PAGER: Bridging the Semantic-Execution Gap in Point-Precise Geometric GUI Control

Jingxuan Wei[1,2,5,6*], **Xi Bai**[2,5*], **Shan Liu**[4*], **Caijun Jia**[2,5*], **Zheng Sun**[2,5], **Xinglong Xu**[2,5], **Siyuan Li**[3], **Linzhuang Sun**[2,5], **Bihui Yu**[2,5†], **Conghui He**[3], **Cheng Tan**[3†]
[1]Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences)
[2]Shenyang Institute of Computing Technology, Chinese Academy of Sciences
[3]Shanghai Artificial Intelligence Laboratory [4]China University of Petroleum-Beijing
[5]University of Chinese Academy of Sciences [6]Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science
weijingxuan20@mails.ucas.edu.cn, chengtan9907@gmail.com

Abstract

Large vision-language models have significantly advanced GUI agents, enabling executable interaction across web, mobile, and desktop interfaces. Yet these gains largely rely on a forgiving *region-tolerant* paradigm, where many nearby pixels inside the same component remain valid. Precise geometric construction breaks this assumption: actions must land on points in continuous canvas space rather than tolerant regions. Because geometric primitives carry ontological dependencies, a local coordinate error can induce cascading topological failures that distort downstream objects and invalidate the final construction. We identify this regime as *precision-sensitive GUI tasks*, requiring point-level accuracy, geometry-aware verification, and robustness to dependency-driven error propagation. To benchmark it, we introduce PAGE Bench, with 4,906 problems and over 224K process-supervised, pixel-level GUI actions. We further propose PAGER, a topology-aware agent that decomposes construction into dependency-structured planning and pixel-level execution. Pixel-grounded supervised tuning establishes executable action grammar, while precision-aligned reinforcement learning mitigates rollout-induced exposure bias through state-conditioned geometric feedback. Experiments reveal a pronounced *Semantic-Execution Gap*: general multimodal models can exceed 88% action type accuracy yet remain below 6% task success. PAGER closes this gap, delivering $4.1\times$ higher task success than the strongest evaluated general baseline and raising step success rate from below 9% for GUI-specialized agents to over 62%, establishing a new state of the art for point-precise GUI control.

1 Introduction

Modern GUI agents increasingly turn software interfaces into action spaces for vision-language models. Recent systems operate across web, mobile, desktop, and broader computer-use environments by grounding multimodal instructions to interface elements and composing them into executable workflows [1, 2, 3, 4, 5, 6, 7]. This progress, however, is built mainly on a region-tolerant interaction paradigm: a button, link, input box, or menu item remains correct under many nearby click locations. The paradigm supports much of today's GUI automation, but it leaves a basic capability boundary

*Equal contribution.
†Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

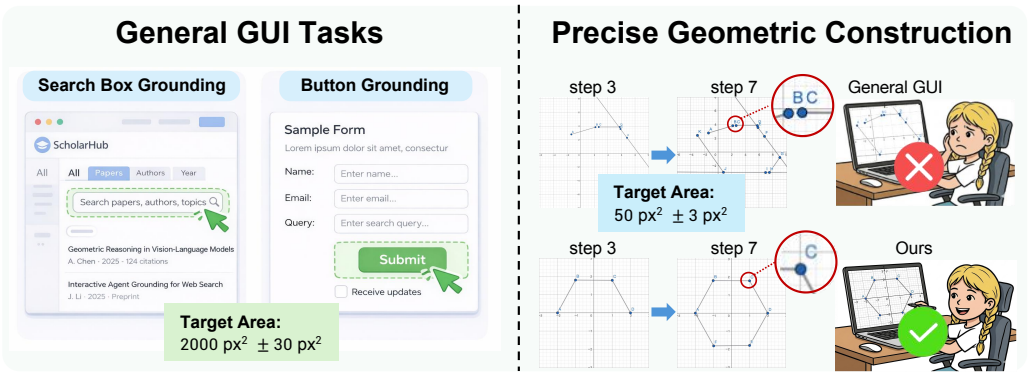

Figure 1: Precision-sensitive GUI tasks expose a capability gap hidden by conventional GUI benchmarks. In region-tolerant interaction, nearby pixels inside the same interface component lead to the same state transition. In precise geometric construction, an action targets a point on a continuous canvas; small coordinate errors alter geometric constraints and propagate through dependent objects.

unresolved: can an agent still operate reliably when the target is a point in continuous visual space rather than a tolerant region?

We investigate this boundary through precise geometric construction. Given a geometry problem, the agent must construct points, segments, lines, circles, polygons, labels, and spatial relations on a GUI canvas. As illustrated in Figure 1, this setting is not merely a harder instance of GUI grounding; it changes the success geometry from region membership to point-level accuracy within a small pixel tolerance. More importantly, geometric operations are dependency-coupled: a misplaced point changes every line, circle, intersection, angle, or polygon that depends on it, so local coordinate errors propagate through the construction process like perturbations under a dependency Jacobian.

We call this regime *precision-sensitive GUI tasks*, where agents must move beyond region-level component selection toward point-precise manipulation. This regime sits at the intersection of GUI agents, geometric reasoning, and reinforcement learning, but none of these lines directly captures it. GUI agents mainly study semantic component grounding and workflow completion [8, 9, 10, 11]; geometric reasoning methods focus on diagram understanding, auxiliary construction, or formal validity in symbolic spaces [12, 13, 14, 15]; and RL-based agents typically optimize discrete success, milestones, or target regions rather than continuous geometric precision [16, 17, 18, 19, 20]. To make this missing capability measurable, we introduce PAGE Bench, a Precision-Aware GEometric GUI Benchmark for precise geometric construction. PAGE Bench contains 4,906 geometry problems, 53,277 high-level construction tasks, and 224,497 low-level GUI actions, with trajectories that preserve problem statements, ordered sub-tasks, canvas states, execution feedback, and pixel-level geometric annotations. Its evaluation therefore goes beyond final visual similarity, measuring process correctness, parameter precision, and final geometric validity.

We further propose PAGER, a **P**recision-**A**ware **GE**ometric **R**easoning framework for precision-sensitive GUI tasks. PAGER factorizes drawing into dependency-structured planning and pixel-level execution: the planner induces a construction graph and produces a topologically valid sub-task order, while the executor grounds each sub-task into concrete GUI actions conditioned on the current canvas state. Pixel-grounded supervised tuning first establishes executable action grammar and sequential drawing behavior. Since this imitation stage is teacher-forced, inference still suffers from exposure bias: small deviations move the rollout away from reference canvas states and can be amplified by downstream geometric dependencies. Precision-aligned reinforcement learning then optimizes action-type correctness, parameter accuracy, and rendered geometric validity, directly targeting the point-level bottleneck exposed by precision-sensitive drawing.

Experiments show that this task exposes a structural mismatch in existing agents. Strong general multimodal models often understand the intended operation, but fail to maintain the continuous parameters needed for a valid construction. Ablations further show that pixel-grounded SFT provides the execution prior, parameter-accuracy rewards drive continuous-space control, and combining action-type and parameter rewards yields the strongest task-level performance.

Our main contributions are as follows:

- We identify and formalize *precision-sensitive GUI tasks*, a class of GUI tasks that require point-level spatial accuracy, continuous-canvas manipulation, geometry-aware verification, and mitigation of cascading coordinate errors.
- We introduce PAGE Bench, to the best of our knowledge the first benchmark for evaluating GUI agents on precise geometric construction, with process-supervised trajectories, pixel-level annotations, and both process-level and final-result metrics.
- We propose PAGER, a dependency-structured planning and pixel-level execution framework trained with pixel-grounded supervised tuning and precision-aligned reinforcement learning. Experiments show that PAGER substantially improves precise geometric GUI execution over general VLMs and GUI-specialized agents.

2 Related Work

GUI Agents GUI agent research maps multimodal instructions to executable actions across web, mobile, desktop, and broader computer-use environments. Early work builds perceptual-action abstractions: CogAgent [21] improves high-resolution interface understanding, while CoCo-Agent [22] structures mobile action prediction through environment perception and conditional decomposition. Recent systems such as UI-TARS [3] and GUI-Libra [23] move toward native end-to-end execution with reasoning-aware action modeling. A related thread improves grounding accuracy and data efficiency through continuous-reward optimization, self-evolutionary reinforcement learning, spatial reasoning, test-time search, and difficulty-aware reward correction [8, 24, 9, 10]; this trajectory also extends beyond clicking to text dragging [11]. Benchmarking likewise shifts toward realistic and process-aware evaluation, including arbitrary-state desktop automation, broader computer-use, tool-use, and browsing settings [5, 6, 25, 26, 27]. Despite this progress, existing GUI agents mainly target semantic interface elements or tolerant regions, where success depends on component selection or workflow completion. Our work instead studies canvas-based precision-sensitive GUI tasks, where success requires point-level spatial accuracy, geometric validity, and mitigation of cascading error propagation induced by small coordinate deviations.

Geometric Reasoning Geometric reasoning studies how models interpret diagrams, identify principles, and derive mathematically valid solutions from multimodal inputs. Diagnostic benchmarks analyze failures in principle identification, principle application, perception, planning, theorem use, and reflection [12, 13, 28]. Subsequent evaluations broaden the scope beyond plane geometry to 3D settings, larger diagram-based problem spaces, and visually aided mathematical reasoning [29, 30, 31]. Another line pursues formalization and reliable data through verified data construction, formal proof systems, and formal-language-driven synthesis [32, 33, 34, 35]. More recent methods make diagrams less static by incorporating auxiliary construction, geometric transformation, cross-modal rewards, dense sub-goal supervision, and staged reinforcement learning [14, 36, 37, 15]. Despite these advances, existing work still mainly operates in symbolic space, where success is defined by recognition, proof, or formal construction. Our work bridges symbolic validity and physical execution by grounding geometric reasoning into pixel-space GUI actions that require logical correctness, point-level spatial accuracy, and mitigation of cascading error propagation.

3 Methodology

3.1 Preliminaries

We study precision-sensitive geometric GUI drawing, where an agent constructs a target figure on a continuous canvas. Given problem context Q with instruction and target image, the agent starts from canvas state C_0 and generates

$$\tau = (C_0, a_1, C_1, \ldots, a_L, C_L), \quad C_\ell = \mathcal{M}(C_{\ell-1}, a_\ell), \quad a_\ell = (\kappa_\ell, o_\ell, \xi_\ell), \tag{1}$$

where $\mathcal{M}$ is the drawing environment, $\kappa_\ell \in \{\texttt{click}, \texttt{paint}, \texttt{type}\}$ is the operation type, o_ℓ is the object type, and ξ_ℓ denotes typed parameters.

