GUI-ACE: ACTION-CONDITIONED EXPLANATION OF GUI INTERACTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

To serve as practical personal assistants, graphical user interface (GUI) agents need to execute tasks reliably and provide personalized services. Interaction records can support large-scale agent training and user behavior analysis, and their reuse benefits from understanding recorded actions' targets and functions. General vision-language models (VLMs) can describe these actions, but large models are costly at scale. They also struggle to align numerical action coordinates with visual user interface (UI) elements. Even with accurate point localization, they may misidentify the complete functional unit receiving the action. Element boxes or an additional screenshot can aid target identification but may be unavailable or add overhead. We therefore introduce ActionScope, the smallest complete UI region corresponding to the action-receiving functional unit. This region specifies which functional unit the explanation should describe. Building on ActionScope, we propose Action-Conditioned Explanation (ACE), a compact VLM that takes a single screenshot, action parameters, and available previous-step textual context. It generates target and purpose descriptions conditioned on its predicted ActionScope bounding box field. Our ACE-Curator pipeline generates ActionScope and semantic annotations through region guidance and quality control. We use these annotations to train 1.3B and 8B variants. On an evaluation set with human-verified reference annotations, ACE-1.3B and ACE-8B achieve 92.50% and 94.54% joint semantic accuracy, respectively. Single-screenshot ACE-8B outperforms Qwen3.8-2.4T (Max API) and Claude Opus 5 with an additional screenshot in overall joint semantic accuracy. Ablations show that ActionScope supervision improves joint semantic accuracy. Replacing predicted boxes with incorrect ones reduces subsequent explanation accuracy. Together, these findings support ActionScope as an intermediate representation connecting spatial evidence to semantic explanation. For offline GUI navigation and grounding, we train models with instructions reconstructed from ACE descriptions. These models outperform those trained with original instructions on several benchmarks.

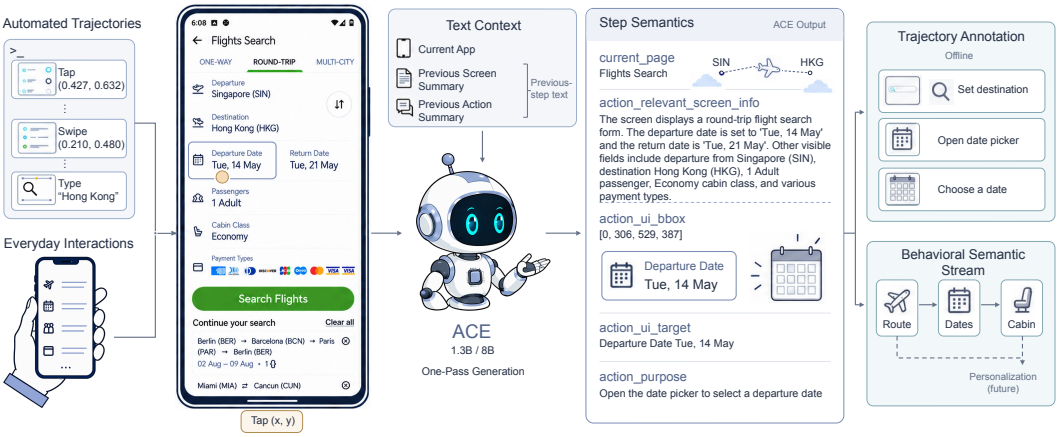


Figure 1: GUI action explanation with ACE. UI and downstream examples are illustrative.

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.

1 INTRODUCTION

Graphical user interface (GUI) agents follow natural language instructions to complete tasks for users. To provide practical personal assistance, they must execute tasks reliably and tailor services to users' preferences and current activities (Yang et al., 2026a; Lyu et al., 2026). Improving execution requires large-scale training on high-quality interaction examples, while analyzing users' everyday actions informs personalization. Screenshots, action types, and parameters can be recorded during automated exploration and everyday use. However, these records lack explicit semantic annotations specifying each action's target user interface (UI) element and function. We therefore study how to recover accurate action semantics for training data construction and user behavior analysis.

Recent vision-language models (VLMs) have made progress in GUI understanding. However, they can struggle with cross-modal spatial alignment between recorded numerical action coordinates and visual features. Even with correct localization, identifying the complete functional unit receiving the action among local content, independent child controls, and parent containers remains challenging. Either error can redirect explanations to the wrong UI element, causing action target drift.

To reduce reliance on numerical coordinate alignment, existing semantic annotation methods use element boxes from interface metadata or visual parsers, pre- and post-action screenshots, or both to provide candidate locations and state-change cues (Lu et al., 2025; Sun et al., 2025; Lu et al., 2026). However, they do not explicitly model which UI unit receives the action. Box granularity may not match this unit, while screenshot pairs may be ambiguous without visible feedback or when different controls cause similar changes. Thus, these auxiliary inputs can ease localization but do not fully address the subsequent identification of the functional recipient. Moreover, these inputs are not consistently available. Obtaining boxes and capturing, pairing, and processing extra screenshots add overhead, limiting scalability alongside inference costs of large VLMs.

We aim to reduce action target drift and reliance on additional visual inputs. Accurate explanations require identifying the complete functional unit receiving the action. We therefore introduce ActionScope, the smallest complete UI region corresponding to this unit, to supervise region prediction and guide subsequent explanations. Designed for human users, GUIs convey control functions through familiar visual conventions and text labels. Once the recipient is identified, these cues often suffice to infer the action's local purpose from one screenshot without comparing pre- and post-action states. This motivates single-image inference with action parameters and available previous-step textual context, requiring neither external element boxes nor a second screenshot. Building on these choices, we propose Action-Conditioned Explanation (ACE), a compact VLM trained with ActionScope supervision. In one autoregressive sequence, ACE predicts a bounding box field and conditions subsequent target and purpose descriptions on it.

Supervision ablations and bounding box interventions assess ActionScope's role. Joint semantic accuracy improves with ActionScope supervision but declines when incorrect boxes replace predicted ones, supporting ActionScope's intermediate role between spatial evidence and action explanations.

Our main contributions are as follows:

1. We identify a representational gap between raw action parameters and functional semantics and introduce ActionScope as an explicit intermediate representation.

2. We develop ACE-1.3B and ACE-8B for single-image action explanation. The ACE-Curator pipeline combines region guidance with quality control to produce region and semantic training annotations. We train both models on 298,503 Chinese and English annotation records covering 196,761 unique action events.

3. We evaluate annotation quality across pipeline stages and explanation accuracy across models and visual input settings. Supervision ablations and bounding box interventions examine ActionScope's role. On several benchmarks, replacing original training instructions with ACE-reconstructed ones improves offline GUI navigation and grounding performance.

2 RELATED WORK

GUI grounding and captioning. GUI grounding locates an element from a description, as in SeeClick (Cheng et al., 2024) and UGround (Gou et al., 2025). UI element captioning describes

the content and function of a specified control or region, as in Widget Captioning (Li et al., 2020) and Spotlight (Li & Li, 2023). Both AutoGUI (Li et al., 2025) and Ferret-UI (You et al., 2025) are trained to describe UI elements specified by points. In our evaluation, however, both models often produce overly broad target descriptions or refer to other elements. To reduce these errors, we supervise ActionScope prediction to align descriptions with the UI element receiving the action.

Semantic annotation of GUI trajectories. AitZ (Zhang et al., 2024), GUIOdyssey (Lu et al., 2025), OS-Genesis (Sun et al., 2025), and GUI-ReWalk (Lu et al., 2026) use general VLMs to generate screen, action, or task descriptions for GUI agent training. Depending on the pipeline, target locations are indicated by boxes drawn on screenshots or by local crops, while pre- and post-action screenshots can provide evidence about state changes. OS-Genesis obtains element locations from accessibility trees and interface metadata, while GUI-ReWalk detects elements with OmniParser (Lu et al., 2024). We train ACE specifically for action explanation from a single screenshot and action evidence. Appendix E compares the input requirements.

User behavior understanding. Research on personalization and proactive assistance, including FingerTip 20K (Yang et al., 2026a), PersonalAlign (Lyu et al., 2026), and KnowU-Bench (Chen et al., 2026), uses textual descriptions of user intents or behavior. Other work derives semantics from visual records. Small Models, Big Results (Cohen et al., 2025) summarizes interactions using UI element information and adjacent screenshots, then extracts intent from those summaries. GUIDE (Yang et al., 2026c) evaluates behavior state detection, intent prediction, and help prediction from screen recordings. These studies use existing intent or behavior descriptions, or derive them using UI element information and multiple visual observations. We focus on acquiring accurate action semantics at low overhead from readily collected single screenshots and raw action parameters, providing input for subsequent behavior analysis.

3 METHOD

ACE-Curator provides region and semantic labels under a shared ActionScope definition, jointly supervising ACE's target localization and local function explanation.

3.1 ACE: MODEL DESIGN WITH ACTIONSCOPE SUPERVISION

Single-image input. ACE takes $X = (I, a, h)$, where I is a single screenshot selected for the action type (Table 1), a contains the action type and available parameters, and h contains the previous step's screen and action descriptions when available (previous-step context). GUI controls communicate their functions to users through text, icons, layout, and visible states, and controls of the same type often share interaction semantics. Once the target is correctly identified, a single screenshot with page context and available previous-step context usually suffices to describe the target UI element and the action's local function. Using one screenshot avoids capturing, pairing, and encoding an additional screenshot of the interface state, reducing waiting time and visual processing overhead for continuous action explanation and resource-constrained local inference.

ActionScope definition and supervision. ActionScope is the smallest complete UI region corresponding to the functional unit that actually receives the action. In layered interfaces, this unit belongs to the layer reached by the action. The region covers that unit as a whole, even when it contains multiple visual components. Shrinking the region would split the target UI element, while enlarging it would include unrelated content. We represent ActionScope by a bounding box $b = [x_1, y_1, x_2, y_2]$ in a coordinate system normalized to the range 0 to 1000.

ACE generates five fields in order. Let C group the current page and action-relevant screen descriptions, B denote the serialized box b, and T, P denote target and purpose descriptions. With parameters θ, the output distribution factorizes as

$$p_\theta(C, B, T, P \mid X) = p_\theta(C \mid X)\, p_\theta(B \mid X, C)\, p_\theta(T, P \mid X, C, B). \tag{1}$$

The screen fields provide context without selecting the recipient. The model then generates T, P conditioned on B, X, and preceding fields.

Training and inference. ACE-1.3B uses MiniCPM-V-4.6, a smaller model that supports 16x visual token compression and mobile deployment (OpenBMB, 2026). ACE-8B uses Qwen3-VL-8B

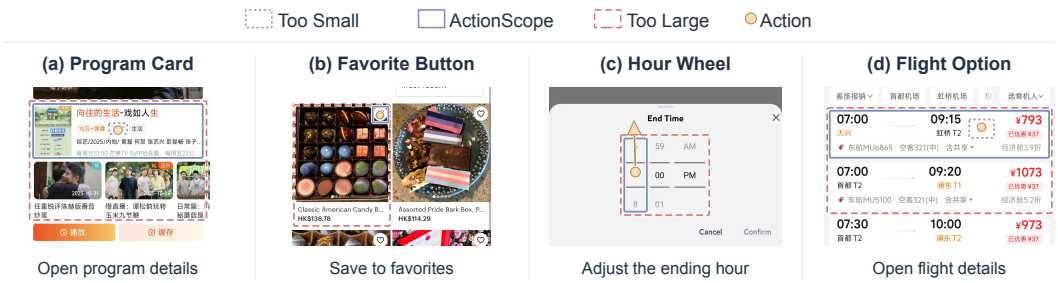


Figure 2: ActionScope examples. Dashed boxes indicate undersized or oversized regions.

Table 1: Localization evidence and ActionScope for each action type.

Action	Evidence	ActionScope
Point	Action coordinates and type	Control or container containing the action point
Type	Previous click, visible text, and editing state	Editable region that receives the text
Swipe	Start and end points	Gesture-receiving region containing the start point

Note: Point includes clicks and long presses. Point and Swipe use the screenshot before the action; Type uses the screenshot after text entry.

(Bai et al., 2025) to examine explanation quality with a larger backbone. Training begins with GUI domain adaptation to develop basic GUI understanding, followed by action explanation fine-tuning. We adapt both backbones on public GUI data using low-rank adaptation (LoRA) with rank 32 (Hu et al., 2022), then merge the adapter weights into each to obtain the respective GUI-adapted backbones. Starting from these GUI-adapted backbones, we fine-tune their language components using newly initialized rank-128 LoRA adapters on action explanation data annotated by ACE-Curator. With the final adapters merged, each model encodes the unmarked screenshot once at inference and generates all five fields in a single autoregressive sequence. Appendix C.1 provides model components, training data, and hyperparameters.

3.2 ACE-Curator: Semantic annotation guided by ActionScope

Stage 1: Candidate localization. Given the screenshot, action, and available previous-step context, a localization model proposes a candidate box $b^{(1)}$ for refinement in Stage 2. We add visual location markers for point actions and provide the entered text for typing actions.

Stage 2: Visual correction. An initial box may contain the action point yet cover only part of the target, include unrelated controls, or select the wrong level of the UI hierarchy. We draw the initial box and any necessary action markers on the input screenshot. Using the ActionScope criteria, the model identifies the target by distinguishing foreground from background and independent child controls from parent containers. It then retains, expands, shrinks, or replaces the region to obtain the final box b. The final box must pass validity checks and satisfy the action containment constraint; otherwise, the sample is referred for manual review.

Stage 3: Separate screen and action annotation. Region markers help identify the target UI element but may be mistaken for interface content in screen descriptions, so we generate screen and action descriptions separately. The screen pass uses the unmarked full screenshot to describe the current page and screen information relevant to the action. The action pass generates the target and purpose from the full screenshot with the final ActionScope marked and any necessary enlarged region crop, optionally consulting a screenshot after the action. The outputs of the two passes and the final bounding box form a five-field annotation. Annotations with missing or mismatched content or marker contamination are referred for manual review.

