H0 outputs are proposals, never labels.

One routed component is invoked, with three roles. Text uses command_semantics, text_delta, and counterfactual_skeptic; pointer uses event_mechanics, temporal_signature, and counterfactual_skeptic; scroll uses viewport_motion, cursor_anchor, and counterfactual_skeptic.

Acceptance in production requires unanimous support with each evidence strength at least 0.8. Frame 0 image, then Frame 1 image.

[Schema]
`tool_call` (native tool object or null), `evidence_strength`, `observed_effect`, `abstain`.

PACE evaluation

[System]
You evaluate multiple proposed GUI-agent NEXT actions. Use the task, visible current GUI state, and trajectory history. Choose the single action an expert should execute now. Compare task progress, visual grounding, action semantics, parameters or coordinates, temporal fit, and safety. Never prefer a candidate because of its letter or displayed position. Return only the required JSON object with the winning letter and one concise evidence sentence.

[User]
Task instruction; two historical screenshot–action pairs; current screenshot; four canonical action candidates in displayed order A–D. Gold identity, candidate provenance, and the subsequent screenshot are withheld.

[Schema]
`winner` $\in \{A, B, C, D\}$ and a one-sentence `evidence` field. A 32-token winner-only JSON variant omits evidence and is used only for a recorded subset of requests.