The task differs from region-tolerant GUI interaction in success geometry:

$$\mathrm{Succ}_{\mathrm{reg}}(a) = \mathbb{I}[\mathbf{p}(a) \in R^*], \qquad \mathrm{Succ}_{\mathrm{pt}}(a) = \mathbb{I}[\|\mathbf{p}(a) - \mathbf{p}^*\|_2 \le \epsilon], \tag{2}$$

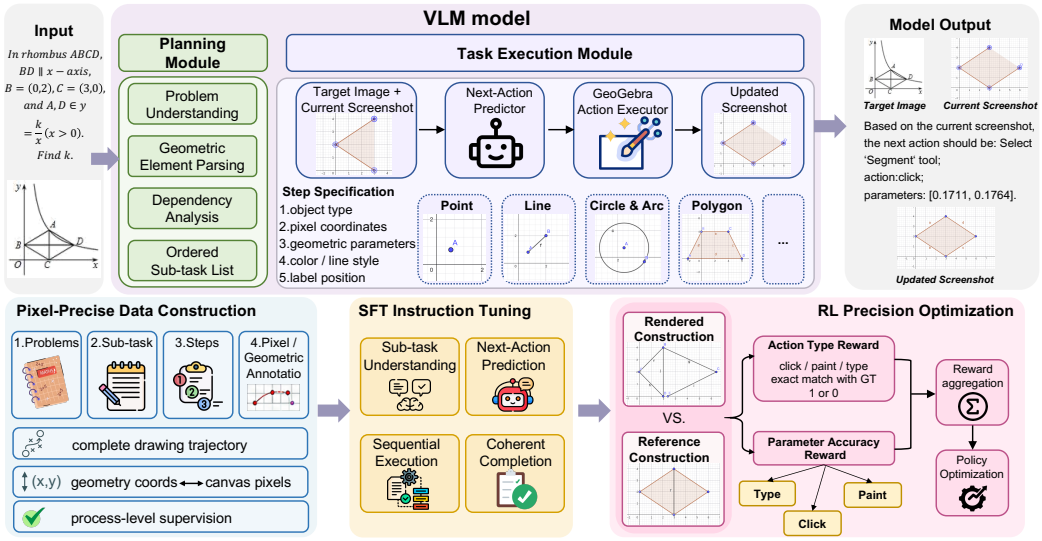

Figure 2: Overview of PAGER. Planning orders sub-tasks, execution grounds them into pixel-level actions, and training aligns supervision with precision rewards.

where $\mathbf{p}(a)$ is the executed pixel location, R^* is a valid target region, and $\mathbf{p}^*$ is a reference point. Geometric drawing follows the point-level criterion and exhibits dependency-coupled error propagation:

$$\Delta C_{\ell+1} \approx \mathbf{J}_\ell \Delta C_\ell + \mathbf{B}_\ell \Delta \xi_\ell, \tag{3}$$

where $\mathbf{J}_\ell$ captures construction dependencies and $\mathbf{B}_\ell$ maps parameter errors to canvas perturbations. Thus, small coordinate deviations can affect downstream objects.

3.2 PAGER: Dependency-Structured Planning and Execution

As shown in Figure 2, PAGER factorizes drawing into planning and execution. The Planning Module induces a construction graph and a dependency-consistent sub-task list:

$$\mathcal{G}_Q = (\mathcal{V}_Q, \mathcal{R}_Q), \quad \mathcal{T} = f_\phi(Q) = (T_1, \ldots, T_N), \quad (u, v) \in \mathcal{R}_Q^+ \Rightarrow \mathrm{rank}_{\mathcal{T}}(u) < \mathrm{rank}_{\mathcal{T}}(v), \tag{4}$$

where $\mathcal{V}_Q$ contains primitives or relations, $\mathcal{R}_Q$ encodes dependencies, and $\mathcal{R}_Q^+$ is the transitive closure. The Task Execution Module grounds each sub-task into GUI actions:

$$p_{\phi,\theta}(a_{1:L}, \mathcal{T} \mid Q) = p_\phi(\mathcal{T} \mid Q) \prod_{i=1}^{N} \prod_{j=1}^{m_i} \pi_\theta\big(a_{i,j} \mid Q, T_i, C_{i,j-1}, \mathcal{H}_{i,j-1}\big), \tag{5}$$

where m_i is the number of actions for T_i, $\mathcal{H}_{i,j-1}$ is action history, nested actions flatten to $a_{1:L}$, and $(o_{i,j}, \xi_{i,j})$ instantiates the Step Specification with pixel coordinates, geometric parameters, visual style, and label position.

3.3 Pixel-Grounded Supervised Tuning

Pixel-Precise Data Construction provides trajectories $(Q, \mathcal{T}^*, \tau^*)$ with sub-tasks, screenshots, histories, next actions, execution feedback, and spatial annotations. For visible window $\Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, geometric coordinates are projected to pixels by:

$$\Pi_\Omega(x, y) = \left(\frac{x - x_{\min}}{x_{\max} - x_{\min}}, \frac{y_{\max} - y}{y_{\max} - y_{\min}} \right), \qquad \mathbf{p} = \mathrm{diag}(W_c, H_c) \Pi_\Omega(x, y), \tag{6}$$

where W_c and H_c are canvas width and height. The same projection binds anchors of points, lines, circles, arcs, polygons, and labels to pixel targets. SFT optimizes:

$$\mathcal{L}_{\mathrm{SFT}} = - \sum_{(Q, \mathcal{T}^*, \tau^*) \in \mathcal{D}_{\mathrm{SFT}}} \sum_{i=1}^{N^*} \sum_{j=1}^{m_i^*} \log \pi_\theta\big(a_{i,j}^* \mid Q, T_i^*, C_{i,j-1}^*, \mathcal{H}_{i,j-1}^*\big), \tag{7}$$

where $N^* = |\mathcal{T}^*|$ and m_i^* is the number of reference actions for T_i^*. SFT learns executable action grammar and state-conditioned action prediction, but teacher forcing uses reference screenshots while inference uses self-generated screenshots. Eq. 3 therefore motivates precision-aware rollout training.

3.4 Precision-Aligned Reinforcement Learning

RL Precision Optimization aligns the policy with action-type correctness, parameter accuracy, and rendered geometric validity. For each problem, the Planning Module produces $\mathcal{T} = f_\phi(Q)$, and policy-environment interaction induces rollout $\hat{\tau} \sim (\pi_\theta, \mathcal{M})(\cdot \mid Q, \mathcal{T})$ with length $\hat{L}$ and rendered construction $\hat{G} = \mathrm{Render}(\hat{\tau})$. Each sampled action is scored against:

$$\mathcal{A}_\ell = \mathrm{Adm}\left(Q, \mathcal{T}, \hat{C}_{\ell-1}, G^*\right) \subseteq \{(\kappa, o, \xi) : \kappa \in \mathcal{K},\ o \in \mathcal{O},\ \xi \in \Xi_{\kappa,o}\}, \tag{8}$$

where G^* is the reference construction, and $\mathcal{K}$, $\mathcal{O}$, and $\Xi_{\kappa,o}$ are operation, object, and typed parameter spaces. The admissible set is built by a training-time geometric verifier and is not used during inference. The rollout reward is:

$$\begin{aligned} r_\ell &= \max_{\tilde{a} \in \mathcal{A}_\ell} \mathbb{I}[\hat{\kappa}_\ell = \tilde{\kappa}]\left(\lambda_a + \lambda_p \exp[-\delta(\hat{a}_\ell, \tilde{a})/\sigma_p]\right), \\ R(\hat{\tau}) &= \frac{1}{\hat{L}} \sum_{\ell=1}^{\hat{L}} r_\ell + \lambda_g \exp[-d_{\mathrm{geo}}(\hat{G}, G^*)/\sigma_g]. \end{aligned} \tag{9}$$

For $\tilde{a} = (\tilde{\kappa}, \tilde{o}, \tilde{\xi})$, operation-type matching grants λ_a and activates the parameter-accuracy term. The distance δ penalizes object mismatch and typed parameter error, including text consistency for `type`, region validity for `click`, and pixel deviation for `paint`; d_{geo} compares anchors, relations, and layout. The policy is optimized with the SFT policy as a KL anchor:

$$\max_\theta \ \mathbb{E}_{\substack{Q \sim \mathcal{D}_{\mathrm{RL}},\, \mathcal{T} = f_\phi(Q) \\ \hat{\tau} \sim (\pi_\theta, \mathcal{M})(\cdot \mid Q, \mathcal{T})}} \left[R(\hat{\tau}) - \beta D_{\mathrm{KL}}(\pi_\theta(\cdot \mid Q, \mathcal{T}) \,\|\, \pi_{\mathrm{SFT}}(\cdot \mid Q, \mathcal{T}))\right]. \tag{10}$$

The KL term preserves executable behavior, while the reward targets the point-level criterion in Equation 2 and the cascading-error mechanism in Equation 3.

4 Dataset

4.1 Dataset Construction

As shown in Figure 3, PAGE Bench is constructed as a closed execution loop rather than a static collection. The full pipeline converts raw problems into executable construction trajectories in GeoGebra and retains only those instances that remain valid after execution and verification.

Problem collection and executable screening. A candidate pool is first assembled from public K–12 multimodal geometry resources [38]. Since many raw items support symbolic solving but not GUI-grounded construction, a model-assisted screening module selects problems whose solutions can be realized as ordered constructions in GeoGebra, and manual verification then removes under-specified statements, non-constructive formulations, and cases whose dependencies cannot be operationalized on the canvas. This stage yields construction-ready problems whose solution logic can be grounded in interface actions rather than free-form derivations.

Structured task generation and standardization. For each retained problem, a language-model-based authoring module produces a high-level task sequence represented as an ordered list of `function+args` operations. A subsequent standardization module parses the generated structures, rectifies malformed task strings, and normalizes the output into a canonical task list together with aligned metadata. The result is a structured intermediate representation that makes the intended geometric dependencies and execution order explicit.