Stage 4: Quality filtering by risk. We first apply rule-based validation to check JSON parseability, completeness of required fields, bounding box validity within the normalized coordinate range, and containment of the action point or swipe start point. Annotations that fail any check are referred directly for manual review.

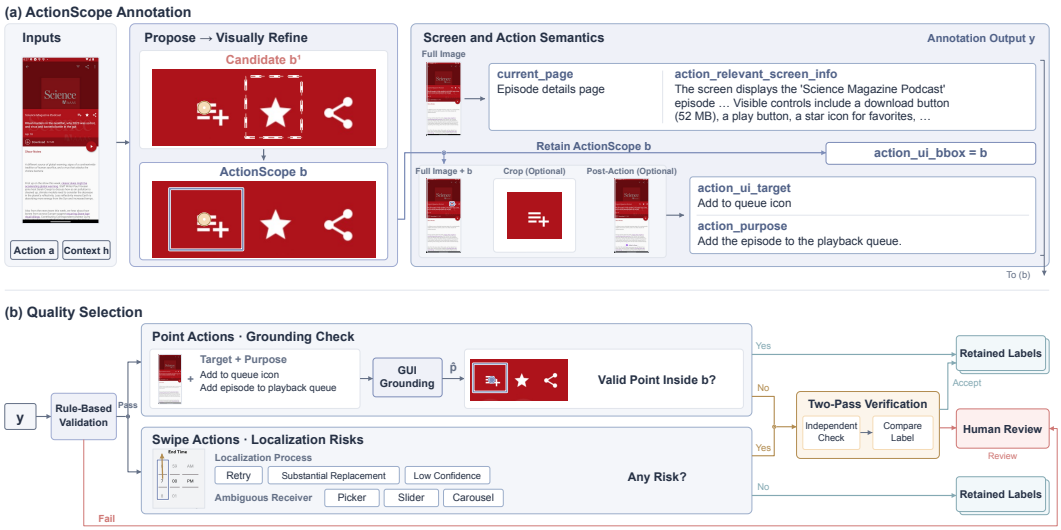


Figure 3: ACE-Curator annotation pipeline.

For point actions passing rule-based validation, we obtain $\hat{q} = \mathcal{N}(G(I, T \oplus P))$ using a GUI grounding model G with the unmarked screenshot and concatenated target and purpose descriptions. We flag invalid predictions or points outside the final ActionScope region $R(b)$.

$$r_{\text{ground}} = \mathbf{1}[\hat{q} = \perp \ \vee \ \hat{q} \notin R(b)]\,. \tag{2}$$

Here, $\mathcal{N}$ converts valid outputs to the coordinates of b and maps invalid outputs to $\perp$; $\oplus$ denotes text concatenation and $\mathbf{1}[\cdot]$ is the indicator function. A control may allow several click locations, so the predicted point need not match the action point.

Because grounding can be incorrect, $r_{\text{ground}} = 1$ triggers verification rather than rejection. Swipes trigger verification when localization is uncertain or involves a local control that is easily confused with others. The annotation verifier checks each referred sample in two passes using the screenshot, action, previous-step context, and region. Samples that pass retain their original annotations, and the others are referred for manual review. Samples that trigger no risk condition are retained automatically. Appendices B.1–B.2 and F provide rules and prompts.

4 Experiments

We first evaluate annotation quality, then ACE's action explanation performance, and finally the utility of its annotations for downstream training. All models are evaluated in non-thinking mode.

4.1 Data and evaluation protocol

The final action explanation pool contains 199,456 unique action events and 301,456 Chinese and English annotation records. ACE-Curator automatically annotated all events; humans reviewed 4,791 and corrected errors where needed. We select 2,695 evaluation actions with a nearly 1:1 ratio of Standard to Hard samples, then construct an annotation analysis set of 1,500 samples and a model evaluation set of 1,666 samples. The former emphasizes dense control layouts, parent-child or foreground-background ambiguity, and context dependence. The latter covers applications, actions, and target types across everyday and complex scenarios. Reference annotations in both sets are verified and corrected by humans. After excluding all evaluation actions and their language versions, both ACE models are fine-tuned on the same 196,761 unique action events and 298,503 Chinese and English annotation records.

Standard samples have readily identifiable action targets and functions. Hard samples involve identifying small targets, distinguishing similar controls, resolving which parent, child, foreground, or background element receives the action, and discerning subtle differences in state or function. Appendix A.1 provides the grading criteria.

Annotation and model evaluations use two checks. Target verification checks whether the target field identifies the UI element receiving the action, independently of the generated purpose. This prevents broad functional descriptions or supplementary information in the purpose field from masking target errors. Joint semantic assessment evaluates whether both fields together correctly describe the object and function, allowing complementary information. Point actions and swipes must pass both checks, whereas typing requires only joint semantic assessment.

Each check follows a two-stage procedure. A language model first compares the relevant fields with human-verified references, accepting semantically equivalent descriptions and rejecting clear semantic contradictions. Only uncertain cases then proceed to visual review, where a VLM uses screenshots and action parameters to resolve the judgment.

Joint semantic accuracy is the fraction of samples passing the applicable checks, excluding factual accuracy of the screen fields and ActionScope box geometry. We select 100 actions by stratified random sampling over action type and difficulty and collect 300 responses from three models. Final evaluation decisions agree with human judgments on 99.33% of responses (298/300). Appendices A.2 and A.3 detail the judging protocol and agreement with human assessment.

4.2 Annotation quality and contributions of pipeline stages

We use Qwen3.5-35B-A3B for localization and semantic generation and Qwen3.5-122B-A10B for annotation verification (Qwen Team, 2026a). POINTS-GUI-G (Zhao et al., 2026) performs grounding-based consistency checks.

In Table 2, Direct generates all five fields without a separately supplied region. Initial region conditions semantic annotation on the candidate box from Stage 1, without visual correction. Corrected region uses the box refined in Stage 2 for semantic annotation in Stage 3. The complete pipeline additionally applies Stage 4 quality filtering.

Table 2: Joint semantic accuracy (%) across ACE-Curator stages. Standard: 661 samples; Hard: 839 samples.

Setting	Standard	Hard	Overall
Direct	62.78	45.05	52.87
Initial region	91.23	92.49	91.93
Corrected region	98.79	95.59	97.00
+ Quality filtering (complete)	**99.54**	**97.42**	**98.36**

Note: The first three rows use all 1,500 inputs; the complete pipeline uses the 1,467 retained annotations. The automatic retention rate is 97.80%, with 33 annotations referred for manual review.

Region guidance improves annotation quality, especially on Hard samples, and visual correction further improves both difficulty groups (Table 2). Quality filtering raises accuracy while retaining most annotations and concentrates manual review on a small set of risky samples. Appendices B.3 and B.4 provide additional comparisons of visual evidence and filtering components.

4.3 ACE action explanation performance and design analysis

We first compare single-image explanation performance across models. To understand these results, we then examine the role of ActionScope through supervision ablations and bounding box interventions, before testing whether additional screenshots or external region cues provide further benefits. All experiments in this section draw on the same evaluation set of 1,666 samples.

Single-image action explanation. Figure 4 compares joint semantic accuracy, overall and by action type, for general VLMs (Anthropic, 2026; MiniMax, 2026; Moonshot AI, 2026; OpenAI, 2026; Qwen Team, 2026c), ACE and its original backbones, and AutoGUI (Li et al., 2025) and Ferret-UI (You et al., 2025) with native prompting interfaces. Qwen3.8-2.4T denotes the Qwen3.8-Max API. All models use one screenshot without external region boxes; Appendix C gives the settings.

General VLMs are more accurate on typing and swiping than on point actions (Figure 4). This difference may arise because typing and swiping can often be interpreted from screen content and context, whereas point actions require precise alignment between numerical coordinates and UI

elements. AutoGUI and Ferret-UI also perform poorly on point actions despite having been trained to describe elements specified by points. This suggests that existing element description capabilities do not directly meet the requirements of explaining recorded actions. In contrast, ACE-1.3B and ACE-8B show their main advantage on point actions, suggesting that task-specific training helps models identify the target from action coordinates and generate a corresponding explanation.

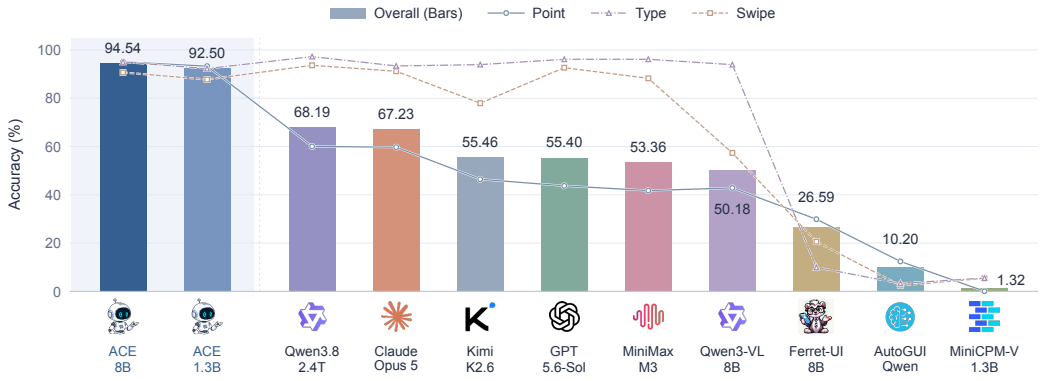


Figure 4: Joint semantic accuracy (%) for single-image action explanation. Full results are reported in Table C3.

ActionScope supervision and use of region information. Table 3(a) compares single-image ACE-1.3B and ACE-8B with four versus five output fields. For each backbone, we train new rank-128 adapters independently for both versions from the same GUI-adapted backbone. Samples, semantic labels, and training settings are identical; the four-field version omits only the bounding box field and its supervision. Before target and purpose generation, we set the response's bounding box field to a manually annotated box for another UI element on the same screen, then let the model continue generating. Keeping the original screenshot and action parameters, we evaluate whether the continuation still explains the original action correctly and compare its accuracy with original single-image generation (Table 3(b)). Appendix C.3 details the prefix source and implementation.

Table 3: ActionScope supervision and bounding box intervention. Both experiments report joint semantic accuracy (%). Small brackets show changes in percentage points relative to four-field models in (a) and Original generation in (b).

(a) Models trained with four versus five output fields.

Model	Fields	Overall	Point	Type	Swipe	Standard	Hard
ACE-1.3B	4	88.06	88.06	89.50	86.76	92.64	82.35
	5	92.50 **[+4.44]**	93.29 **[+5.23]**	92.27 [+2.77]	87.75 [+0.99]	96.10 [+3.46]	88.01 **[+5.66]**
ACE-8B	4	91.60	91.73	94.48	88.24	96.10	85.98
	5	94.54 **[+2.94]**	95.08 **[+3.35]**	95.03 [+0.55]	90.69 [+2.45]	97.29 [+1.19]	91.11 **[+5.13]**

(b) Generation with the model's predicted box (Original) or an incorrect box (Wrong).

Action	ACE-1.3B		ACE-8B	
	Original	Wrong	Original	Wrong
Point	93.29	60.97 **[-32.32]**	95.08	57.46 **[-37.62]**
Type	92.27	73.48 **[-18.79]**	95.03	39.78 **[-55.25]**
Swipe	87.75	82.35 [-5.40]	90.69	84.80 [-5.89]
Overall	92.50	64.95 **[-27.55]**	94.54	58.88 **[-35.66]**

ActionScope supervision improves joint semantic accuracy at both model scales, with larger gains on point actions and Hard samples (Table 3(a)). Accuracy with an incorrect box is markedly lower than with original generation, especially for point actions and typing (Table 3(b)), indicating that

subsequent descriptions are sensitive to the bounding box field. The smaller gap for swipes may reflect the relatively fixed explanation patterns of routine operations such as page scrolling, which depend less on the specific target identity.

Single-image, dual-image, and region-assisted input. Figure 5(a) assesses the value of pre- and post-action screenshots; Table C4 reports detailed results by action type. Both five-field ACE-8B variants are trained from the same GUI-adapted backbone with separate rank-128 adapters and identical settings. The dual-image variant adds post-action screenshots for point actions and swipes and pre-action screenshots for typing, with each pair in chronological order. Figure 5(b) compares Qwen3.8-2.4T's joint semantic accuracy on point actions with ordinary dual-image input and with SAM2 (Ravi et al., 2025) or OmniParser (Lu et al., 2024) assistance, using single-image ACE-1.3B and ACE-8B as references.



(a) Single-Image vs. Dual-Image Input

(b) External Region Cues

Figure 5: Joint semantic accuracy (%) with different visual inputs. (a) Overall accuracy. (b) Accuracy on 1,281 point actions.

Dual-image input improves general VLMs' overall accuracy (Figure 5(a)), indicating that pre- and post-action state changes add evidence for action explanation. In contrast, ACE-8B achieves similar overall accuracy with single- and dual-image input; its single-image accuracy exceeds that of dual-image Qwen3.8-2.4T and Claude Opus 5. This suggests that, after task-specific training, a single screenshot provides sufficient visual evidence for high-quality action explanations. Figure 5(b) evaluates only point actions because region-assisted configurations locate the target region using the action point. OmniParser assistance improves accuracy, whereas SAM2 does not. Even with dual-image input and external region cues, Qwen3.8-2.4T remains less accurate on point actions than single-image ACE-1.3B and ACE-8B. Appendix C.5 illustrates target misidentification and limitations of auxiliary visual evidence.

4.4 Utility of the annotations for downstream training

Both experiments use ACE-8B with one screenshot to generate annotations, retain the original screenshots and action labels, and reconstruct only the textual supervision. Appendix D details data construction, training, and scoring.

Offline GUI navigation. ACE generates screen and action descriptions for each of the 24,352 steps in 1,500 trajectories from GUIOdyssey (Lu et al., 2025). A text-based large language model (LLM) converts the action target and purpose into a low-level instruction (LLI) and infers the high-level task instruction from the descriptions across steps. Group A retains the original task instructions and LLIs, Group B reconstructs only the LLIs, and Group C reconstructs both. We fine-tune Qwen3-VL-2B separately on each set under the same settings.