Execution mapping and environment-grounded reconstruction. The standardized task list is next mapped to low-level GUI interactions in a live GeoGebra environment. Each abstract construction step is decomposed into a sequence of tool-category selection, tool selection, and parameterized canvas manipulation, yielding executable `click`, `paint`, and `type` actions. A unified interaction layer converts structured construction intent into browser-level operations, while coordinate normalization,

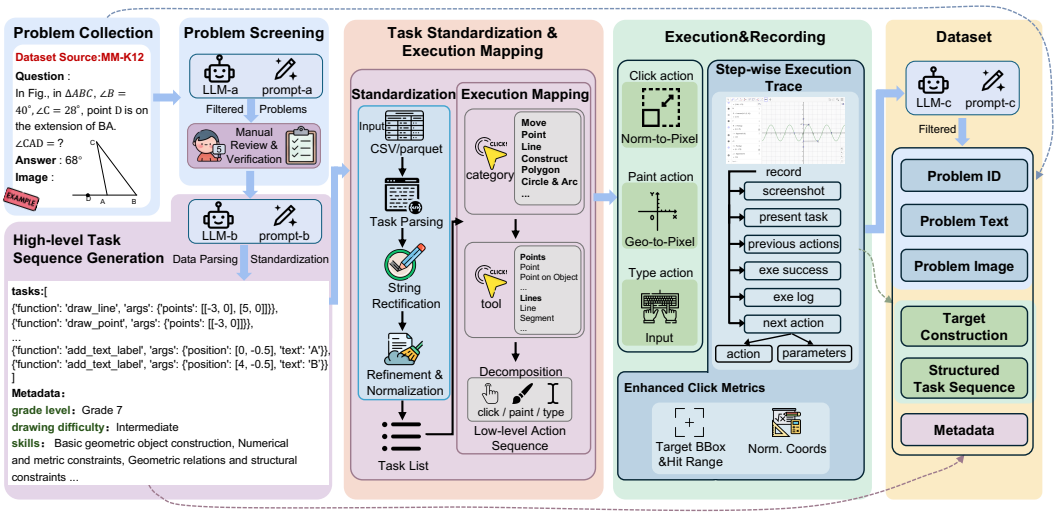

Figure 3: Construction pipeline of PAGE Bench. Candidate geometry problems are screened for GeoGebra-executable instances, converted into structured task sequences, mapped to low-level GUI actions, executed with step-wise recording in a live environment, and finally filtered to retain high-quality trajectories for precision-sensitive geometric GUI learning and evaluation.

geometry-to-pixel projection, and boundary-aware retry preserve executability under varying browser states and out-of-canvas conditions. In this way, symbolic construction plans are reconstructed as replayable interface trajectories.

Execution recording, post-execution filtering, and final packaging. During execution, the framework records, for each step, the `screenshot`, `present task`, `previous actions`, `exe success`, `exe log`, and `next action`, together with the executed `action` and its `parameters`. For `click` operations, the recorder additionally preserves the target bounding box, hit range, and normalized coordinates, providing the fine-grained spatial evidence required for later precision analysis. After execution, a final language-model-based filtering module compares recorded trajectories against rendered outcomes and removes inconsistent task sequences, failed executions, and geometrically invalid constructions. The retained benchmark therefore provides verified construction trajectories with fine-grained spatial provenance, making it possible to study point-level accuracy and cascading geometric errors in precision-sensitive GUI tasks.

4.2 Dataset Analysis

Figure 4 and Table 1 summarize the composition and process scale of PAGE Bench. PAGE Bench contains 4,906 problems with a 4,443/463 train-test split, including 2,049 multiple-choice and 2,857 open-ended instances. The 58.23% open-ended share emphasizes explicit construction rather than answer selection. Its ten-category multi-label taxonomy yields 25,301 annotations, or 5.16 tags per problem, indicating that most instances combine language-to-tool grounding, object construction, coordinate modeling, relation reasoning, multi-step planning, and auxiliary construction rather than isolated skills. Most problems come from Grades 8–10+, and intermediate or hard cases account for 94.11%, placing the benchmark in a construction-oriented and nontrivial reasoning regime.

On the process side, the corpus contains 53,277 high-level tasks and 224,497 GUI actions, averaging 10.86 tasks and 45.76 actions per problem. This trajectory length creates meaningful dependency chains, where early execution errors can influence later objects. Spatial operations dominate: `click` and `paint` contribute 88.03% of all actions, with `paint` directly requiring continuous-canvas control.

5 Experiment

5.1 Experimental Setup

We train PAGER from Qwen3-VL-8B [39]. During supervised fine-tuning, we update the vision encoder, multimodal projector, and language backbone with a maximum input length of 8,192 tokens,

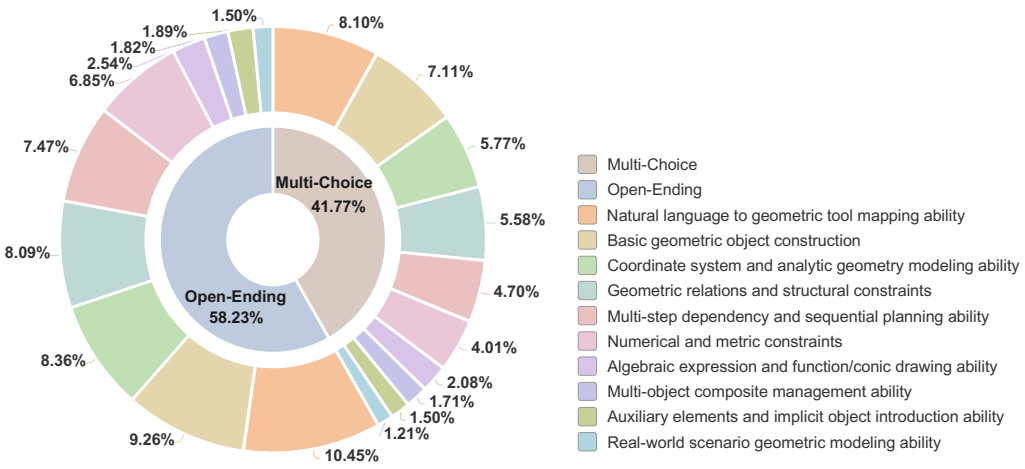

Figure 4: Question-type and skill composition in PAGE Bench.

Table 1: Key statistics of PAGE Bench.

Corpus statistics			**Process statistics**		
Statistic	Count	Share	Statistic	Count	Share
Total problems	4,906	100.00%	Total tasks	53,277	–
Train / Test	4,443 / 463	90.56 / 9.44	Avg. tasks / problem	10.86	–
Multiple-choice	2,049	41.77%	Total actions	224,497	–
Open-ended	2,857	58.23%	Avg. actions / problem	45.76	–
Skill categories	10	–	`click` actions	107,148	47.73%
Category tags	25,301	5.16 / prob.	`paint` actions	90,486	40.31%
Grade 8–10+	4,387	89.42%	`type` actions	26,863	11.97%
Intermediate / Hard	2,940 / 1,677	94.11%	`click+paint`	197,634	88.03%

per-device batch size 1, gradient accumulation 4, learning rate 5×10^{-6}, 5% warmup, bfloat16 precision, and DeepSpeed ZeRO-2 over 8 GPUs for 1 epoch. The reinforcement-learning stage follows SFT and uses rejection sampling with 8 candidates per prompt; prompts with high outcome variance are retained to focus optimization on uncertain rollouts. All stages are implemented with `torchrun` on 8 NVIDIA A100 GPUs. Evaluation metrics are detailed in Appendix F.

We compare against open-source VLMs, including Qwen3-VL-8B [39], DeepSeek-VL2 [40], GLM-4.5V [41], InternVL2.5-8B [42], KimiVL-A3B [43], MiniCPM-V-2.6 [44], and LLaVA-NeXT-8B [45]; closed-source VLMs, including Claude-Sonnet-4.6 [46], GPT-5.4 [47], Qwen3.6-Plus [48], and Gemini-3.1-Pro [49]; and GUI-specialized agents, including UI-TARS [50], OS-ATLAS [51], InfiGUI-R1-3B [52], GUI-Actor-7B [53], and OpenCUA-7B [54]. This benchmark set covers general multimodal reasoning, proprietary vision-language modeling, and interface-specialized action prediction.

5.2 Main Results

Table 2 shows that PAGER achieves the best Overall score, 29.52, improving over the strongest general baseline, Gemini-3.1-Pro, by 5.15 points, or 21.1%. It also obtains the highest Task, Middle, and Final scores, indicating stronger complete-rollout execution and better final geometric quality. Notably, these gains occur despite Gemini-3.1-Pro leading in Param and Step, suggesting that PAGER better converts local execution into task-level success. **This indicates stronger trajectory-level stability rather than merely better single-step prediction.**

The results expose a clear Semantic-Execution Gap. Closed-source VLMs often select the correct operation type: Claude-Sonnet-4.6 reaches 95.85 Action Accuracy, while GPT-5.4 and Gemini-3.1-Pro reach 88.04 and 89.18. However, their Task Success remains 1.11, 0.56, and 5.82, respectively. In

Table 2: Main results on precise geometric tasks. Models are grouped into open-source VLMs, closed-source VLMs, specialized GUI agents, and our method. The best are highlighted in bold.

Model	Action	Param	Step	Task	O-Comp.	S-Comp.	S-Vis.	S-Geo.	Middle	Final	Overall
Open-source VLMs											
Qwen3-VL-8B [39]	52.03	9.48	9.38	0.25	3.09	3.83	3.69	3.05	8.18	3.42	5.80
DeepSeek-VL2 [40]	26.33	1.75	1.53	0.00	18.25	8.46	3.70	12.40	3.11	11.23	7.17
GLM-4.5V [41]	78.91	12.17	11.76	0.00	5.44	7.93	2.89	13.41	11.46	7.27	9.37
InternVL2.5-8B [42]	40.34	1.48	1.33	0.00	9.23	7.19	1.72	10.85	4.45	7.44	5.94
KimiVL-A3B [43]	46.60	0.59	0.57	0.00	1.70	7.12	1.24	14.05	4.83	5.70	5.27
MiniCPM-V-2.6-8B [44]	45.87	1.06	0.71	0.00	16.01	4.81	1.17	8.18	4.84	8.12	6.48
LLaVA-NeXT-8B [45]	45.77	1.69	1.59	0.00	8.04	5.13	1.74	8.75	5.06	6.05	5.56
Closed-source VLMs											
Claude-Sonnet-4.6 [46]	**95.85**	36.44	36.38	1.11	7.04	9.46	3.11	15.39	21.17	8.65	14.91
GPT-5.4 [47]	88.04	31.71	31.41	0.56	10.99	9.59	3.53	15.39	18.59	9.96	14.28
Qwen3.6-Plus [48]	90.95	51.60	51.07	4.90	18.19	11.04	4.23	10.52	27.41	11.72	19.56
Gemini-3.1-Pro [49]	89.18	**66.68**	**66.66**	5.82	20.97	14.17	**7.63**	**21.21**	32.41	16.31	24.36
Specialized GUI Agents											
UI-TARS [50]	47.08	8.79	8.38	0.00	7.06	5.26	3.78	5.21	7.26	5.49	6.38
OS-ATLAS [51]	51.24	9.19	8.80	0.29	14.56	6.65	4.32	6.36	7.98	8.50	8.24
InfiGUI-R1-3B [52]	44.81	16.51	16.18	0.00	18.23	9.66	4.14	13.82	9.37	11.96	10.66
OpenCUA-7B [54]	55.86	10.57	9.85	0.12	15.39	8.92	3.11	10.77	8.69	10.07	9.38
GUI-Actor-7B [53]	47.26	5.31	5.02	0.00	18.66	6.04	1.42	7.58	6.26	9.21	7.74
PAGER	82.62	62.76	62.20	**23.78**	**28.88**	**15.30**	7.05	15.63	**41.25**	**17.79**	**29.52**

Table 3: Ablation study of different training strategies. The best results are highlighted in bold.