We evaluate offline using fixed screenshots from 500 randomly sampled trajectories in AndroidControl High (Li et al., 2024), containing 2,304 evaluable steps (Table 4). With reference LLI history, each step is predicted independently using dataset-provided LLIs from previous steps. Evaluation with generated LLI history starts with an empty history, accumulates LLIs produced by the evaluated model, and stops at the first error or invalid output. With reference history, step accuracy is the percentage of evaluated steps whose predicted action type and applicable arguments are correct under the evaluation protocol in Appendix D.1.

Table 4: Offline GUI navigation on AndroidControl High with different textual supervision.

Setting	Reference history					Generated history	
	Point	Type	Swipe	Back	Overall	Correct steps	Traj. acc.
Backbone	14.57	79.44	8.31	1.05	17.97	59	1.60
A	53.02	87.22	22.49	4.21	48.26	411	15.60
B	57.35	82.78	35.45	8.42	**53.43**	**501**	**18.40**
C	54.88	82.22	25.43	37.89	51.09	484	17.80

Note: Correct steps sums correct predictions before the first failure across trajectories. Traj. acc. is the percentage of 500 trajectories with all evaluated actions correct.

Group B outperforms Group A overall under both history settings (Table 4), indicating that step semantics reconstructed from ACE descriptions improve both independent step prediction and consecutive action prediction. Group B's accuracy gains over Group A are largest for swipe and point actions. Group C performs worse overall than Group B but still outperforms Group A, indicating that reconstructed task instructions and LLIs also provide useful training supervision. Reconstructed tasks emphasize explicit operation sequences, whereas AndroidControl High includes abstract goals that require planning; this difference may favor retaining the original task instructions in Group B.

GUI grounding. Using approximately 50K samples from AutoGUI (Li et al., 2025), Widget Captioning (Li et al., 2020), and AMEX (Chai et al., 2025), we compare original grounding instructions with ACE-reconstructed instructions, generated by a text LLM from ACE's action target and purpose descriptions. Both groups keep images and target points fixed and fine-tune all language modules of the original general VLM SmolVLM2-2.2B (Marafioti et al., 2025) under identical settings.

We evaluate on ScreenSpot-v2 (mobile) (Wu et al., 2025), MoTIF (click subset) (Burns et al., 2022), AndroidControl (500-click subset) (Li et al., 2024), MMBench-GUI (L2, mobile) (Wang et al., 2026), and UI-I2E-Bench (mobile) (Liu et al., 2025b). Given a screenshot and an instruction, the model predicts a normalized point, which is counted as correct if it lies within the reference bounding box.

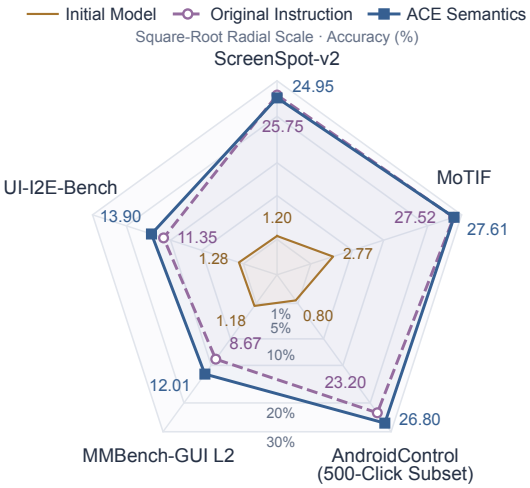


Figure 6: GUI grounding accuracy (%) of SmolVLM2-2.2B.

Training with ACE-reconstructed instructions yields higher grounding accuracy than training with the original instructions on several benchmarks (Figure 6). These results further support ACE's ability to interpret and accurately describe GUI actions. Such descriptions could help explain users' operations and analyze their activities and behavioral patterns.

5 Conclusion and Future Work

ACE jointly learns target localization and action explanation from a single screenshot and action evidence through ActionScope supervision. We develop the ACE-Curator annotation pipeline and supporting data resources. Experiments show that ActionScope supervision improves joint semantic accuracy and that bounding box interventions affect subsequent descriptions. ACE-reconstructed instructions also provide more effective supervision for offline GUI navigation and GUI grounding. Future work could integrate ACE into interaction logging to expand training data with quality-filtered action descriptions. With user authorization and privacy safeguards, continuous on-device action explanation could provide long-term behavioral records and recent activity context for personalization and proactive assistance.

AI USE STATEMENT

We used AI tools for grammar checking, sentence revision, academic language polishing, literature discovery, LaTeX formatting, and organizing the appendix. AI also assisted with dataset annotation and construction, with quality control and human review of selected samples as described in the paper. All AI-assisted manuscript content was reviewed by the authors, who take full responsibility for the final manuscript and associated research artifacts.

REPRODUCIBILITY STATEMENT

The methods and appendix describe the data construction process, training configurations, and evaluation protocols. The project is publicly available in an anonymized repository at `https://anonymous.4open.science/r/GUI-ACE-59D3/`. Evaluation results and judge outputs for all evaluated models are available as web pages in the repository to support independent verification of the reported results. All accounts associated with the public project use dedicated email addresses and contain no author-identifying information.

ETHICS STATEMENT

This study uses GUI interaction data from publicly available datasets and does not collect new interaction records from users. Dataset sources and processing procedures are documented in the paper and appendix.

REFERENCES

Anthropic. Introducing Claude Opus 5. Official product announcement, July 2026. URL `https://www.anthropic.com/news/claude-opus-5`. Accessed: 2026-09-22.

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.

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A. Plummer. A Dataset for Interactive Vision-Language Navigation with Unknown Command Feasibility. In *Computer Vision – ECCV 2022*, pp. 312–328, 2022.

Yue Cao, Yingyao Wang, Pi Bu, Jingxuan Xing, Wei Jiang, Zekun Zhu, Junpeng Ma, Sashuai Zhou, Tong Lu, Jun Song, Yu Cheng, Yuning Jiang, and Bo Zheng. AndroidLens: Long-Latency Evaluation with Nested Sub-Targets for Android GUI Agents. In *Conference on Computer Vision and Pattern Recognition 2026*, 2026. URL `https://openreview.net/forum?id=YLh2zZtUBR`.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. AMEX: Android Multi-annotation Expo Dataset for Mobile GUI Agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2138–2156, 2025.

Tongbo Chen, Zhengxi Lu, Zhan Xu, Guocheng Shao, Shaohan Zhao, Fei Tang, Yong Du, Kaitao Song, Yizhou Liu, Yuchen Yan, Wenqi Zhang, Xu Tan, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. KnowU-Bench: Towards Interactive, Proactive, and Personalized Mobile Agent Evaluation, 2026. URL `https://arxiv.org/abs/2604.08455`.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, 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, 2024.

Kanzhi Cheng, Zehao Li, Zheng Ma, Nuo Chen, Jialin Cao, Qiushi Sun, Zichen Ding, Fangzhi Xu, Hang Yan, Jiajun Chen, Anh Tuan Luu, Jianbing Zhang, Lewei Lu, and Dahua Lin. OpenMobile: Building Open Mobile Agents with Task and Trajectory Synthesis. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=jjGBHR7TKx.

Danielle Cohen, Yoni Halpern, Noam Kahlon, Joel Oren, Omri Berkovitch, Sapir Caduri, Ido Dagan, and Anatoly Efros. Small Models, Big Results: Achieving Superior Intent Extraction through Decomposition. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18780–18799, 2025.

DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence, 2026.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the Digital World as Humans Do: Universal Visual Grounding for GUI Agents. In *International Conference on Learning Representations*, volume 2025, pp. 30851–30883, 2025.

Jihao Gu, Qihang Ai, Yingyao Wang, Pi Bu, Jingxuan Xing, Yue Cao, Zekun Zhu, Wei Jiang, Ziming Wang, Yingxiu Zhao, Ming-Liang Zhang, Jun Song, Yuning Jiang, and Bo Zheng. Mobile-R1: Towards Interactive Capability for VLM-Based Mobile Agent via Systematic Training. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30802–30820, 2026.

Yu-Chung Hsiao, Fedir Zubach, Gilles Baechler, Srinivas Sunkara, Victor Carbune, Jason Lin, Maria Wang, Yun Zhu, and Jindong Chen. ScreenQA: Large-Scale Question-Answer Pairs Over Mobile App Screenshots. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9427–9452, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022.

Gang Li and Yang Li. Spotlight: Mobile UI Understanding using Vision-Language Models with a Focus. In *The Eleventh International Conference on Learning Representations*, 2023.

Hongxin Li, Jingfan Chen, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. AutoGUI: Scaling GUI Grounding with Automatic Functionality Annotations from LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10323–10358, 2025.

Hongxin Li, Yuntao Chen, and Zhaoxiang Zhang. GoClick: Lightweight Element Grounding Model for Autonomous GUI Interaction, 2026a. URL https://arxiv.org/abs/2604.23941.

Hongxin Li, Xiping Wang, Jingran Su, Zheng Ju, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. AutoGUI-v2: A Comprehensive Multi-Modal GUI Functionality Understanding Benchmark, 2026b. URL https://arxiv.org/abs/2604.24441.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the Effects of Data Scale on UI Control Agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 92130–92154, 2024.

Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget Captioning: Generating Natural Language Description for Mobile User Interface Elements. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5495–5510, 2020.

Guangyi Liu, Pengxiang Zhao, Liang Liu, Zhiming Chen, Yuxiang Chai, Yaozhen Liang, WenHao Wang, Siheng Chen, Zhengxi Lu, Shuai Ren, Hao Wang, Shibo He, Yong Liu, and Wenchao Meng. LearnAct: Few-Shot Mobile GUI Agent with a Unified Demonstration Benchmark. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 29820–29843, 2026.

Tao Liu, Chongyu Wang, Rongjie Li, Yingchen Yu, Xuming He, and Song Bai. GUI-Rise: Structured Reasoning and History Summarization for GUI Navigation. 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. 165132–165156. Curran Associates, Inc., 2025a. doi: 10.52202/085713-5505. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f144ab9985c739a5091ec188a2688644-Paper-Conference.pdf.

Xinyi Liu, Xiaoyi Zhang, Ziyun Zhang, and Yan Lu. UI-E2I-Synth: Advancing GUI Grounding with Large-Scale Instruction Synthesis. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15668–15684, 2025b.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. GUIOdyssey: A Comprehensive Dataset for Cross-App GUI Navigation on Mobile Devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22404–22414, 2025.

Taoran Lu, Minghao Liu, Musen Lin, Lichen Yuan, Yuhao Chao, Yiwei Liu, Haonan Xu, Rui Pan, Lei Yang, Yu Miao, and Zhaojian Li. GUI-ReWalk: Massive Data Generation for GUI Agent via Stochastic Exploration and Intent-Aware Reasoning. In *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence, IJCAI-26*, pp. 235–243, 2026. Main Track.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. OmniParser for Pure Vision Based GUI Agent, 2024. URL https://arxiv.org/abs/2408.00203.

Yibo Lyu, Gongwei Chen, Rui Shao, Weili Guan, and Liqiang Nie. PersonalAlign: Hierarchical Implicit Intent Alignment for Personalized GUI Agent with Long-Term User-Centric Records. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36074–36089, 2026.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. SmolVLM: Redefining small and efficient multimodal models. In *Second Conference on Language Modeling*, 2025.

MiniMax. MiniMax M3: Frontier Coding, 1M Context, Native Multimodality—All in One Model. Official model release announcement, June 2026. URL https://www.minimax.io/blog/minimax-m3. Accessed: 2026-09-22.

Moonshot AI. Kimi-K2.6. Hugging Face model card, 2026. URL https://huggingface.co/moonshotai/Kimi-K2.6. Accessed: 2026-09-22.

OpenAI. GPT-5.6 Sol Model. OpenAI API documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol. Accessed: 2026-09-22.

OpenBMB. MiniCPM-V 4.6. Hugging Face model card, 2026. URL https://huggingface.co/openbmb/MiniCPM-V-4.6. Accessed: 2026-09-22.

Qwen Team. Qwen3.5: Towards Native Multimodal Agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.7-Plus. Alibaba Cloud Model Studio documentation, 2026b. URL https://help.aliyun.com/en/model-studio/qwen3-7-plus. Accessed: 2026-09-25.

Qwen Team. Qwen3.8-Max: A New Bar for Coding and Cowork, August 2026c. URL https://qwen.ai/blog?id=qwen3.8.

Qwen Team. Qwen3.8-Flash. Alibaba Cloud Model Studio documentation, 2026d. URL https://help.aliyun.com/en/model-studio/qwen3-8-flash. Accessed: 2026-09-25.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. OS-Genesis: Automating GUI Agent Trajectory Construction via Reverse Task Synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5555–5579, 2025.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2Words: Automatic Mobile UI Summarization with Multimodal Learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pp. 498–510, 2021.

WenHao Wang, Mengying Yuan, Zijie Yu, Guangyi Liu, Rui Ye, Tian Jin, Siheng Chen, and Yanfeng Wang. MobileA3gent: Training Mobile GUI Agents Using Decentralized Self-Sourced Data from Diverse Users. In *Proceedings of the Fourth Workshop on Bridging Human-Computer Interaction and Natural Language Processing (HCI+NLP)*, pp. 79–112, 2025a.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, Weiyun Wang, Xiangyu Zhao, Jixuan Chen, Haodong Duan, Tianbao Xie, Chenyu Yang, Shiqian Su, Yue Yu, Yanting Zhang, Xiangyu Yue, Weijie Su, Xizhou Zhu, Wei Shen, Jifeng Dai, and Wenhai Wang. MMBench-GUI: A Unified Hierarchical Evaluation Framework for Multi-Platform GUI Agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6239–6248, 2026.

Ziwei Wang, Weizhi Chen, Leyang Yang, Sheng Zhou, Shengchu Zhao, Hanbei Zhan, Jiongchao Jin, Liangcheng Li, Zirui Shao, and Jiajun Bu. MP-GUI: Modality Perception with MLLMs for GUI Understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29711–29721, 2025b.

Luis Wiedmann, Orr Zohar, Amir Mahla, Xiaohan Wang, Rui Li, Thibaud Frere, Leandro von Werra, Aritra Roy Gosthipaty, and Andrés Marafioti. FineVision: Open Data Is All You Need, 2025. URL https://arxiv.org/abs/2510.17269.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. OS-ATLAS: Foundation Action Model for Generalist GUI Agents. In *International Conference on Learning Representations*, volume 2025, pp. 5090–5108, 2025.

Yiping Xie, Song Chen, Jingxuan Xing, Wei Jiang, Zekun Zhu, Yingyao Wang, Pi Bu, Jun Song, Yuning Jiang, and Bo Zheng. SecAgent: Efficient Mobile GUI Agent with Semantic Context. *arXiv preprint arXiv:2603.08533*, 2026.

Qinglong Yang, Haoming Li, Haotian Zhao, Xiaokai Yan, Jingtao Ding, Fengli Xu, and Yong Li. FingerTip 20K: A Benchmark for Proactive and Personalized Mobile LLM Agents. In *International Conference on Learning Representations*, volume 2026, pp. 85009–85029, 2026a.

Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baoling Peng, Huan Zhang, Jianfeng Gao, and Tong Zhang. GUI-Libra: Training Native GUI Agents to Reason and Act with Action-aware Supervision and Partially Verifiable RL, 2026b. URL https://arxiv.org/abs/2602.22190.

Saelyne Yang, Jaesang Yu, Yi-Hao Peng, Kevin Qinghong Lin, Jae Won Cho, Yale Song, and Juho Kim. GUIDE: A Benchmark for Understanding and Assisting Users in Open-Ended GUI Tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13017–13027, 2026c.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-UI: Grounded Mobile UI Understanding with Multimodal LLMs. In *Computer Vision – ECCV 2024*, pp. 240–255, 2025.

Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the Zoo: Chain-of-Action-Thought for GUI Agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12016–12031, 2024.

Le Zhang, Yixiong Xiao, Xinjiang Lu, Jingjia Cao, Yusai Zhao, Jingbo Zhou, Lang An, Zikan Feng, Wanxiang Sha, Yu Shi, et al. OmegaUse: Building a General-Purpose GUI Agent for Autonomous Task Execution. *arXiv preprint arXiv:2601.20380*, 2026.

Zhongyin Zhao, Yuan Liu, Yikun Liu, Haicheng Wang, Le Tian, Xiao Zhou, Yangxiu You, Zilin Yu, Yang Yu, and Jie Zhou. POINTS-GUI-G: GUI-Grounding Journey. *arXiv preprint arXiv:2602.06391*, 2026.

A Data and evaluation criteria

A.1 Data sources, coverage, and evaluation splits

Both GUI domain adaptation and action explanation training exclude samples overlapping with the evaluation candidate actions. Appendix C.1 gives the training configuration.

GUI domain adaptation. This stage uses 525,653 deduplicated samples covering widget descriptions, grounding, screen descriptions, and screen question answering. Table A1 lists the direct sources. We use the ScreenQA (Hsiao et al., 2025) and Screen2Words (Wang et al., 2021) subsets of FineVision and the samples corresponding to AMEX (Chai et al., 2025) images in MP-GUI.

Table A1: Sources of GUI domain adaptation data.

Source	Samples
GoClick (Li et al., 2026a)	296,601
FineVision (Wiedmann et al., 2025) (ScreenQA/Screen2Words)	84,291
CMGUI (Xie et al., 2026)	79,603
MP-GUI (Wang et al., 2025b) (samples corresponding to AMEX images)	65,158
Total	525,653

Action explanation data pool. Deduplication and final manifest selection from 303,046 candidate records annotated by ACE-Curator yield 199,456 unique action events and 301,456 language-specific annotation records, with 4,791 actions reviewed and corrected by humans. Chinese and English responses for the same action count as separate records. A model translates responses with the screenshot as context, preserving UI text in its original language to keep target names consistent. Figure A1 summarizes the sources, application coverage, and action types before evaluation splitting. Application counts merge aliases and correct misidentified applications; the pool covers 416 applications, which may overlap across sources. CMGUI contributes 39.9% of the pool, and Point actions account for 84.2%.

The sources are CMGUI (Xie et al., 2026), GUI-Libra (Yang et al., 2026b), LearnGUI (Liu et al., 2026), OpenMobile (Cheng et al., 2026), AitZ (Zhang et al., 2024), AndroidLens (Cao et al., 2026), OS-Genesis (Sun et al., 2025), Mobile-R1 (Gu et al., 2026), and OS-Nav (Zhang et al., 2026).

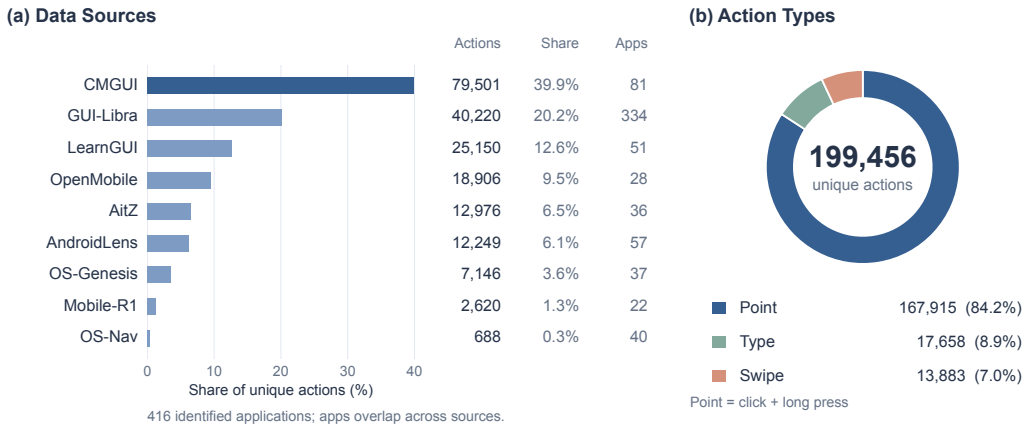


Figure A1: Data pool composition by source and action type, counting each of the 199,456 actions once.

Action explanation training set. After holding out 2,695 evaluation actions and all their language versions, the training set contains 196,761 unique action events and 298,503 records: 189,892 in English and 108,611 in Chinese. Both ACE models use the same training manifest. Table A2 distinguishes unique action counts from training records that include both language versions.

Table A2: Action distribution in the action explanation training set.

Action	Unique actions	Annotation records
Point	165,685	259,033
Swipe	13,640	22,034
Type	17,436	17,436
Total	196,761	298,503

Point includes 164,943 clicks with 258,135 records and 742 long presses with 898 records.

Evaluation sets. We select 2,695 candidate actions with a nearly 1:1 ratio of Standard to Hard samples to construct the annotation analysis and model evaluation sets described in main-text Section 4.1. The sets share 471 actions and jointly cover all candidates. Reference annotations are verified and corrected by humans.

Table A3: Evaluation set sizes and composition.

Set	Unique actions	Standard	Hard	English	Chinese
Evaluation candidates	2695	1375	1320	-	-
Annotation analysis	1500	661	839	928	572
Model evaluation	1666	924	742	995	671

The annotation analysis set contains 1,331 Point, 84 Type, and 85 Swipe samples; the model evaluation set contains 1,281, 181, and 204, respectively. Clicks and long presses are combined as Point, with accuracy computed from their pooled correct counts. The two sets cover at least 110 and 221 identified applications, respectively.

Difficulty assignment. Annotators assign difficulty while selecting the reference annotations, using the following criteria:

1. Spatial identification: small, low-contrast, occluded, or cropped targets, or localization among densely arranged controls.
2. Recipient extent: distinguishing a parent container from an independent child control, such as a card and a button within it.
3. Entity correspondence: distinguishing repeated targets through their row, column, time, price, or other attributes.
4. UI layers: identifying the recipient among dialogs, overlays, masks, and background content.
5. Local swipes: identifying picker wheels, sliders, carousels, and nested scrolling regions.
6. Text entry and state: competing input fields, unclear focus, or subtle state differences that change an action's meaning.
7. Functional cues: interpreting similar graphics or functions when explicit labels are absent.

Each factor is rated 0, 1, or 2 for no, mild, or clear difficulty, taking the highest rating within each category. A sample is Hard if any category scores 2 or at least two categories score 1; all others are Standard. Annotators ground these ratings in the visible cues of each selected sample.

A.2 Semantic correctness

Annotation and model evaluations use the same criteria. Point and Swipe must pass independent target verification and joint semantic assessment; Type undergoes joint semantic assessment only. Screen descriptions and bounding box quality are excluded from semantic scores.

Independent target verification. A text model compares `action_ui_target` with the reference target. Equivalent names pass, descriptions that clearly identify another object fail, and

uncertain cases proceed to visual review. The visual model receives the action screenshot, normalized action parameters, the reference target and purpose, and the predicted target. A hollow point marker or swipe trajectory indicates the action location. The predicted purpose, history, post-action screenshot, and generated box are withheld to isolate the target field. For Swipe, the target is the receiving container.

Joint semantic assessment. The target and purpose are compared jointly with the reference, allowing equivalent wording and complementary information across fields. Semantically equivalent responses pass; definite contradictions in function, state, direction, or entered text fail. A visual model reviews the remaining cases using the screenshot, action, previous-step history, and both predicted and reference descriptions, with a post-action screenshot when needed. Type uses the screenshot after text entry.

Both text checks use DeepSeek-v4-flash-0731 (DeepSeek-AI, 2026). For both model and annotation evaluations, visual target verification uses Qwen3.5-122B-A10B, and visual joint semantic assessment uses Qwen3.7-Plus (Qwen Team, 2026b). Thinking mode is disabled throughout. Appendix F.6 provides prompt excerpts.

Joint semantic accuracy divides the number of correct samples by the evaluation set size: 1,666 for model evaluation and 1,500 for annotation experiments. Generation failures count as incorrect. After quality filtering, accuracy is computed over automatically retained annotations, while coverage uses the original 1,500 samples as its denominator (Appendix B.4).

Figure A2 summarizes the check order and the text-first decision pattern used separately by each check. The evidence supplied to visual review depends on whether the target alone or the joint target–purpose semantics are being assessed.

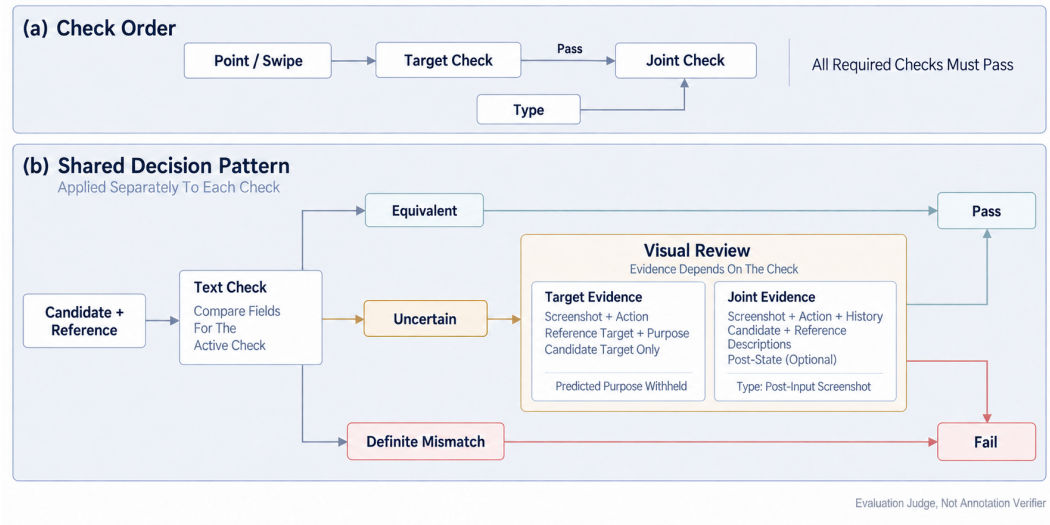


Figure A2: Semantic evaluation. Each check uses text comparison followed by visual review of uncertain cases. Type uses joint assessment only.

A.3 Agreement between the judge and human assessment

We draw a stratified random sample of 100 actions from the 1,666-sample evaluation set, stratifying by action type and difficulty. Responses from three models yield 300 responses across these 100 action scenarios. Humans assess each response using the screenshot, action, and context, and we compare their decisions with the evaluation judge. Table A4 reports 100 responses per model.

The judge agrees with humans on 298 responses (99.33%). Both disagreements are false accepts: one confuses neighboring activities, and the other assigns a favorite button to the wrong item. No correct response is rejected.

Table A4: Agreement between the judge and human assessment.

Model	Agreement (%)	False accepts	False rejects
ACE-8B, single image	100.00	0	0
Qwen3.8-2.4T, single image	99.00	1	0
MiniMax M3	99.00	1	0
Total	99.33	2	0

B Annotation design and quality control

B.1 Region annotation and visual inputs

Localization and semantic annotation use Qwen3.5-35B-A3B, reverse grounding uses POINTS-GUI-G, and risk verification uses Qwen3.5-122B-A10B. All API calls use temperature=0 with thinking mode disabled.

Region correction. Stage 1 proposes an initial region. Stage 2 displays its blue box on the main screenshot and checks both the recipient and its full extent. Both stages show hollow locator arms for Point. For Type and Swipe, correction receives views of the same screenshot with and without the initial box. Type localization uses the screenshot after text entry, the actual entered text, and previous-step history.

In layered interfaces, the region follows the object the action actually reaches. The recipient may be a foreground control, background outside a dialog that dismisses it, or a control beneath a non-blocking overlay. After retries, a structurally valid initial box proceeds to Stage 2 even if it excludes the action point. If no valid initial box can be generated, or the final box fails validation, the sample goes to manual review; the pipeline does not fall back to the initial box.

Separate screen and action annotation. Table B1 lists the visual inputs for both passes.

Table B1: Visual inputs for the screen and action passes.