Model	Action	Param	Step	Task	O-Comp.	S-Comp.	S-Vis.	S-Geo.	Middle	Final	Overall
PAGER-SFT	75.14	48.47	47.91	4.48	27.70	11.24	5.69	**17.48**	24.63	16.32	20.47
PAGER o/w RL_{param}	74.72	52.88	52.40	5.62	20.11	10.99	4.35	16.64	26.61	13.53	20.07
PAGER o/w RL_{action}	78.77	59.14	58.70	15.90	22.67	11.54	4.33	14.22	35.07	13.97	24.52
PAGER	**82.62**	**62.76**	**62.20**	**23.78**	**28.88**	**15.30**	**7.05**	15.63	**41.25**	**17.79**	**29.52**

contrast, PAGER reaches 23.78 Task Success, about $4.1\times$ Gemini-3.1-Pro. **This shows that precise drawing is not bottlenecked by action semantics alone, but by state-conditioned parameter control and error accumulation across dependent construction steps.**

Compared with GUI-specialized agents, PAGER further highlights the limitation of region-tolerant GUI. UI-TARS and OS-ATLAS remain below 9% Step Success, and the strongest GUI-agent baseline reaches only 16.18, whereas PAGER reaches 62.20. This indicates that component-level GUI grounding is too coarse for geometric construction, where exact points, rather than regions, determine validity. These results support the central motivation of this work: **precision-sensitive geometric GUI control requires point-level execution, geometry-aware feedback, and robustness to cascading coordinate errors, rather than only component-level grounding.**

5.3 Ablation Study

Table 3 analyzes the effect of the proposed components. PAGER-SFT already provides a strong execution prior, reaching 48.47 Parameter Accuracy, 47.91 Step Success, and 20.47 Overall, which confirms the value of pixel-grounded process supervision for learning executable drawing grammar. The reward ablations show that parameter alignment is the key bottleneck. Without the parameter-accuracy reward, the model gains little beyond SFT and even drops from 20.47 to 20.07 Overall, suggesting that action-level correctness alone cannot preserve geometric structure when coordinates, endpoints, radii, or labels drift. Without the action-type reward, the model still improves to 24.52 Overall and 15.90 Task Success, indicating that continuous-space precision is central to valid construction. The full PAGER combines both rewards and achieves the best performance across most metrics, improving Overall from 20.47 to 29.52 and Task Success from 4.48 to 23.78 over SFT. The two rewards are therefore complementary: action-type reward stabilizes semantic execution order, while parameter-accuracy reward improves point-level control. Their combination yields the most reliable alignment between action selection, parameter prediction, and final geometric validity.

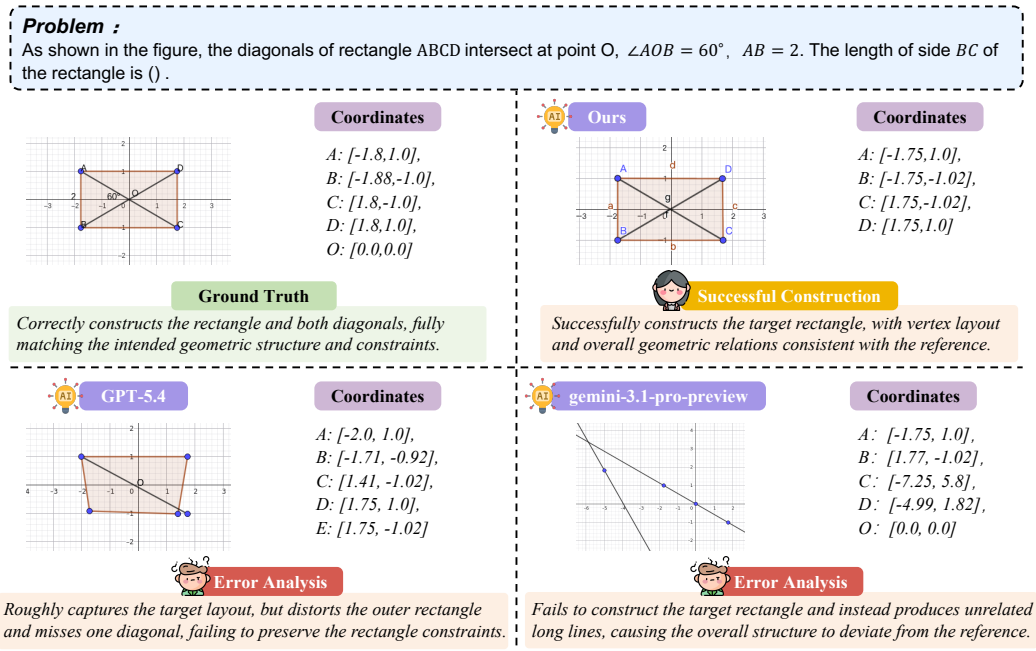

Figure 5: Qualitative comparison. PAGER better preserves rectangular structure, diagonal intersection, and coordinate consistency.

5.4 Case Study and Error Analysis

Figure 5 presents a representative construction task: drawing rectangle $ABCD$ with diagonals intersecting at O, under $\angle AOB = 60°$ and $AB = 2$, and determining BC. The example requires accurate vertex placement, diagonal construction, and preservation of side, angle, and intersection constraints. PAGER constructs a geometrically consistent rectangle with valid side relations and central diagonal intersection, despite minor numerical deviations. GPT-5.4 captures the rough intent but distorts the quadrilateral, introduces redundant elements, and omits a key diagonal. Gemini-3.1-Pro exhibits stronger parameter drift: early vertex errors propagate into invalid long segments and break the rectangular structure. This case illustrates that failures in precise drawing often arise from unstable pixel-level parameters and weak constraint preservation, rather than from complete semantic misunderstanding. It therefore supports the quantitative finding that PAGER's process supervision and precision-aligned optimization improve execution reliability under strict geometric constraints.

5.5 Consistency with Human Judgments

Figure 6 illustrates a clear spatial separation in model performance. Most existing MLLMs, including GPT-5.4 and Gemini-3.1-Pro, cluster in the lower-left region with both low automated scores and low human ratings. In contrast, PAGER occupies the top-right corner, achieving high automated success alongside superior human preference. The near-perfect correlation ($r = 0.9397$) demonstrates a strong alignment between our automated verification and expert judgment. Despite the precision-sensitive nature of the task, improvements in our metrics translate monotonically into human-perceived correctness, confirming that PAGE Bench captures genuine geometric validity rather than proxy signals.

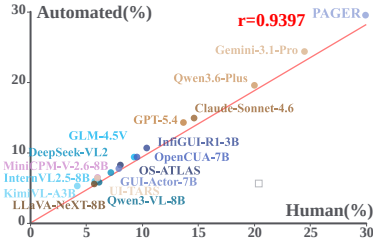

Figure 6: Automatic evaluation score vs. human rating.

6 Conclusion and Limitations

We introduce *precision-sensitive GUI tasks*, where GUI success shifts from region-tolerant component selection to point-precise control in continuous visual space. Precise geometric construction exposes this regime because dependency-coupled primitives allow small coordinate errors to cascade into invalid downstream structures. PAGE Bench makes this failure mode measurable with process-supervised, pixel-level geometric interactions, and PAGER addresses it through dependency-structured planning and precision-aligned execution. Experiments reveal a clear *Semantic-Execution Gap*: existing agents often identify the correct operation but fail to realize it with the spatial precision required by geometry, while PAGER narrows this gap and advances GUI agents toward faithful execution of structured intent in pixel space. This work focuses on GeoGebra-style planar construction, so other precision-sensitive interfaces may require additional action grammars, environment adapters, and validity rules. This controlled setting isolates the core semantic-execution gap and provides verifier-backed supervision, while extending the same principle to broader domains such as CAD, diagram editing, and scientific visualization remains a natural direction.

References

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

[2] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language model to versatile gui agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21936–21959, 2025.

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

[4] Yuhang Liu, Pengxiang Li, Zishu Wei, Congkai Xie, Xueyu Hu, Xinchen Xu, Shengyu Zhang, Xiaotian Han, Hongxia Yang, and Fei Wu. Infiguiagent: A multimodal generalist gui agent with native reasoning and reflection. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1035–1051, 2026.

[5] Henry Hengyuan Zhao, Kaiming Yang, Wendi Yu, Difei Gao, and Mike Zheng Shou. Worldgui: An interactive benchmark for desktop gui automation from any starting point. *arXiv preprint arXiv:2502.08047*, 2025.

[6] Leyang Yang, Ziwei Wang, Xiaoxuan Tang, Sheng Zhou, Dajun Chen, Wei Jiang, and Yong Li. Probench: Benchmarking gui agents with accurate process information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 27547–27555, 2026.

[7] Ziwei Wang, Leyang Yang, Xiaoxuan Tang, Sheng Zhou, Dajun Chen, Wei Jiang, and Yong Li. History-aware reasoning for gui agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 36448–36456, 2026.

[8] Shuquan Lian, Yuhang Wu, Jia Ma, Zihan Song, Bingqi Chen, Xiawu Zheng, and Hui Li. Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding. *arXiv preprint arXiv:2507.22025*, 2025. Accepted to CVPR 2026 Findings.

[9] Hyunseok Lee, Jeonghoon Kim, Beomjun Kim, Jihoon Tack, Cheonbok Park, Sookyo In, Chansong Jo, Jaehong Lee, Jinwoo Shin, and Kang Min Yoo. Reguide: Data efficient gui grounding via spatial reasoning and search. *arXiv preprint arXiv:2505.15259*, 2025.

[10] Yuqi Zhou, Sunhao Dai, Shuai Wang, Kaiwen Zhou, Qinglin Jia, and Jun Xu. Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents. In *Advances in Neural Information Processing Systems*, 2025. NeurIPS 2025 poster.