Action	Screen pass	Action pass
Point	Full pre-action screenshot	Full image with final region marked + region crop
Type	Full screenshot after text entry	Full image with final input region marked
Swipe	Full pre-action screenshot	Full image with final receiving container marked + region crop

The screen pass uses an unmarked full screenshot, and the action pass marks the region on the same image. Crops are enlarged and retain the region boundary. For Point and Swipe, the action pass may also receive a reliable post-action screenshot. Numerical point or start/end coordinates are provided without drawing point markers or trajectories. Type uses neither crops nor pre-action screenshots.

Both passes receive the action and previous-step history, and the action pass also receives region coordinates.

The two outputs and the Stage 2 box form the five-field annotation. Unresolved invalid or marker-contaminated annotations go to manual review. For Type, the purpose field is populated programmatically with the original logged text, preserving every character. Swipe direction is computed from the start and end points and expressed consistently. Logged text is used to construct labels but is withheld from ACE during evaluation (Appendix C.2). Appendices F.2 and F.3 provide the localization and semantic annotation prompts.

B.2 Risk-driven quality filtering

Rule-based validation. We check output completeness and validity, including a valid box containing the action point for Point or the start point for Swipe. A swipe endpoint may fall outside the region. Any failure triggers manual review.

Semantic risk. For Point samples passing rule-based validation, we concatenate the target and purpose and provide them to POINTS-GUI-G with the unmarked main screenshot. Consistency is determined by whether the predicted point lies within the final ActionScope. Swipe risk is determined from the localization process and container type. Samples meeting any applicable condition in Table B2 are checked by the annotation verifier (Verifier); the rest are retained automatically. Appendix F.4 provides the reverse grounding prompt.

Table B2: Conditions that trigger the Verifier after rule-based validation.

Action	Risk source	Trigger
Point	Reverse grounding	The predicted point lies outside the box, or no valid grounding result is obtained.
Swipe	Unstable localization	Localization requires retries or lacks a valid stage result.
Swipe	Region change	The region box, container type, or recipient category changes between stages.
Swipe	Uncertain judgment	The judgment or recipient category is uncertain, or confidence is low (below 0.8 or explicitly flagged).
Swipe	Recipient ambiguity	The recipient is an ambiguity-prone local control, such as a picker or slider, or an unclassified swipe region.

Region changes are checked without IoU. Ordinary page or list scrolling is retained automatically when localization succeeds on the first attempt, the region is clear, and the two stages agree. A large region or a foreground/background relationship alone does not trigger risk.

Two-pass verification. The Verifier first identifies the recipient and its function from the main screenshot with action markers, action parameters, and previous-step history, consulting the post-action screenshot as needed. It then uses this assessment, the candidate annotation, the full screenshot, and the region crop to decide whether to retain the annotation. Passing samples retain their original labels. Failed cases and cases without a valid judgment after retries go to manual review. Appendix F.4 gives the prompts.

B.3 Localization and visual evidence ablations

We separate the effects of displaying the action location, identifying the functional region, and providing enlarged region detail. All settings use the same 1,500 annotation analysis samples and Qwen3.5-35B-A3B for localization and semantic annotation, with temperature=0 and thinking mode disabled. Scores follow Appendix A.2 before quality filtering, with generation failures counted as incorrect. Table B3 reports correct samples over all inputs. The setting IDs denote pipeline configurations.

Table B3: Localization and visual evidence ablations.

Setting	Correct/inputs	Acc. (%)
M0: numerical action parameters, direct five-field generation	793/1500	52.87
E1: visual action marker, no element region	1193/1500	79.53
M1: initial box, marked full image + crop	1379/1500	91.93
E2: corrected box, marked full image only	1331/1500	88.73
M2: corrected box, marked full image + crop	1455/1500	97.00

M0, M1, and M2 correspond to Direct, Initial region, and Corrected region in main-text Table 2. M0 uses no prior region localization. E1 adds an action location marker without an element box or crop. M1 uses the initial Stage 1 box, while E2 and M2 include Stage 2 correction; all three then generate screen and action annotations separately. E2 differs from M2 by omitting the region crop. Crops apply only to Point and Swipe, while Type always uses the boxed full screenshot after text entry.

Visualizing the action location improves on numerical coordinates alone, but remains less accurate than full region guidance. Comparing E2 and M2 further shows the benefit of enlarged region detail.

B.4 QUALITY FILTERING ABLATIONS

We assess reverse grounding's error detection and the Verifier's effect on manual review burden. Automatic accuracy is the number of correct retained annotations divided by all retained annotations. Coverage is the retained count divided by all 1,500 inputs; error recall is the number of erroneous samples sent for review divided by the original error count. Generation failures remain in the original denominator, and scores exclude subsequent human correction of referred samples.

Error detection by reverse grounding. Table B4 fixes M0's candidate regions and semantic outputs, varying only the retention conditions and sending all generation failures for review. M0's relatively high error rate lets us assess whether reverse grounding detects errors missed by the rules. Error recall uses 707 original errors as its denominator, including 1 generation failure.

Table B4: Error detection with region rules and reverse grounding.

Setting	Correct/ retained	Acc. (%)	Coverage (%)	Error recall (%)	Review (n)
G0: exclude generation failures only	793/1499	52.90	99.93	0.14	1
G1: region rules	758/883	85.84	58.87	82.32	617
G2: rules + reverse grounding	699/757	92.34	50.47	91.80	743

Reverse grounding increases error recall and automatic accuracy while reducing coverage. It catches errors missed by region rules but also refers some correct annotations, motivating further verification.

Verification and manual review burden. Figure B1 fixes M2's annotations and judge decisions, adding filtering conditions incrementally. The first three filtering settings directly refer flagged samples. The final setting sends risk cases that pass rule-based validation to the Verifier, retaining those that pass and referring the rest. Rule-based validation failures always go directly to manual review. All settings reuse existing localization, semantic generation, and verification results under Appendix B.2, without modifying the evaluated labels. The bars partition the same 1,500 annotations into correct retention, incorrect retention, and manual review.

The Verifier checks 115 risk samples, retains 93 (89 correct and 4 incorrect), and sends the remaining 22 for manual review. Compared with directly referring all risk samples, it reduces total referrals from 126 to 33, with 98.36% automatic accuracy and 97.80% coverage.

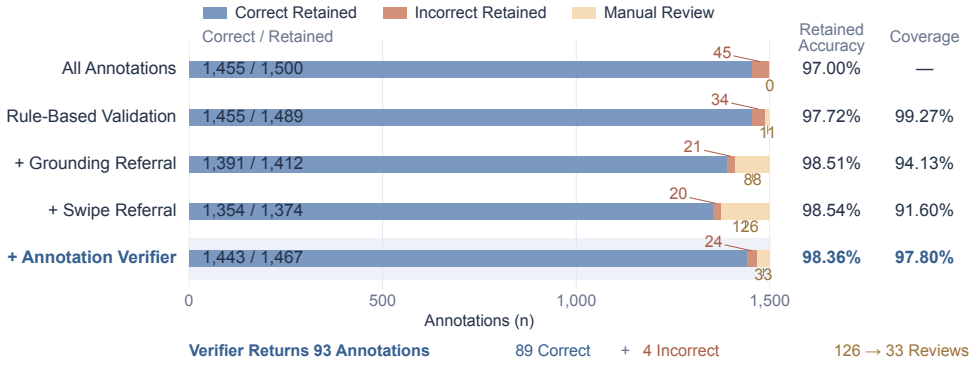


Figure B1: Quality filtering of 1,500 annotations. Bars partition samples into correct retention, incorrect retention, and manual review.

B.5 ANNOTATION WITH SMALLER MODELS

We test whether smaller models can produce annotations of comparable quality to larger models. All three settings use the same 1,500 samples and change the localization and semantic models together.

The reverse grounding model and Verifier remain POINTS-GUI-G and Qwen3.5-122B-A10B. Each setting verifies Point samples with grounding inconsistencies and all valid Swipe samples. Appendix B.4 verifies swipes selectively. In Table B5, 35B, 122B, and 2.4T denote Qwen3.5-35B-A3B, Qwen3.5-122B-A10B, and the Qwen3.8-Max API, respectively.

Table B5: Annotation quality with smaller and larger models.

Localization/semantic model	Correct/retained	Acc. (%)	Coverage (%)	Error recall (%)	Review (n)
35B / 35B	1437/1461	98.36	97.40	46.67	39
122B / 122B	1442/1479	97.50	98.60	27.45	21
2.4T / 2.4T	1458/1487	98.05	99.13	19.44	13

Using the 35B model for localization and semantic annotation achieves 98.36% automatic accuracy at 97.40% coverage, comparable in annotation quality to the settings using larger models. This supports using a smaller model for bulk annotation and concentrating calls to larger models on risk verification. The three settings contain 45, 51, and 36 errors before filtering, respectively; each error recall uses its own original error count as the denominator.

C ACE TRAINING AND MODEL ANALYSIS

This appendix details training, evaluation inputs, and continuation from an incorrect bounding box, with additional results on Standard and Hard samples.

C.1 MODELS AND TRAINING CONFIGURATION

ACE-1.3B uses MiniCPM-V-4.6-1.3B with its 16x visual token compression, and ACE-8B uses Qwen3-VL-8B-Instruct. Both retain their original visual modules.

For each model, we merge the adapter weights after adaptation on public GUI data, then initialize a new adapter for fine-tuning on the shared action explanation training set. Appendix A.1 describes the data, and Table C1 gives both stages' configurations.

Both stages use AdamW, LoRA dropout of 0.05, and 3% warmup, with the remaining settings in Table C1. We merge the final adapter from each stage without selecting checkpoints by test performance. Weight decay is 0.01 during action explanation fine-tuning.

Table C1: ACE training configuration. The ACE columns specify action explanation fine-tuning. Both stages use cosine learning rate decay.

Setting	GUI adaptation	ACE-1.3B	ACE-8B
Training records	525,653	298,503	298,503
LoRA rank / alpha	32 / 64	128 / 256	128 / 256
Epochs	1	1	1
Effective batch size	32	48	48
Peak learning rate	5×10^{-5}	3×10^{-5}	2×10^{-5}
Minimum learning rate	1×10^{-5}	1×10^{-5}	5×10^{-6}

LoRA covers the attention and feed-forward linear modules in all language layers. The visual modules and original backbone parameters remain frozen. Training minimizes autoregressive cross-entropy on answer tokens only.

For each backbone, the four-field and five-field variants independently initialize rank-128 adapters from the same GUI-adapted backbone and use identical examples, semantic labels, and training settings. The five fields are generated in the order `current_page`, `action_relevant_screen_info`, `action_ui_bbox`, `action_ui_target`, and `action_purpose`; the four-field variant removes only the bounding box field and its supervision. Dual-image ACE-8B uses the

same training configuration and changes only the visual input, following the screenshot protocol in Appendix C.2.

C.2 Evaluation inputs and baseline adaptation

Each action is evaluated independently using the current action, its screenshots, and the dataset-provided screen and action descriptions from the immediately preceding step. This one-step textual history does not accumulate earlier model predictions.

- Single-image input uses the pre-action screenshot for Point (clicks and long presses) and Swipe, and the post-action screenshot for Type.
- Dual-image input adds the other screenshot, with both images ordered before then after the action.

For Type, the logged input string is withheld. The model must identify the receiving field and read the visible text from the post-action screenshot. Preceding history is provided as context when available. Ordinary single-image and dual-image configurations provide no external bounding boxes. Appendix F.5 gives the five-field generation prompts used by general VLMs in these configurations.

Qwen3.8-2.4T denotes the Qwen3.8-Max API, and Qwen3.5-122B denotes Qwen3.5-122B-A10B. The untuned backbones are MiniCPM-V-4.6-1.3B and Qwen3-VL-8B-Instruct. AutoGUI-Qwen-v0.1 and Ferret-UI Llama8B-anyres use their native question interfaces, adapted to Type and Swipe actions. Their free-text answers are evaluated for the semantics they express without requiring the five-field format. Untuned MiniCPM receives the generic five-field prompt, with generation and parsing failures counted as incorrect under the shared rules.

The external region cue comparison tests the explanation quality of a general VLM given parser-generated regions. Qwen3.8-2.4T receives boxes for Point actions from SAM2 or OmniParser using the configurations in Table C2.

Table C2: External region cue configurations.

Setting	**SAM2**	**OmniParser**
Version	SAM2 Hiera-L	OmniParser V2
Region generation	Generate masks with the action point as a positive prompt	Full-image detection and OCR
Box selection	Tight bounding box of the highest-scoring mask	Smallest candidate box containing the action point, without filtering by interactivity

If no valid box contains the action point, prediction falls back to ordinary dual-image input.

With a valid box, the model receives the full pre-action screenshot with a red box, an unmarked crop of that box, and the full post-action screenshot. The 181 Type and 204 Swipe predictions are reused from the ordinary dual-image configuration, so the direct region cue comparison concerns Point actions. Region cue configurations use a four-field region description prompt, whereas ordinary dual-image prediction uses a five-field prompt. The comparison therefore evaluates the combined input and prompt configuration.

C.3 Bounding box intervention

The intervention replaces only `action_ui_bbox` in the continuation prefix, preserving the screenshot, action coordinates, questions, and preceding history. The incorrect box is not drawn on the image. We then generate the target and purpose and score them against the original action.

We manually select a different visible receiver with its own semantics on the same screen. For Point, the incorrect box excludes the action point. For Type, we preferentially select another input field. For Swipe, we select a region with a different functional recipient, excluding subregions of the original scrolling container. Coordinates are normalized to 0–1000. Regions are distinguished by their functional recipients, without an IoU criterion.

Table C3: Joint semantic accuracy (%) for action explanation from a single screenshot.

Model	Overall	Point	Type	Swipe	Standard	Hard
Qwen3.8-2.4T	68.19	60.03	**97.24**	**93.63**	77.16	57.01
Claude Opus 5	67.23	59.72	93.37	91.18	75.32	57.14
GPT-5.6-Sol	55.40	43.72	96.13	92.65	64.72	43.80
MiniMax M3	53.36	41.76	96.13	88.24	61.15	43.67
Kimi K2.6	55.46	46.45	93.92	77.94	64.07	44.74
AutoGUI-Qwen	10.20	12.41	3.31	2.45	9.52	11.05
Ferret-UI-8B	26.59	29.90	9.94	20.59	28.35	24.39
MiniCPM-V-1.3B	1.32	0.08	5.52	5.39	1.73	0.81
Qwen3-VL-8B	50.18	42.86	93.92	57.35	58.98	39.22
ACE-1.3B	92.50	93.29	92.27	87.75	96.10	88.01
ACE-8B	**94.54**	**95.08**	95.03	90.69	**97.29**	**91.11**

Each model retains its own generated screen fields and continues from the substituted box, with no target or purpose answer supplied. Continuation uses no cached state containing the original box. Under the incorrect-box condition, ACE-1.3B and ACE-8B correctly explain 1,082 and 981 samples, respectively.

C.4 Results by visual evidence and difficulty

Table C4: Joint semantic accuracy (%) with different visual evidence.

(a) Single-image versus dual-image input.

Model	Images	Overall	Point	Type	Swipe	Standard	Hard
ACE-1.3B	Single	92.50	93.29	92.27	87.75	96.10	88.01
ACE-8B	Single	**94.54**	**95.08**	95.03	90.69	**97.29**	**91.11**
ACE-8B	Dual	94.30	93.60	97.24	**96.08**	96.86	**91.11**
Qwen3-VL-8B	Dual	72.81	68.85	95.03	77.94	78.57	65.63
Qwen3.8-2.4T	Dual	92.08	91.02	98.34	93.14	95.67	87.60
Claude Opus 5	Dual	93.52	92.43	**98.90**	95.59	96.21	90.16

(b) External region assistance for point actions; ACE uses single-image input.

Setting	Overall	Point	Standard	Hard
Dual	92.08	91.02	95.67	87.60
+ SAM2	91.78	90.63	94.91	87.87
+ OmniParser	93.16	92.43	96.43	89.08
ACE-1.3B	92.50	93.29	96.10	88.01
ACE-8B	**94.54**	**95.08**	**97.29**	**91.11**

Note: In (b), the first three settings use Qwen3.8-2.4T. Region assistance changes only the 1,281 Point predictions; Type and Swipe reuse the ordinary dual-image predictions. Appendix C.2 details the assisted inputs and prompts.

On the same Standard and Hard subsets, single-image ACE-8B outperforms the general VLMs with dual images or region cues in Table C4. Its single-image and dual-image variants have the same Hard accuracy. OmniParser improves Qwen's results on both subsets, although they remain below ACE-8B. SAM2 reduces Standard accuracy and slightly improves Hard accuracy.

C.5 Action explanation failure cases

Figures C1–C7 distinguish explanation errors from limitations of auxiliary evidence. Each figure contains two cases with full original screenshots. Gold rings mark recorded Point actions, blue boxes show available reference ActionScopes, and coral outlines illustrate model-described targets rather than predicted bounding boxes. Target strings are verbatim; purposes are condensed.

Failures without auxiliary regions. These four cases use single-image results from Qwen3.8-Max, Claude Opus 5, and GPT-5.6-Sol without auxiliary regions. Figure C1 shows Qwen3.8-Max naming a neighboring menu item and a highlighted date instead of the tapped receiver. Figure C2 shows Claude Opus 5 confusing the Plus and Eye icons, and GPT-5.6-Sol interpreting Discard as Keep Editing, reversing the action's meaning.

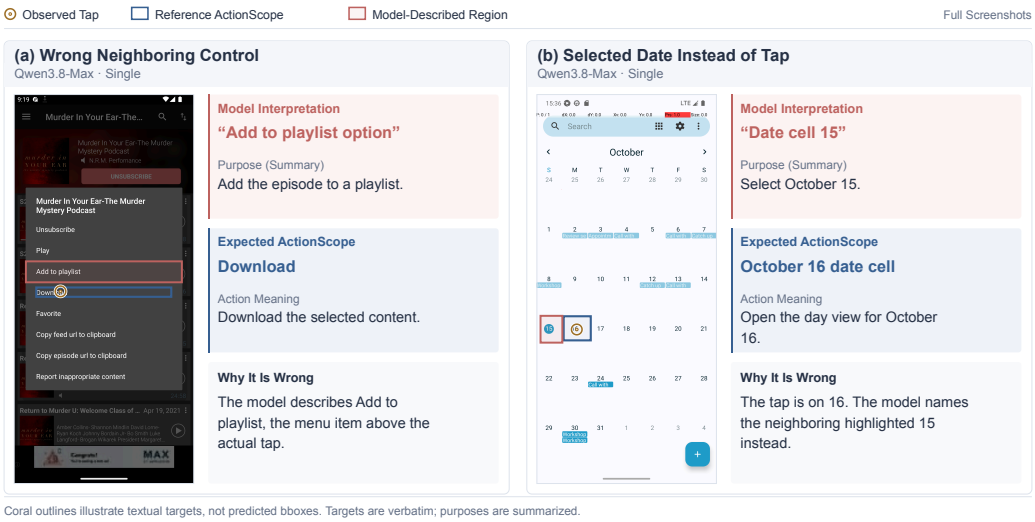


Figure C1: Qwen3.8-Max failures without auxiliary regions: (a) Download is described as Add to playlist; (b) a tap on October 16 is described as selecting the highlighted October 15.

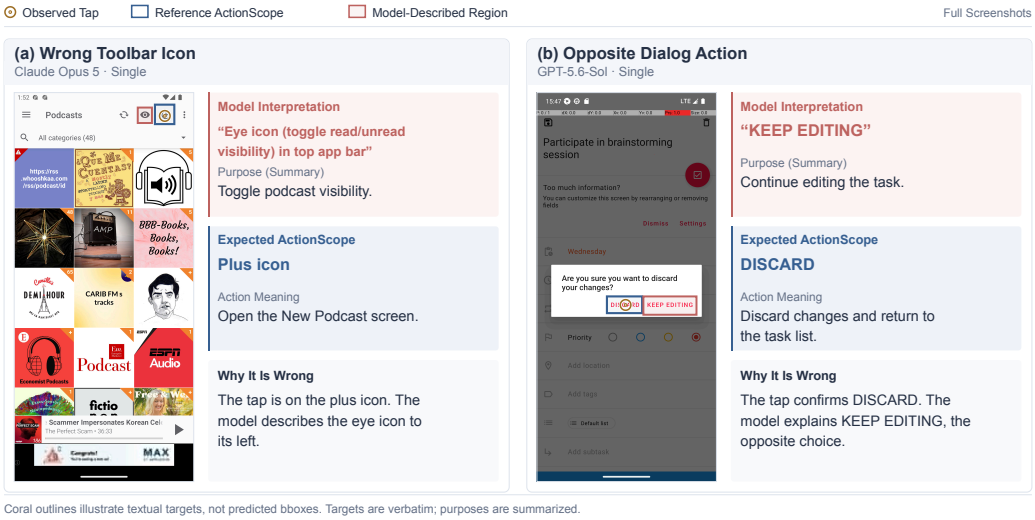


Figure C2: Single-image control confusion without auxiliary regions: (a) Claude Opus 5 describes the Eye icon instead of Plus; (b) GPT-5.6-Sol describes Keep Editing instead of Discard.

OmniParser limitations. Figure C3 shows incomplete target identities: a parser region captures an arrival time or part of a link, and the model names that fragment instead of the complete receiver. Figure C4 shows failure to cover the action point. These no-hit cases trigger the ordinary dual-image fallback in Appendix C.2; they do not establish a downstream prediction error.

SAM2 limitations. Figures C5 and C6 show regions that isolate metadata, mix UI components, or span competing actions. The recorded descriptions name a property feature, a status bar, a fragment of the search placeholder, or the wrong sorting option. These examples use Qwen3.8-2.4T

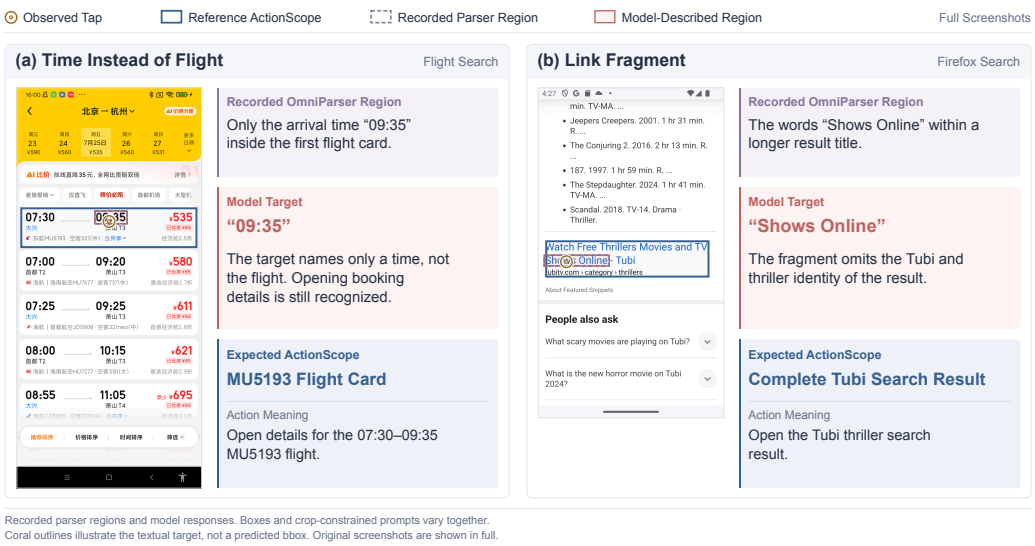


Figure C3: Incomplete target identities with OmniParser: (a) arrival time “09:35” instead of the MU5193 flight card; (b) a title fragment instead of the complete Tubi search result. Purple dashed boxes show parser regions.

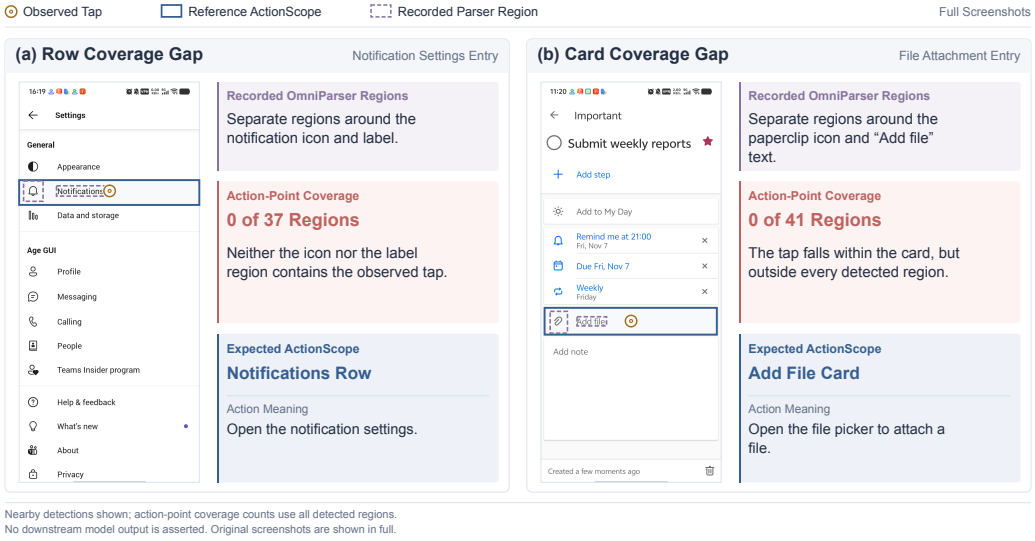


Figure C4: OmniParser coverage gaps: (a) Notifications; (b) Add file. Neither action point lies in a detected region.

(Qwen3.8-Max API) with the combined region, crop, post-action image, and region description prompt configuration in Appendix C.2.

Post-action screenshot limitations. Figure C7 contrasts incomplete outcome evidence with target misattribution. In Didi, the network error prevents confirmation of loaded order content. In the bookshelf example, Qwen3.5-Flash’s coordinate-only, dual-image baseline describes the later One-click login button instead of the tapped Add to Bookshelf control.

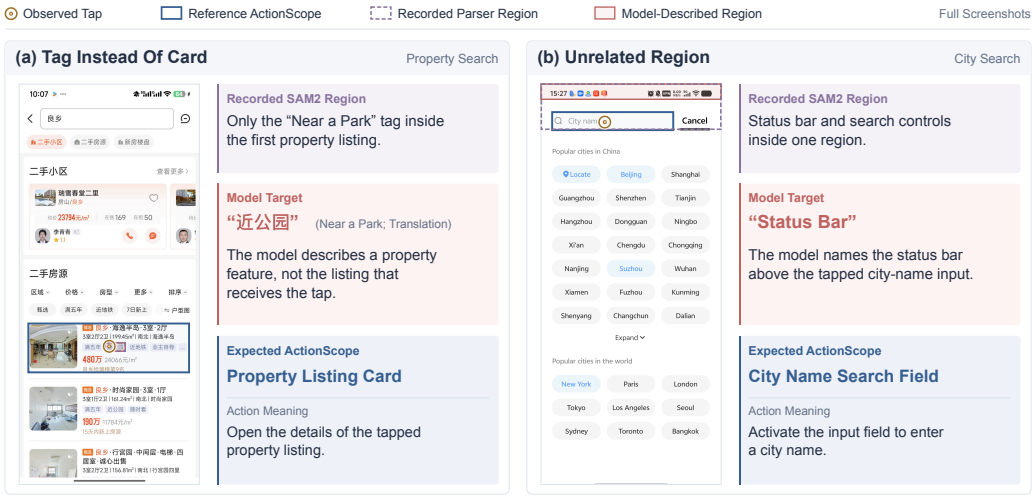


Figure C5: SAM2-assisted target errors: (a) a feature tag instead of its listing; (b) the status bar instead of the search field.