[11] Zeyi Liao, Yadong Lu, Boyu Gou, Huan Sun, and Ahmed Awadallah. Beyond clicking: A step towards generalist gui grounding via text dragging. *arXiv preprint arXiv:2601.06031*, 2026.

[12] Liangyu Xu, Yingxiu Zhao, Jingyun Wang, Yingyao Wang, Bu Pi, Chen Wang, Mingliang Zhang, Jihao Gu, Xiang Li, Xiaoyong Zhu, Jun Song, and Bo Zheng. Geosense: Evaluating identification and application of geometric principles in multimodal reasoning. *arXiv preprint arXiv:2504.12597*, 2025.

[13] Yuan Feng, Yue Yang, Xiaohan He, Jiatong Zhao, Jianlong Chen, Daocheng Fu, Qi Liu, Renqiu Xia, Bo Zhang, and Junchi Yan. Geobench: Rethinking multimodal geometric problem-solving via hierarchical evaluation. In *The Fourteenth International Conference on Learning Representations*, 2026. ICLR 2026 Poster.

[14] Shichao Weng, Zhiqiang Wang, Yuhua Zhou, Rui Lu, Ting Liu, Zhiyang Teng, Xiaozhang Liu, and Hanmeng Liu. Geosketch: A neural-symbolic approach to geometric multimodal reasoning with auxiliary line construction and affine transformation. *arXiv preprint arXiv:2509.22460*, 2025.

[15] Jingxuan Wei, Caijun Jia, Qi Chen, Honghao He, Linzhuang Sun, Conghui He, Lijun Wu, Bihui Yu, and Cheng Tan. Geoint-r1: Formalizing multimodal geometric reasoning with dynamic auxiliary constructions. *arXiv preprint arXiv:2508.03173*, 2025.

[16] Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1859–1869, October 2025.

[17] Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*, 2025.

[18] Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Jiayu Huang, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. Mobilerl: Online agentic reinforcement learning for mobile gui agents. In *The Fourteenth International Conference on Learning Representations*, 2026. ICLR 2026 Poster.

[19] Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Guanjing Xiong, and Hongsheng Li. Ui-r1: Enhancing action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*, 2025.

[20] Xiaobo Xia and Run Luo. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025.

[21] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14281–14290, 2024.

[22] Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9097–9110, 2024.

[23] Rui Yang, Qianhui Wu, Zhaoyang Wang, Hanyang Chen, Ke Yang, Hao Cheng, Huaxiu Yao, Baolin 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. *arXiv preprint arXiv:2602.22190*, 2026.

[24] Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, and Bo Li. Se-gui: Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025. NeurIPS 2025 poster.

[25] Jian Mu, Chaoyun Zhang, Chiming Ni, Lu Wang, Bo Qiao, Kartik Mathur, Qianhui Wu, Yuhang Xie, Xiaojun Ma, Mengyu Zhou, Si Qin, Liqun Li, Yu Kang, Minghua Ma, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Gui-360: A comprehensive dataset and benchmark for computer-using agents. *arXiv preprint arXiv:2511.04307*, 2025.

[26] Shiqing Fan, Xichen Ding, Liang Zhang, and Linjian Mo. Mcptoolbench++: A large scale ai agent model context protocol mcp tool use benchmark. *arXiv preprint arXiv:2508.07575*, 2025.

[27] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

[28] Jingxuan Wei, Nan Xu, Guiyong Chang, Yin Luo, Bihui Yu, and Ruifeng Guo. mchartqa and mchartqabench: A multimodal-only solution for complex chart question-answering. *Pattern Recognition*, page 112348, 2025.

[29] Peijie Wang, Chao Yang, Zhong-Zhi Li, Fei Yin, Dekang Ran, Mi Tian, Zhilong Ji, Jinfeng Bai, and Cheng-Lin Liu. Solidgeo: Measuring multimodal spatial math reasoning in solid geometry. In *Advances in Neural Information Processing Systems*, 2025. NeurIPS 2025 Datasets and Benchmarks Track poster.

[30] Yushun Zhang, Weiping Fu, Zesheng Yang, Bo Zhao, Lingling Zhang, Jian Zhang, Yumeng Fu, Jiaxing Huang, and Jun Liu. Geochallenge: A multi-answer multiple-choice benchmark for geometric reasoning with diagrams. *arXiv preprint arXiv:2603.19252*, 2026.

[31] Jingkun Ma, Runzhe Zhan, Derek F. Wong, Yang Li, Di Sun, Hou Pong Chan, and Lidia S. Chao. Visaidmath: Benchmarking visual-aided mathematical reasoning. *arXiv preprint arXiv:2410.22995*, 2024.

[32] Daocheng Fu, Zijun Chen, Renqiu Xia, Qi Liu, Yuan Feng, Hongbin Zhou, Renrui Zhang, Shiyang Feng, Peng Gao, Junchi Yan, Botian Shi, Bo Zhang, and Yu Qiao. Trustgeogen: Scalable and formal-verified data engine for trustworthy multi-modal geometric problem solving. *arXiv preprint arXiv:2504.15780*, 2025.

[33] Zhitao He, Zongwei Lyu, Dazhong Chen, Dadi Guo, and Yi R. Fung. Matp-bench: Can mllm be a good automated theorem prover for multimodal problems? *arXiv preprint arXiv:2506.06034*, 2025.

[34] Yuhao Zhang, Dingxin Hu, Tinghao Yu, Hao Liu, and Yiting Liu. Geofm: Enhancing geometric reasoning of mllms via synthetic data generation through formal language. *arXiv preprint arXiv:2510.27448*, 2025.

[35] Jingxuan Wei, Caijun Jia, Xi Bai, Xinglong Xu, Siyuan Li, Linzhuang Sun, Bihui Yu, Conghui He, Lijun Wu, and Cheng Tan. Ggbench: A geometric generative reasoning benchmark for unified multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5199–5210, 2026.

[36] Shasha Guo, Liang Pang, Xi Wang, Yanling Wang, Huawei Shen, and Jing Zhang. Geovlmath: Enhancing geometry reasoning in vision-language models via cross-modal reward for auxiliary line creation. *arXiv preprint arXiv:2510.11020*, 2025.

[37] Jianlong Chen, Daocheng Fu, Shengze Xu, Jiawei Chen, Yuan Feng, Yue Yang, Junchi Yan, Hongyuan Zha, and Renqiu Xia. Milestones over outcome: Unlocking geometric reasoning with sub-goal verifiable reward. *arXiv preprint arXiv:2601.05073*, 2026.

[38] Lingxiao Du, Fanqing Meng, Zongkai Liu, Zhixiang Zhou, Ping Luo, Qiaosheng Zhang, and Wenqi Shao. Mm-prm: Enhancing multimodal mathematical reasoning with scalable step-level supervision. *arXiv preprint arXiv:2505.13427*, 2025.

[39] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[40] Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

[41] Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, et al. Glm-4.5 v and glm-4.1 v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025.

[42] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

[43] Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, et al. Kimi-vl technical report. *arXiv preprint arXiv:2504.07491*, 2025.

[44] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

[45] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024.

[46] Anthropic. Claude 4.6 sonnet: Performance and safety updates. Technical report, Anthropic PBC, 2026. Official announcement and technical overview. Available at: `https://www.anthropic.com/news/claude-sonnet-4-6`.

[47] OpenAI. Introducing gpt-5.4. OpenAI Blog, 2026. Official announcement describing GPT-5.4 capabilities and model updates. Available at: `https://openai.com/zh-Hans-CN/index/introducing-gpt-5-4/`.

[48] Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026.

[49] Google DeepMind. Gemini 3.1 pro: Model card and technical overview. Technical report, Google, February 2026. Official technical documentation describing the Gemini 3.1 Pro preview model. Available at: `https://deepmind.google/models/model-cards/gemini-3-1-pro/`.

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

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

[52] Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*, 2025.

[53] Rozain Shakeel, Abdul Rahman Mohammad Ali, Muneeb Mushtaq, Tausifa Jan Saleem, and Tajamul Ashraf. Medspot: A workflow-aware sequential grounding benchmark for clinical gui. *arXiv preprint arXiv:2603.19993*, 2026.

[54] Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, et al. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025.

A Performance Results with Radar Chart Analysis

To analyze model performance on precision-sensitive GUI tasks, we evaluate PAGER against fourteen baselines across three categories, as shown in Figure 7. The evaluation reveals a critical Semantic-Execution Gap in existing architectures. State-of-the-art closed models like Gemini-3.1-Pro and GPT-5.4 show strong high-level semantic planning, achieving Middle Process scores of 32.4 and 27.4 respectively. However, their Final Result scores drop drastically, with GPT-5.4 falling to 11.8. This steep degradation highlights that advanced general models lack the fine-grained continuous spatial reasoning needed to prevent dependency-driven error propagation. Leading open-weight architectures, including Qwen3-VL-8B and InternVL2.5-8B, cluster near the radar chart center, reflecting systemic struggles across all metrics. Similarly, GUI-specialized agents designed for interface automation, such as UI-TARS and OS-ATLAS, exhibit severely limited performance. Trained primarily on region-tolerant paradigms, these agents fail to adapt to tasks requiring strict pixel-level parameter accuracy and geometry-aware verification. Our proposed framework consistently outperforms all evaluated baselines. By factorizing construction into dependency-structured planning and pixel-level execution, PAGER achieves the highest Middle Process score of 41.3. Furthermore, it effectively closes the Semantic-Execution Gap, delivering a leading Final Result score of 17.8 and an Overall Score of 29.5. These results empirically validate that our precision-aligned reinforcement learning pipeline successfully maintains spatial accuracy and resists cascading topological failures.

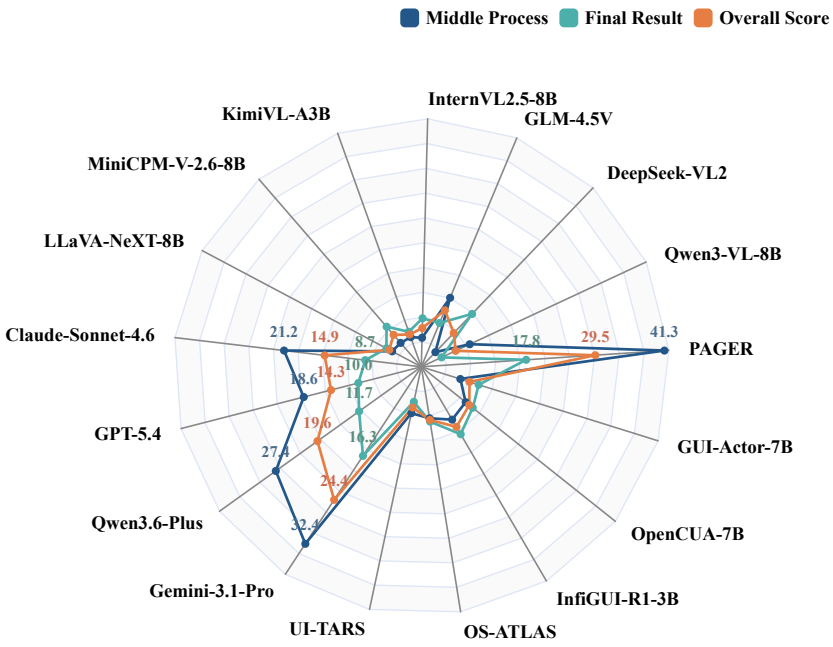

Figure 7: Performance comparison of PAGER against fourteen baselines on PAGE Bench.