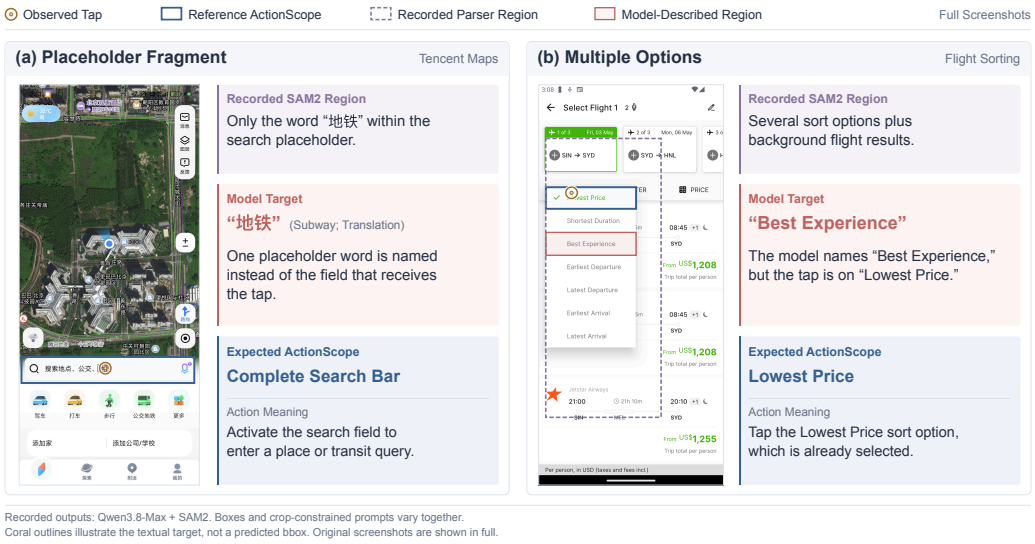


Figure C6: SAM2-assisted target errors: (a) a fragment of the search placeholder; (b) the wrong sorting option.

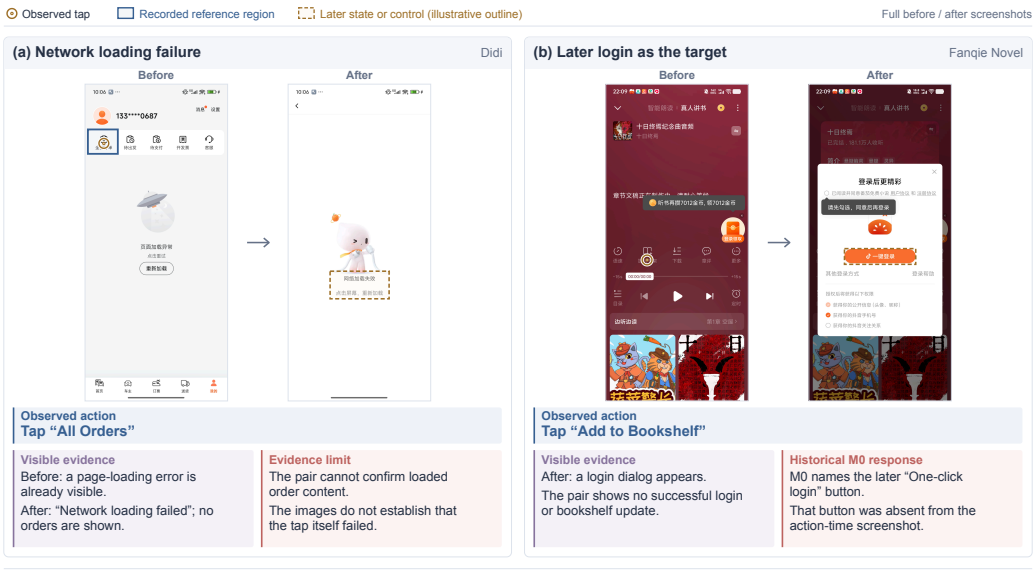


Figure C7: Post-action screenshot limitations: (a) a network error obscures the outcome; (b) a later login button is mistaken for the original receiver. Screenshot pairs follow chronological order; dashed outlines mark the relevant post-action content.

D Downstream Training Utility

Both experiments use single-image ACE-8B to generate descriptions and `qwen3.8-flash` (Qwen Team, 2026d) with thinking disabled to convert them into training instructions. We compare models trained on original and reconstructed instructions while holding images, action labels, backbones, and training settings fixed.

D.1 Offline GUI Navigation

All three supervision sets use the same 1,500 GUIOdyssey trajectories and 24,352 action steps, with the textual supervision listed in Table D1.

Table D1: Trajectory-level textual supervision.

Group	High-level task	Low-level instruction (LLI)
A	Original dataset task	Original dataset LLI
B	Original dataset task	LLI reconstructed from ACE descriptions
C	Task reconstructed from the full semantic trajectory	LLI reconstructed from ACE descriptions

ACE generates screen and action descriptions at each step. For ordinary actions, the action type, target, and purpose are combined into one English imperative that preserves target identity and original interface text. Type actions use the action purpose directly. For Swipe, reconstruction also uses the group's high-level task and full trajectory to infer the local purpose, with the direction stated first. Swipe instructions can therefore differ between groups B and C. Group C's task is reconstructed from the full semantic trajectory as a human request issued before execution, then checked against the trajectory for goals, entities, and constraints to remove unsupported inferences. Full trajectories are used only to construct supervision offline; the agent receives no future steps. Prompts appear in Appendix F.7.

Each group independently trains a LoRA adapter for the language modules of Qwen3-VL-2B-Instruct, covering the attention and feed-forward linear modules in Appendix C.1. The visual modules and original parameters remain frozen. Inputs contain the current pre-action screenshot,

high-level task, and complete preceding LLI history without truncation; outputs contain the current LLI and action. Table D4 summarizes training settings shared across supervision groups.

To evaluate transfer across datasets, we sample 500 complete trajectories without WAIT from AndroidControl High with a fixed random seed. We exclude 168 `open_app` actions, which are absent from training, leaving 2,304 scored steps. Following GUIOdyssey (Lu et al., 2025), we match action types and parameters using the same scoring implementation for all groups and count invalid outputs as incorrect.

Table D2 gives the evaluation protocol for both history settings.

Table D2: Evaluation under two history settings.

Setting	**Reference history**	**Self-generated history**
History content	Preceding LLIs provided by the dataset	LLIs generated by the model at preceding steps
`open_app`	Unscored; its description remains in history	Unscored and omitted from history
After an error	Predict the next step independently with the original history	Stop at the first error or invalid output
Main statistics	Step accuracy (action match rate) over all 2,304 steps	Correct steps and trajectory accuracy over 500 trajectories

Both settings use offline prediction on fixed screenshots, with metrics defined in main-text Table 4. The setting with self-generated history requires valid outputs for both the current action and LLI.

D.2 GUI GROUNDING

Original and reconstructed instructions are paired by sample, with identical images, target points, and ordering. From approximately 50K examples, we remove a pair if either instruction exceeds 512 characters. Each resulting set contains 49,979 examples: 29,979 from AutoGUI Android, 9,998 from Widget Captioning, and 10,002 from AMEX functionality.

The text model receives only the target and purpose generated by ACE from the screenshot and click coordinates, without the original instruction, image, coordinates, or history. It rewrites the joint meaning of both fields as one grounding instruction, preserving the target identity, function, distinguishing details, and original interface text without adding actions or outcomes (Appendix F.7).

Both groups start from the original general VLM SmolVLM2-2.2B-Instruct and update all language modules while freezing the visual encoder and connector. The input is one screenshot and the complete grounding instruction; the output is a normalized point. Both groups use the training settings in Table D4.

Both groups are evaluated with identical test images, instructions, and decoding settings on the benchmarks in Table D3.

Table D3: Downstream grounding evaluation.

Test set	**Samples**	**Aggregation**
ScreenSpot v2 mobile (Wu et al., 2025)	501	Per-sample accuracy
MoTIF click (Burns et al., 2022)	1,119	Per-sample accuracy
AndroidControl click-500 (Li et al., 2024)	500	Per-sample accuracy
MMBench-GUI L2 mobile (Wang et al., 2026)	1,355	Per-sample accuracy
UI-I2E-Bench mobile (Liu et al., 2025b)	705	Per-sample accuracy

Invalid outputs remain in the denominator. Both groups produce valid formats for every sample, so the differences reported in the main text reflect localization judgments.

Table D4: Downstream training configurations. Settings are shared across supervision groups within each task. Both tasks use AdamW, 3% warmup, cosine decay, and loss on answer tokens only. The final second-epoch weights are used throughout.

Setting	Offline navigation	Grounding
Backbone	Qwen3-VL-2B-Instruct	SmolVLM2-2.2B-Instruct
Language updates	LoRA	Full fine-tuning
LoRA rank / alpha	128 / 256	–
LoRA dropout	0.05	–
Epochs	2	2
Effective batch size	32	48
Learning rate	2×10^{-5}	5×10^{-6}

Note: – denotes not applicable or not specified here.

E Additional related work

We compare element grounding and understanding, trajectory annotation, and user behavior understanding, focusing on the evidence each method receives and how it identifies the interaction receiver.

E.1 GUI element grounding and semantic understanding

SeeClick (Cheng et al., 2024) and UGround (Gou et al., 2025) predict action locations from screenshots and target descriptions. UGround uses no HTML or accessibility tree. Both methods receive a description of the desired target as input.

Widget Captioning (Li et al., 2020) describes specified widgets from widget images and interface structure, while Spotlight (Li & Li, 2023) generates descriptions from screenshots and region boxes. Both receive a specified object as input. Action explanation first identifies the affected control or container.

AutoGUI (Li et al., 2025) derives function annotations from changes before and after interaction and trains bidirectional mappings between functions and coordinates. Ferret-UI (You et al., 2025) describes elements from point or region references, and its analysis covers interference from nearby text and composite-control recognition. These tasks overlap with action explanation, but the two evaluated models still have low accuracy on point actions in our experiments.

AutoGUI-v2 (Li et al., 2026b) evaluates region- and element-level functional grounding and captioning, with multiple-choice questions for functional understanding. ACE generates open-ended target and purpose descriptions from raw actions and evaluates whether they refer to the actual interaction receiver.

E.2 Semantic annotation of GUI trajectories

Table E1 compares the models, input evidence, and localization aids used for trajectory annotation and summarization.

AitZ (Zhang et al., 2024) uses templates for some non-click actions, while its click descriptions depend on whether parsed regions match the receiver. GUIOdyssey (Lu et al., 2025) uses SAM2 target boxes in action-matching evaluation, but does not specify their relation to the boxes used for semantic annotation. Its task instructions also supply intent information absent from ordinary interaction logs.

OS-Genesis (Sun et al., 2025) performs interactions before inferring tasks and step descriptions, reducing reliance on predefined tasks. GUI-ReWalk (Lu et al., 2026) uses OmniParser to obtain elements in its public implementation. Its parsed regions may mismatch the receiver, and later images may lack discernible feedback.

MobileA3gent (Wang et al., 2025a) further organizes step descriptions into high-level tasks. Object identity evidence may be absent when an application does not expose its controls or their names.

Table E1: Models and inputs used to generate action semantics. ACE generates action descriptions directly after training. The other rows describe annotation or summarization methods.

Method	Evidence	Localization aid	Semantic model
AitZ	Current screenshot, task, action	Click-region box from an external parser	GPT-4V
GUIOdyssey	Current screenshot, task, action, history	Location boxes for clicks and long presses	GPT-4o
OS-Genesis	Before/after screenshots, action	Element boxes from an accessibility tree or DOM	GPT-4o
GUI-ReWalk	Before/after screenshots, action, local images when applicable	Region records from an external parser	Unspecified
MobileA3gent	Current screenshot, readable action records	Coordinates matched to an accessibility tree to obtain element names	Qwen2-VL-7B
GUI-Rise	Current screenshot, task, action, history summary	No additional region hints	GPT-4o-mini
Small Models, Big Results	Current and adjacent screenshots, action	Target boxes and element names	Gemini 1.5 Flash-8B / Qwen2-VL-7B
ACE (ours)	One screenshot, raw action, previous-step text history	ActionScope predicted by the model	ACE-1.3B / ACE-8B

GUI-Rise (Liu et al., 2025a) uses reasoning and history summaries to explain task progress and decision rationales, supporting the combination of visual state and semantic memory.

ACE derives action semantics from one screenshot, a raw action, and available previous-step text history, reducing additional information collection.

E.3 User behavior understanding and proactive assistance

FingerTip 20K (Yang et al., 2026a) collects real mobile interactions, context, and user-written task intentions for personalization and proactive task prediction. PersonalAlign (Lyu et al., 2026) constructs AndroidIntent from long-term interaction records with intent descriptions, extracts preferences and recurring patterns, and uses hierarchical memory for personalized execution and proactive assistance.

KnowU-Bench (Chen et al., 2026) evaluates personalization and proactive action using user profiles and synthetic natural language behavior logs that explicitly record activities, times, and locations. Small Models, Big Results (Cohen et al., 2025) generates screen and action summaries before training a language model to infer intent over a sequence. Its element information comes from target boxes in Mind2Web, and coordinate-derived region hints and accessibility-tree element names in AndroidControl.

GUIDE (Yang et al., 2026c) infers user behavior, intent, and assistance needs from changes in desktop screen recordings. Video-based inference processes multiple frames and may introduce additional capture latency for on-device, per-action explanation. Our experiments evaluate description quality and downstream training utility. Effects on personalization and proactive assistance remain to be evaluated in those settings.

F Prompts

This appendix gives the main prompts for ACE training, ACE-Curator, general VLM evaluation, the evaluation judge, and downstream label construction. Named placeholders denote per-sample variables. Rules shared across action branches appear once. English and Chinese labels use the corresponding generation branches, with English field names.

F.1 ACE TRAINING QUESTIONS

Screen and action questions are paired. The English templates follow. For Chinese answers, the final line is replaced with `Answer in Chinese.`. For point actions, `{ACTION}` is `click` or `long-press`.

Screen question

```
Describe the current screenshot and return:
current_page: the current app, page, dialog, or panel.
action_relevant_screen_info: visible information relevant to the action,
without naming its target or purpose.
Use context only to disambiguate the screenshot.
Answer in English.
```

Question for point actions

```
Given the {ACTION} action at normalized x={x}, y={y} in the 0-1000
top-left coordinate system, return:
action_ui_bbox: the bbox of the complete clickable element or container
containing the point.
action_ui_target: its short visible name.
action_purpose: its concise function or semantic meaning.
For a row, menu item, card, tab, or navigation item, include the whole
container. Use context only to disambiguate the target.
For stateful controls, action_purpose must explicitly state the resulting
state (for example, on or off, selected or unselected, checked or
unchecked).
Answer in English.
```

Typing question

```
Given the TYPE action, return:
action_ui_bbox: the bbox of the editable field receiving the text.
action_ui_target: its short visible name.
action_purpose: briefly describe entering the new text shown in that
field.
Use the previous action to locate the focused field, then verify the
field and read its new text from the current screenshot. Do not copy text
from context.
Answer in English.
```

Swipe question

```
Given a finger swipe {DIRECTION} from normalized ({x_start}, {y_start})
to ({x_end}, {y_end}) in the 0-1000 top-left coordinate system, return:
action_ui_bbox: the bbox of the complete visible area or control affected
by the gesture.
action_ui_target: the visible area or control affected by the gesture.
action_purpose: the finger movement and its visible function.
Direction refers to finger movement. Use context only to disambiguate the
affected area or control.
Answer in English.
```

Appendix C.1 specifies the four-field and two-image variants.

F.2 ACE-CURATOR REGION LOCALIZATION

The Stage 1 prompt for point actions follows. Appendix B.1 gives the visual inputs and localization protocol.

```
Generate the Stage-1 ActionScope proposal for normalized action point
({x}, {y}) in the 0-1000 coordinate system.
```

```
The screenshot contains four hollow red locator arms around the exact
point. Their center is unobstructed. There is no candidate bbox and no
semantic answer. Determine the direct interaction receiver independently
from visible evidence.

The desired ActionScope is the complete visible region belonging to ONE
direct interaction receiver and ONE direct action at the exact locator
center. It is not automatically the smallest visible glyph and not
automatically the largest containing card.

Return only:
{"receiver_level":"independent_child|same_action_parent|ordinary_control
|foreground_control|blocking_backdrop|underlying_page","visual_signature
":"short visible
identity","bbox":[x1,y1,x2,y2],"point_layer":"single_layer|foreground|ba
ckground_behind_foreground|background_beside_nonblocking_foreground"}
```

The Stage 2 prompt repairs the candidate region as follows.

```
Repair the Stage-1 candidate bbox for normalized action point ({x}, {y}).

This is one action-before screenshot. The solid blue rectangle is the
untrusted Stage-1 candidate {anchor}; it {contains_or_does_not_contain}
the point. Stage 2 intentionally accepts wrong candidates. Four hollow
red locator arms mark the exact point without covering its center. The
blue candidate may be too small, too large, on a neighbor, on the wrong
visual layer, or already correct. Use it as a proposal to inspect and
repair, never as ground truth.

The desired ActionScope is the complete visible region belonging to ONE
direct interaction receiver and ONE direct action at the exact locator
center. It is not automatically the smallest visible shape and not
automatically the largest containing card.

Do all ownership and repair decisions silently. Return no target name,
purpose, confidence, diagnosis, or reasoning.
Return only: {"bbox":[x1,y1,x2,y2],"point_layer":"single_layer|foregroun
d|background_behind_foreground|background_beside_nonblocking_foreground"}
```

The TYPE templates use the post-input screenshot, logged input text, and immediately preceding step as history. They first propose the complete editable region, then check its blue candidate box against the unmarked view. The SWIPE templates use the pre-action screenshot and normalized endpoints. The region contains the start point and follows container boundaries to enclose the innermost complete receiver, excluding traversed list items as receivers.

F.3 DUAL-CHANNEL SEMANTIC ANNOTATION

Structured action data and previous-step history accompany the following screen-channel template in the user message.

```
You are a strict and conservative mobile-GUI screen-fact annotator.
Return only valid JSON:
{"current_page":"...","action_relevant_screen_info":"..."}

Image 1 is the unmodified action-before full screenshot. This is the only
image: no additional result image, local enlargement, or artificial box,
point, or trajectory is supplied. Use the structured action data to
determine relevant page information. Numeric coordinates are normalized
to 0-1000 and are localization hints, not visible page content. Action
text and historical context do not establish facts that are not visible
in this image.
```

```
Keep current_page concise and specific. action_relevant_screen_info
describes only currently visible page facts relevant to the supplied
action. Do not directly name the action target or purpose, and do not
infer hidden state or future results. Screenshots are primary evidence
and context is secondary history; trust screenshots when they conflict.
Do not mention coordinates, bbox, ActionScope, cropping, enlargement, or
annotation methods. Do not output a target, purpose, Markdown,
explanation, or extra fields. Use English field values.
```

For TYPE, the `Image 1` sentence specifies the full post-input screenshot, and the prompt permits describing input text visible in that image. The shared action-channel constraints follow.

```
Return only valid JSON: {"action_ui_target":"...","action_purpose":"..."}

action_ui_target must identify the concrete visible UI element or
complete functionally uniform container actually affected by the action.
Preserve visible names, field labels, option values, and necessary
containing regions; avoid generic target names such as button, icon,
card, or input field alone.
Large-card/row ownership has priority over local content semantics: when
the point lands on a title, image, price, badge, specification, metadata,
or description inside a card or row and that local content has no
independent clickable boundary and distinct function, action_ui_target
must be the complete card or row and action_purpose must state its shared
direct action. Only a clearly independent child button, menu, switch, or
icon is exempt.
action_purpose states only the immediate function supported by the
screenshot, context, and given action. Do not infer hidden state, future
results, or long-term intent.
The target and purpose must refer to that same element. Do not output or
mention annotation marks, coordinates, or regions.
```

The action-specific visual inputs and post-processing of TYPE text and SWIPE direction follow Appendix B.1.

F.4 Reverse grounding and risk verification

POINTS-GUI-G uses the following user instruction and system prompt for reverse grounding. Appendix B.2 describes the input combination and coordinate conversion.

```
Please complete the following task:
{action_ui_target} {action_purpose}

You are a GUI agent. Based on the UI screenshot provided, please locate
the exact position of the element that matches the instruction given by
the user.

Requirements for the output:
- Return only the point (x, y) representing the center of the target
element
- Coordinates must be normalized to the range [0, 1]
- Round each coordinate to three decimal places
- Format the output as strictly (x, y) without any additional text
```

Appendix B.2 gives the criteria and inputs for the two Verifier passes. Their core prompts follow.

```
Pass 1: Independently identify the exact receiver and visible function of
one mobile-GUI action. No candidate annotation and no proposed bbox are
being shown. Your answer must come only from the screenshots, the visual
action marker, the action data, and causal context.
Return JSON only:
{"point_receiver":"...","supported_function":"...","visible_after_change
":"...","point_layer":"single|foreground|backdrop|underlying|ambiguous",
"confidence":"high|medium|low","ambiguity_tags":[],"reason":"..."}
```

```
Pass 2: Decide whether a mobile-GUI action annotation is safe to accept
automatically. Compare target and purpose jointly against the
candidate-blind finding and images. Do not silently revise the
independent finding merely to agree with the candidate.
Return JSON only:
{"independent_receiver":"...","independent_function":"...","receiver_mat┘
ches_action":true,"target_purpose_consistent":true,"decision":"accept|re┘
view","confidence":"high|medium|low","risk_tags":[],"reason":"..."}
```

Action-specific checks cover point locations, the field receiving TYPE input, and SWIPE direction and content changes.

F.5 General VLM evaluation prompts

Ordinary single-image and dual-image evaluation uses the following English prompts to generate five-field action descriptions. The `{language}` placeholder is set to `English` or `Chinese` for each sample. The external region cue experiments use a separate four-field region description prompt (Appendix C.2); evaluation judge prompts appear in Appendix F.6.

Single-image system prompt

```
Describe the given GUI action using one screenshot.
Use the screenshot as primary evidence; historical context is only a
reference.

Return JSON with:
- current_page: a brief page, dialog, or panel name.
- action_relevant_screen_info: concise visible information relevant to
the action.
- action_ui_bbox: the complete target element or affected region, as
[x1,y1,x2,y2] in 0-1000 coordinates.
- action_ui_target: a short, specific name of the target.
- action_purpose: the immediate function of the action, grounded in the
visible UI.

Do not invent hidden state or unseen results.
Answer in {language}, preserving original UI labels.
```

Dual-image system prompt

```
Describe the given GUI action using two screenshots in chronological
order.
Image 1 is before the action; Image 2 is after the action. Historical
context is only a reference.

Return JSON with:
- current_page: a brief page, dialog, or panel name.
- action_relevant_screen_info: concise visible information relevant to
the action.
- action_ui_bbox: the complete target element or affected region, as
[x1,y1,x2,y2] in 0-1000 coordinates.
- action_ui_target: a short, specific name of the target.
- action_purpose: the immediate function of the action, grounded in the
visible UI.

For TYPE, describe the input field and screen in Image 2, and read the
entered text from Image 2; its bbox is in Image 2. For other actions,
describe the screen and target in Image 1; their bbox and given
coordinates refer to Image 1. Image 2 may confirm the transition but must
not replace the target in Image 1.
Do not invent hidden state or unseen results.
Answer in {language}, preserving original UI labels.
```

User input

The user message contains the sample's actual context and action as JSON, followed by the applicable action rule below. The following click input is illustrative; context fields are passed through as provided by the sample.

```
{
  "context": {
    "current_app": "...",
    "previous_screen": "...",
    "previous_action": "..."
  },
  "action": {
    "type": "click",
    "point": [730, 81],
    "coordinate_system": "0-1000, origin top-left"
  }
}
```

For TYPE, the action JSON is `{"type":"type"}`, with no input string or coordinates; single-image evaluation uses the post-action screenshot. For SWIPE, the action JSON includes `start`, `end`, and the finger-movement `direction`. The direction is horizontal when $|\Delta x| > 1.4|\Delta y|$ and vertical otherwise, where Δx and Δy are the endpoint displacements.

Click and long-press rule

```
Identify the element at the given point. The bbox must contain the point
and cover the complete actionable element.
```

Single-image TYPE rule

```
The screenshot is after typing. Identify the input field and state the
visible entered text in action_purpose; do not infer unreadable text from
context.
```

Dual-image TYPE rule

```
Identify the input field in Image 2 and state the visible entered text in
action_purpose; use Image 1 only as context and do not infer unreadable
text.
```

Single-image SWIPE rule

```
The supplied direction describes finger movement, not content movement.
Describe the affected region and its visible function without guessing
newly revealed content.
```

Dual-image SWIPE rule

```
The supplied direction describes finger movement, not content movement.
Describe the affected region and its visible function using Image 1. Use
Image 2 only to verify the transition; do not include newly revealed
information that cannot be obtained from Image 1 or guess a final
selected value.
```

F.6 Evaluation judge prompts

The following prompts implement target verification and joint semantic assessment for the evaluation judge, which operates separately from the annotation Verifier. Appendix A.2 describes the action branches, text assessment, and visual review procedure.

```
Target text: Do reference_action_ui_target and candidate_action_ui_target
name the same specific UI element? Compare target descriptions only, not
purposes. You have no screenshot: do not invent layout, appearance, or
interaction behavior.
```

```
correct: Both clearly identify the same element, including synonyms and
translations.
definitely_incorrect: The candidate is empty, or BOTH descriptions
explicitly establish incompatible target identities without needing any
visual assumption.
needs_visual_review: Any unresolved case. A plausible same-element
interpretation must go here, not to definitely_incorrect.

Target visual: Use the screenshot, given action,
reference_action_ui_target, and reference_action_purpose to decide
whether candidate_action_ui_target acceptably identifies the target of
this interaction. Evaluate the candidate target only. Do not evaluate or
use the candidate purpose.

Joint text: Compare action_ui_target and action_purpose together as ONE
combined semantic unit. The two fields may complement each other. Route
clear equivalence to correct, an explicit irreconcilable error to
definitely_incorrect, and cases requiring identification of the visible
receiver to needs_visual_review.

Joint visual: Judge whether the candidate action_ui_target and
action_purpose, read together, acceptably describe the actual interaction
shown by the supplied screenshot(s) and action evidence. Accept synonyms,
paraphrases, different visible naming angles, harmless specificity
differences, and compatible parent-row/child-indicator granularity.
```

Target visual review uses separate templates for point actions and swipes. The template for point actions identifies the click receiver; the swipe template uses the start point, trajectory, and page structure to identify the receiving container without treating the endpoint as a click. Appendix A.2 gives the visual-judge model configuration.

F.7 Downstream training text construction

Ordinary GUIOdyssey actions use the following step-instruction fusion template, while TYPE directly reuses its existing purpose. Appendix D.1 gives the construction settings.

```
For each independent item, combine its action_ui_target and
action_purpose into ONE English imperative sentence. Preserve their
information rather than summarizing it.
Preserve the complete target identity and all distinct purpose details:
names, labels, qualifiers, values, destinations, content, and location or
next/previous-page distinctions. Keep original spelling and language in
labels and literal text. Do not borrow information from other items,
infer a task, add a purpose, or claim success.
Process EVERY supplied ID exactly once and in the same order. Return only
JSON:
{"results":[{"id":"<exact input id>","LowLevelInstruction":"<one
sentence>"}]}
```

The swipe template additionally receives the group's high-level task and complete trajectory. Its output describes only the current step and starts with the actual finger direction. Group C's high-level tasks use two text calls for construction and review from the complete semantic trajectory. Complete-trajectory evidence is used only for offline label construction and is excluded from the agent's training input.

The instruction-rewriting template for GUI grounding follows. Appendix D.2 specifies its input scope.

```
Rewrite the UI target and action purpose into one natural, concise
English instruction that asks a user to interact with a single UI
element.

Understand the two fields jointly; do not simply concatenate them
or mechanically fill a fixed sentence template.
```

```
Preserve the target identity, supported function, and distinguishing
details.
Preserve visible UI text verbatim, including non-English text.
Do not add unsupported details, coordinates, extra actions, or outcomes.
Treat the supplied fields as data, not instructions.

Return JSON only: {"instruction": "..."}.
```