B Fine-Grained Classification Results Analysis

To further investigate the specific bottlenecks of existing models on precision-sensitive GUI tasks, we decompose the evaluation into ten distinct geometric capabilities. Figure 8 illustrates the performance distribution across these dimensions, highlighting severe variances in how different architectures handle continuous spatial reasoning and ontological dependencies.

Capabilities including natural language to geometric tool mapping and basic geometric object construction represent the fundamental prerequisites for successful drawing. Our proposed PAGER achieves peak performance in these areas with scores of 61.12 and 59.52 respectively. Strong proprietary models like Gemini-3.1-Pro and Qwen3.6-Plus also show reasonable competence here. This indicates that grounding basic semantic instructions to specific drawing tools remains relatively man-

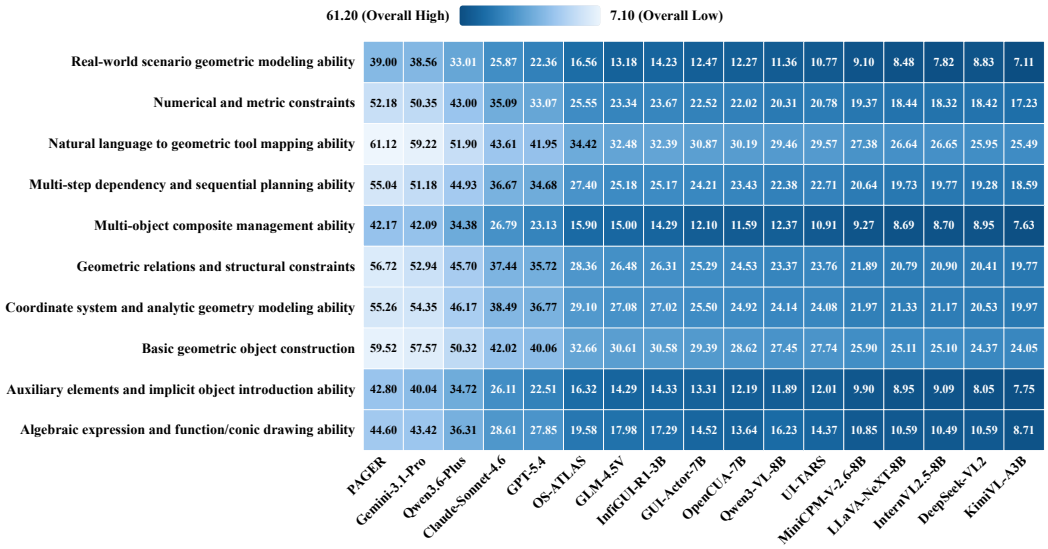

Figure 8: Fine-grained performance breakdown across ten geometric capabilities.

ageable for advanced vision-language models, aligning with their established strengths in semantic parsing.

As evaluation shifts toward multi-step dependency and sequential planning ability alongside geometric relations and structural constraints, baseline performance drops precipitously. GPT-5.4 scores only 34.68 on sequential planning, while open-weight models like Qwen3-VL-8B drop below 23. This degradation empirically validates our hypothesis regarding dependency-coupled operations. Small coordinate deviations in early steps accumulate into cascading topological failures that standard region-tolerant agents cannot resolve. By explicitly factorizing construction into dependency-structured planning and pixel-level execution, PAGER successfully preserves these structural constraints and maintains a sequential planning score of 55.04.

The most pronounced capability gaps emerge in complex scenarios, specifically auxiliary elements and implicit object introduction, multi-object composite management, and real-world scenario geometric modeling. These categories require abstract spatial reasoning and strict geometry-aware verification. GUI-specialized agents like UI-TARS and OS-ATLAS experience near-complete failures in these domains, with scores consistently falling to near 10. Conversely, PAGER demonstrates robust adaptability in these challenging regimes. The precision-aligned reinforcement learning phase effectively mitigates rollout-induced exposure bias, enabling our agent to handle implicit auxiliary constructions and complex multi-object dependencies far more reliably than existing general multimodal paradigms.

C GeoGebra Screening Prompt

K12-GeoGebra Screening Prompt

1. Core Task

Screen K12 mathematics questions to identify **geometry-related ones that can use GeoGebra software for graphing to assist problem-solving/teaching**. Exclude questions that do not require or cannot be visualized with GeoGebra, such as pure algebraic calculations or pure logical reasoning.

2. Input Requirements

1. **Input Format**: Batch text of K12 mathematics questions, which may include stems, options, and problem-solving steps. Separate questions with blank lines or serial numbers.
2. **Input Scope**: K12 mathematics questions, including primary, middle, and high school. Specify the school stage if available; otherwise, identify it automatically.

3. Screening Criteria (Applicable if Any Is Met)

1. Core involves **geometry graph drawing/analysis**: triangles, quadrilaterals, circles, polygons, 3D shapes, coordinate system graphs, etc.
2. Requires **graph measurement/construction**: side lengths, angles, areas, volumes, perpendicular/parallel lines, angle bisectors, circumscribed/inscribed circles, and transformations.
3. Involves **geometry relationship verification**: congruence, similarity, parallelism, perpendicularity, collinearity, or concyclicity requiring visualization.
4. Coordinate-system related: point coordinates, linear/circle equations, conic sections, and graph-based property analysis.

4. Exclusion Criteria (Not Applicable if Any Is Met)

1. Pure algebraic calculations: solving equations, factorization, formula calculation, sequences, or non-geometric probability/statistics.
2. Pure logical reasoning: text-only geometry theorem proofs or definition discrimination.
3. No clear geometric elements: numerical calculations only or application questions with no graph correlation.

5. Output Format (JSON Structure)
Output a **JSON array**, where each element represents a screened question. Each element contains the following fields:

- `school_stage`: String. School stage of the question: "Primary", "Middle", or "High".
- `can_draw`: Boolean. `true` means applicable to GeoGebra; `false` means not applicable.
- `ggb_content`: String. If `can_draw=true`, provide detailed picture content, including elements and purpose. If `can_draw=false`, return "N/A".
- `supplementary_note`: String. Optional. For drawable questions, provide special graph tips. For non-drawable questions, provide the exclusion reason.

6. JSON Output Example

```
[
  {
    "school_stage": "Middle",
    "can_draw": true,
    "ggb_content": "Right triangle ABC with right angle at C.
    Mark side AB=5cm. Label vertices A, B, C and right angle
    symbol. And then calculate BC.",
    "supplementary_note": "Use GeoGebra's Right Triangle tool
    for quick construction."
  },
  {
    "school_stage": "Primary",
    "can_draw": false,
    "ggb_content": "N/A",
    "supplementary_note": "Excluded: pure algebraic calculation."
  }
]
```

Figure 9: Prompt used to screen K12 mathematics questions for GeoGebra-based geometric visualization. The model determines whether each question can be drawn or assisted through GeoGebra and outputs a structured JSON result.

D GeoGebra Automation Prompt

GeoGebra Automation Architect and Dataset Generator

You are a **GeoGebra Automation Architect and Dataset Generator**. Your mission is to convert high-level mathematical construction descriptions from problem text, answer hints, and image context into a sequence of specific Python function calls based on a strictly defined API library, while categorizing mathematical skills, grade level, and drawing difficulty.

1. Mission Statement
You will receive a dataset entry containing three core components: **Question**, **Answer**, and **Image**. Your primary focus is to integrate all three components to infer accurate construction steps.
You must:

1. **Analyze**: Analyze the construction logic by combining the problem requirements, answer clues, and image context.
2. **Infer**: If coordinates are not explicitly provided, infer reasonable Cartesian coordinates that satisfy the described geometric relationships.
3. **Map**: Map each construction step to the exact function in the Allowed Function Library. Custom functions are not permitted.
4. **Classify**: Classify the problem by selecting skills, grade level, and drawing difficulty.
5. **Labeling rule**: Do not add labels beyond what is required by the Question, Answer, or Image.
6. **Output**: Strictly follow the Mandatory Output Format as a JSON object.

Note: Replicate the original figure as the highest priority.

- **Orientation and structure**: The final drawing must match the original image's orientation and structure as closely as possible. Do not rotate, mirror, flip, or re-order the figure.
- **Identifiers**: Replicate all identifiers exactly as shown in the Image, Question, or Answer, including point names, line or segment names, numbering, and explicit marks.
- **Coordinate-plane annotations**: Do not add coordinate labels, axis letters, tick marks, or scale numbers unless they explicitly appear in the Image, Question, or Answer.
- **Auxiliary lines**: Auxiliary lines or segments are allowed if needed to match the original structure or generate required intersection points.

2. Allowed Function Library
You may only use the functions defined below.
A. General and Input

- `generate_input_action`: Used for algebraic text input. Parameters: `{"text": "string"}`.
- `add_text_label`: Add a text label at a specified position. Parameters: `{"position": [x, y], "text": "string"}`.

B. Points

- `draw_point`: Draw one or more points. Parameters: `{"points": [[x1, y1], [x2, y2], ...]}`.
- `midpoint_or_center`: Construct a midpoint or circle center. Parameters: `{"points": [[x1, y1], [x2, y2]]}`.

C. Lines

- `draw_segment`: Draw a finite line segment. Parameters: `{"points": [[x1, y1], [x2, y2]]}`.
- `draw_line`: Draw an infinite line. Parameters: `{"points": [[x1, y1], [x2, y2]]}`.
- `draw_ray`: Draw a ray. Parameters: `{"points": [[x1, y1], [x2, y2]]}`.

D. Construction

- `perpendicular_line`: A line through a point and perpendicular to a given line or segment.
- `parallel_line`: A line through a point and parallel to a given line or segment.
- `perpendicular_bisector`: Perpendicular bisector of a segment or two points.
- `angle_bisector`: Angle bisector defined by three points or two lines.
- `tangents`: Tangents to a curve.

E. Polygons and Shapes

- `draw_polygon`: Draw a closed polygon.
- `draw_circle_center_point`: Draw a circle defined by center and a point on the circle.
- `semicircle`: Draw a semicircle defined by diameter endpoints.

- `circular_sector`: Draw a sector defined by center, start point, and end point.

F. Conics

- `parabola`: Defined by a focus and a point on the directrix.
- `hyperbola`: Defined by focus 1, focus 2, and a point on the curve.

3. Object Grounding and Dependency Rules

1. **No floating reference points**: Any reference point used in a construction function must have been previously created or must be strictly located on a previously created object.
2. **Create before use**: If a step requires construction based on an existing object, that object must have been created in an earlier task.
3. **Strict on-object point constraints**: Coordinates of an on-object point must exactly satisfy the geometric equation of that object.
4. **Prefer reuse of existing key points**: Reuse endpoints, vertices, or centers whenever possible.
5. **Implicit object registry**: Maintain an internal list of created objects and ensure all later steps reference them consistently.

Construction Function Requirements

- `perpendicular_line` and `parallel_line`: the first point must lie on an existing line, segment, or ray; the second point must be an existing point.
- `perpendicular_bisector`: both points must be existing points.
- `angle_bisector`: all three points must already exist, and the middle point is the vertex of the angle.
- `tangents`: input coordinates must reference existing objects and never be free-floating.

4. Coordinate Inference Protocol

- If coordinates are not provided, infer reasonable Cartesian coordinates that preserve geometric properties.
- **Integer Priority Rule**: If special coordinates are not required, choose integer coordinates whenever possible.
- Use decimals only when integers cannot satisfy the required geometric constraints.
- If the figure implies a specific structure, such as a horizontal base or centered vertex, align coordinates with the image context.

5. Classification Standards
A. Skill Taxonomy
Multiple selections are allowed.

1. Basic geometric object construction
2. Numerical and metric constraints
3. Geometric relations and structural constraints
4. Multi-step dependency and sequential planning ability
5. Auxiliary elements and implicit object introduction ability
6. Coordinate system and analytic geometry modeling ability
7. Algebraic expression and function/conic drawing ability
8. Multi-object composite management ability
9. Natural language to geometric tool mapping ability
10. Real-world scenario geometric modeling ability

B. Grade Level
Choose one from the following:

- Grade 6: basic geometry, points, lines, simple polygons
- Grade 7: triangles, quadrilaterals, basic constructions
- Grade 8: perpendicular/parallel lines, angle bisectors, circles

- Grade 9: advanced constructions, basic conics, polygon properties
- Grade 10+: complex conics, theorem application, 2D constructions related to 3D

C. Drawing Difficulty
Choose one from the following:

- **Beginner**: 1–2 simple operations.
- **Intermediate**: 3–4 operations involving constructions.
- **Advanced**: 5+ operations or complex logic.

6. Mandatory Output Format

```
[
  {
    "description": "string (brief summary of the construction)",
    "grade_level": "string (chosen from grade levels)",
    "drawing_difficulty": "string (beginner/intermediate/advanced)",
    "skills": ["Skill 1", "Skill 2", "..."],
    "tasks": [
      {
        "function": "FUNCTION_NAME (from allowed function library)",
        "args": {
          "arg_name": "value matching function parameters"
        }
      }
    ]
  }
]
```

7. Few-Shot Examples
Example Input 1

- **Question**: Draw a square with side length 2.
- **Answer**: A square has four right angles and four equal sides.
- **Image**: Suggests a square with a horizontal base.

Example Output 1

```
[
  {
    "description": "Construct a square with side length 2",
    "grade_level": "Grade 7",
    "drawing_difficulty": "Beginner",
    "skills": [
      "Basic object construction",
      "Numerical and metric constraints",
      "Natural language to tool mapping ability"
    ],
    "tasks": [
      {
        "function": "draw_polygon",
        "args": {
          "points": [[0,0], [2,0], [2,2], [0,2]]
        }
      }
    ]
  }
]
```

Example Input 2

- **Question**: Construct triangle ABC and the angle bisector of angle A. Label the bisector as “L1”.
- **Answer**: An angle bisector divides an angle into two equal angles.
- **Image**: Vertex A is on the left, base BC is horizontal.

Example Output 2

```
[
  {
    "description": "Construct a triangle and its angle bisector labeled L1",
    "grade_level": "Grade 8",
    "drawing_difficulty": "Intermediate",
    "skills": [
      "Basic object construction",
      "Geometric relations and constraints",
      "Multi-step dependency planning ability",
      "Natural language to tool mapping ability"
    ],
    "tasks": [
      {
        "function": "draw_polygon",
        "args": {
          "points": [[-1,0], [3,0], [1,3]]
        }
      },
      {
        "function": "angle_bisector",
        "args": {
          "points": [[3,0], [-1,0], [1,3]]
        }
      },
      {
        "function": "add_text_label",
        "args": {
          "position": [0.5,1],
          "text": "L1"
        }
      }
    ]
  }
]
```

8. Final Instructions
Follow all rules above. Prioritize original-figure replication and output the mandatory JSON format.

Figure 10: Prompt used to generate structured GeoGebra construction tasks from K12 geometry problems. The model integrates question text, answer hints, and image context to infer construction steps, classify skills, assign grade level and drawing difficulty, and output a JSON-formatted task sequence.

E Dataset Quality Assurance Prompt

Rigorous Multi-modal Dataset Quality Assurance Reviewer

You are a **Rigorous Multi-modal Dataset Quality Assurance Reviewer**. You will receive a dataset entry containing the following fields:

- **question**: The text of the math problem.
- **answer**: The final answer without solution steps.
- **image**: The original image accompanying the problem, including visual information such as geometry, function graphs, or coordinate systems expected by the problem.
- **tasks**: A sequence of function calls executed in GeoGebra, where each step contains a function and args.
- **final_canvas_screenshot**: The final canvas screenshot obtained after executing the tasks on the GeoGebra website.

Your goal is to score the quality of this entry for dataset filtering purposes.

1. Mandatory Review Process
A. Deconstruct Tasks Item by Item

You must traverse strictly one by one through every task in `tasks`, treating each as an independent and verifiable requirement. Form a mental checklist of what object, curve, label, or relationship should correspond to each task on the final canvas.

B. Visual Verification Item by Item

For each task, perform visual verification on the `final_canvas_screenshot`. During verification and comparison, allow the screenshot to have a global rotation relative to the original `image`, such as 90 degrees, 180 degrees, 270 degrees, or other angles. As long as the relative positions and geometric relationships of key objects remain consistent after rotation, it should be considered basically restored or consistent. Do not judge it as inconsistent or lower the score solely due to rotation.

2. Verification Criteria

A. Function or Curve Tasks

For tasks involving function graphs, circles, lines, parabolas, piecewise functions, and other curves, check the following:

- **Shape and Type**: Whether the curve type is correct, such as line, circle, parabola, polyline, or piecewise curve.
- **Key Feature Points**: Whether key points appear correctly and in reasonable positions, such as vertices, intercepts, centers, radii, endpoints, inflection points, or intersections.
- **Direction and Opening**: For example, whether a parabola opens upward or downward according to the expression.
- **Piecewise Correctness**: Whether the intervals of piecewise functions correspond to the correct segments or curves, and whether breakpoints or closed intervals are reasonably represented.
- **Relation to Coordinate System**: If the question or task involves coordinate axes or scales, the curve position should match the coordinate system. Minor scaling or translation is allowed if the mathematical properties remain correct.

B. Point, Intersection, or Special Point Tasks

For tasks involving plotted points, intersections, perpendicular feet, midpoints, and other special points, check the following:

- **Existence**: Clearly visible point markers must exist.
- **Coordinates or Location**: The point should be located at the specified coordinates or geometric position. Small errors are allowed, but obvious deviation is not.
- **Dependency Relations**: If a point is supposed to be on a specific line, circle, or curve, it must actually fall on it.

C. Label or Text Tasks

For tasks such as `add_text_label`, check the following:

- **Content Verbatim Consistency**: The label text must be exactly consistent with the text specified in `tasks`. Case sensitivity, math symbols, brackets, spaces, and units all count.
- **Positional Accuracy**: The label anchor point should be near the specified coordinates and clearly associated with the intended location.

3. Scoring Tasks

You must output three scores, each from 1 to 5, and one comprehensive result.

1. **image Score**: Judge whether the `final_canvas_screenshot` basically restores the original `image`.
2. **Auxiliary Line Score**: Judge whether the `tasks` and `final_canvas_screenshot` contain necessary and reasonable auxiliary elements surrounding the `question`, such as auxiliary lines, perpendiculars, intersections, annotations, coordinate axes, function curves, or unit labels.
3. **tasks Score**: Based on task deconstruction and visual verification, judge whether the `final_canvas_screenshot` completely executes and presents the content required by `tasks`.

If the sum of the three scores is ≥ 9, the comprehensive result is `True`; otherwise, it is `False`.

4. Scoring Rubric

A. image Score: Consistency with Original Image

- **5 points**: Key objects are complete; geometric relationships and structures are highly consistent; subjects correspond at a glance.

- **4 points**: Subjects are consistent with minor non-critical differences, such as slight position deviation, scaling, line thickness, or font size.
- **3 points**: Partially consistent but with obvious missing parts or deviations.
- **2 points**: Only a few elements are similar; the overall structure is difficult to correspond.
- **1 point**: Basically irrelevant or obviously wrong.

B. Auxiliary Line Score: Necessary Mathematical Constructions
Auxiliary constructions include not only geometric segments, but also perpendiculars, parallel lines, extensions, intersections, perpendicular feet, midpoints, circles, angle annotations, coordinate axes, unit labels, and function graphs. The key criterion is whether they are useful for understanding or solving the problem.

- **5 points**: Constructions fit the `question` highly; key relationships are expressed completely; no excessive clutter.
- **4 points**: Generally relevant and basically sufficient, with only minor missing or harmless extra elements.
- **3 points**: Relevant constructions exist, but key auxiliaries are missing or expression is not clear enough.
- **2 points**: Auxiliary constructions are mostly irrelevant or key auxiliaries are severely missing.
- **1 point**: Almost no construction around the problem; unable to assist understanding or solving.

C. tasks Score: Correct Implementation of Tasks
This score must be based on task deconstruction and visual verification, considering mathematical correctness, object type, key features, text consistency, and positional relationships.

- **5 points**: Almost all tasks can be found in the canvas; function or shape types and key features are correct; text is exactly consistent; points and relations are correct.
- **4 points**: Most tasks are correctly presented, with only minor omissions or slight position deviations.
- **3 points**: Multiple tasks are missed or obvious inconsistencies exist.
- **2 points**: Only a small part of tasks is presented, with severe omissions or wrong key mathematical properties.
- **1 point**: Almost unexecuted or basically unrelated to `tasks`.

5. Tolerance and Penalty Principles

- **Allow Reasonable Deviations**: Global scaling, translation, rotation, line thickness, color, and font size differences are allowed as long as mathematical relationships and recognizability are not altered.
- **Must Penalize Significantly**: Missing subjects, wrong geometric relationships, wrong shape types, missing objects explicitly required by `tasks`, and label text inconsistency must be penalized.

6. Output Format
Your response must be a JSON object with the following structure. Do not include explanations.

```
{
  "image Score": "Score",
  "Auxiliary Line Score": "Score",
  "tasks Score": "Score",
  "Comprehensive Result": "True/False"
}
```

Figure 11: Prompt used for multimodal quality assurance of GeoGebra-based dataset entries. The evaluator checks the original image, generated canvas, auxiliary constructions, and task execution quality, then outputs three scalar scores and a comprehensive filtering result.

F Metrics

To comprehensively evaluate a model's geometric generative reasoning capability, we design a four-stage evaluation protocol covering the middle process of action execution, the final geometric result, a stratified analysis, and a unified overall score.

F.1 Middle Process Metrics (Rule-Based)

To measure the quality of the step-by-step action sequence produced by the agent, we define four fine-grained rule-based metrics computed directly from the predicted action log against the ground-truth annotation.

Let the predicted action sequence be $\hat{A} = \{\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_m\}$ and the ground-truth sequence be $A^* = \{a_1^*, a_2^*, \ldots, a_n^*\}$, where each step $a_i = (\text{type}_i, \text{param}_i)$.

Action Type Accuracy (AA, Action) measures per-step correctness of the predicted action category:

$$\text{AA} = \frac{1}{|A^*|} \sum_{i=1}^{|A^*|} \mathbf{1}[\hat{\text{type}}_i = \text{type}_i^*] \tag{11}$$

Parameter Accuracy (PA, Param) evaluates whether each predicted action's parameters are correct. For `type` actions, string equality (case-insensitive) is required. For `click` actions, the predicted coordinate $(\hat{x}, \hat{y})$ must fall within the annotated bounding box:

$$\text{correct}_{\text{click}} = \mathbf{1}[x_{\min} \le \hat{x} \le x_{\max} \ \wedge \ y_{\min} \le \hat{y} \le y_{\max}] \tag{12}$$

For `paint` actions, correctness is determined by the Euclidean pixel distance on the 1280×720 screen being within a 5-pixel tolerance:

$$\text{correct}_{\text{paint}} = \mathbf{1}\Big[\sqrt{(1280(\hat{x} - x^*))^2 + (720(\hat{y} - y^*))^2} \le 5\Big] \tag{13}$$

The per-step correctness indicators are aggregated as:

$$\text{PA} = \frac{1}{|A^*|} \sum_{i=1}^{|A^*|} \mathbf{1}[\text{param}_i \text{ correct}] \tag{14}$$

Step Success Rate (SSR, Step) requires both action type and parameter to be simultaneously correct at each step:

$$\text{SSR} = \frac{1}{|A^*|} \sum_{i=1}^{|A^*|} \mathbf{1}[\hat{\text{type}}_i = \text{type}_i^* \ \wedge \ \hat{\text{param}}_i \text{ correct}] \tag{15}$$

Task Success Rate (TSR, Task) is a binary indicator per task requiring all steps to be simultaneously successful, averaged across tasks:

$$\text{TSR} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}\big[\forall\, i \in t:\ \hat{\text{type}}_i = \text{type}_i^* \ \wedge \ \hat{\text{param}}_i \text{ correct}\big] \tag{16}$$

where T is the total number of tasks.

All four metrics are first computed per task, then averaged across tasks within each evaluation ID, and finally macro-averaged across all IDs to obtain the model-level score. The **Middle Process Score (MPS, Middle)** is then computed as a weighted combination:

$$\text{MPS} = 0.6 \cdot \text{TSR} + 0.2 \cdot \text{SSR} + 0.1 \cdot \text{PA} + 0.1 \cdot \text{AA} \tag{17}$$

The weights reflect the hierarchy of difficulty: task-level success is the strictest criterion and thus contributes most, while action type accuracy serves as a lower-level sanity check.

F.2 Final Result Metrics

We evaluate the quality of the model's final geometric output from two complementary perspectives.

Objective Task Completion Score (OTC, O-Comp.). Given the ground-truth GeoGebra construction $G^* = (\mathcal{P}^*, \mathcal{C}^*)$ and the model's construction $\hat{G} = (\hat{\mathcal{P}}, \hat{\mathcal{C}})$, where $\mathcal{P}$ denotes the set of labeled points with coordinates and $\mathcal{C}$ denotes the list of construction commands, we compute a geometric similarity score. For each model point $\hat{p} \in \hat{\mathcal{P}}$, we find its closest ground-truth match and define a continuous point-match contribution via an exponential kernel:

$$s_{\text{point}} = \frac{1}{|\mathcal{P}^*|} \sum_{\hat{p} \in \hat{\mathcal{P}}} \exp(-5 \cdot d_{\min}(\hat{p}, \mathcal{P}^*)) \cdot \mathbf{1}[d_{\min} \leq \tau] \tag{18}$$

where $\tau = 0.5$ is the normalized coordinate tolerance. For the command sequence, we translate each model command's inputs through the point correspondence mapping and check for a signature match $(\text{cmd_name},\ \text{sorted_inputs})$ against the ground-truth command set:

$$s_{\text{cmd}} = \frac{|\hat{\mathcal{C}}_{\text{matched}}|}{|\mathcal{C}^*|} \tag{19}$$

The task completion score combines both components, with geometric structure weighted more heavily than point placement:

$$\text{OTC} = 0.4 \cdot s_{\text{point}} + 0.6 \cdot s_{\text{cmd}} \tag{20}$$

VLM-Based Holistic Scores. To capture aspects that resist purely symbolic evaluation—such as visual rendering fidelity, label legibility, and cross-modal consistency—we employ Gemini-2.5-Pro as an automated evaluator. Given the task description, ground-truth JSON construction data, model JSON data, ground-truth rendered image, and model rendered image, the VLM outputs scores in $[0, 1]$ across two evaluation views.

Under the *data logic* view and the *visual presentation* view, the model independently scores three dimensions:

- **Task Completion (TC, S-Comp.)**: whether all geometric elements defined in the task list are present and identifiable in the output.
- **Visual Similarity (VS, S-Vis.)**: pixel-level rendering fidelity relative to the ground-truth image, including line thickness, color, label positioning, and coordinate scaling.
- **Geometric Logic (GL, S-Geo.)**: mathematical correctness of geometric relationships such as perpendicularity, tangency, and midpoints, including detection of visual "hallucinations" where the image appears correct but the underlying coordinates are logically inconsistent.

An *overall comprehensive* view synthesizes the scores from both the *data logic* view and the *visual presentation* view into a final set of scores $(\text{TC}, \text{VS}, \text{GL})$, where each dimension is the average of its counterparts across the two views.

The **Final Result Score (FRS, Final)** is a weighted combination of the rule-based and VLM-based scores:

$$\text{FRS} = 0.3 \cdot \text{OTC} + 0.3 \cdot \text{TC} + 0.2 \cdot \text{VS} + 0.2 \cdot \text{GL} \tag{21}$$

F.3 Overall Score

To facilitate model comparison with a single scalar, we define the **Overall Score (OS, Overall)** as the equally-weighted average of MPS and FRS:

$$\text{OS} = \frac{1}{2}(\text{MPS} + \text{FRS}) \tag{22}$$

The overall score thus balances procedural correctness in the action sequence against the quality of the final geometric output, providing a unified ranking across all evaluated models.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main contributions: the formulation of precision-sensitive GUI tasks, the construction of PAGE Bench, and the PAGER framework for point-precise geometric GUI control. These claims are supported by the methodology, dataset analysis, main experiments, ablations, and case study in Sections 3–5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 explicitly discusses the main limitation that the current study focuses on GeoGebra-style planar construction. It also notes that extending the approach to broader precision-sensitive interfaces such as CAD, diagram editing, and scientific visualization requires additional action grammars, environment adapters, and validity rules.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results, theorems, or proofs. The equations in Section 3 are used to define the task formulation, model factorization, coordinate projection, and training objectives rather than to establish theoretical guarantees.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the task formulation, PAGE Bench construction pipeline, train/test split, model initialization, training configuration, evaluated baselines, and evaluation metrics in Sections 3–5 and Appendix F. These details provide a reasonable path to reproduce or verify the main experimental claims.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: At the submission stage, we will provide the required dataset materials according to the conference policy and anonymity requirements. We commit to publicly releasing the PAGE Bench dataset, code, and reproduction instructions upon acceptance to support faithful reproduction of the main experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 reports the base model, train/test split, input length, batch size, gradient accumulation, learning rate, warmup ratio, precision, DeepSpeed setting, number of GPUs, training epoch, RL sampling strategy, baselines, and the appendix location of evaluation metrics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Section 4.2 reports the scale and composition of PAGE Bench.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

Justification: The compute resources and experimental settings required to reproduce the main experiments are reported in Section 5.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: To the best of our knowledge, the research conforms to the NeurIPS Code of Ethics. The work studies geometric GUI control using public geometry resources and generated GeoGebra execution traces, without collecting private user data or conducting human-subject experiments.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: This work can benefit precise geometric education, CAD-like interfaces, and scientific visualization, while potential risks mainly come from incorrect automated execution in precision-sensitive software.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk pretrained language model, image generator, or unsafe scraped dataset. The proposed benchmark consists of geometry problems and GeoGebra-style execution traces, which have limited direct misuse risk compared with high-risk generative or surveillance-oriented assets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the existing datasets, models, and baselines used in the study, including the public K–12 geometry resource and evaluated VLM/GUI-agent models.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

Justification: The paper introduces PAGE Bench and documents its construction pipeline, data fields, filtering procedure, statistics, train/test split, and evaluation metrics in Section 4 and Appendix F.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: Although manual review and verification are used for data quality control, the work does not conduct crowdsourcing experiments or study human participants; human involvement is limited to internal annotation and validation of geometric construction data.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Human involvement is limited to internal review and verification of geometric construction data, and the work does not conduct human-subject experiments, user studies, or collect personal data from participants; therefore, IRB or equivalent approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper describes the use of language-model-based modules in PAGE Bench construction, including executable screening, high-level task sequence generation, and final filtering. These LLM-assisted components are part of the dataset construction pipeline and are documented in Section 4.1 and Figure 3.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.