Solving Is Not Drawing: A Benchmark for Diagrammatic Reasoning in Olympiad Geometry

Anonymous ACL submission

Abstract

Foundation models such as GPT and Claude now solve olympiad-level mathematics with remarkable proficiency, so much so that geometry problem solving has become a standard proxy for their mathematical reasoning. Yet solving a geometry problem and drawing the figure it depends on are not the same skill: progress often hinges on a faithful diagram with the right auxiliary constructions and incidences, and it is unclear that a model which reasons its way to the answer can also produce one. A growing collection of benchmarks, including MathVista and MathVerse, measures whether models reach the correct answer, but to our knowledge, none isolate the distinct ability to construct the diagram itself, leaving this capability unmeasured. We introduce an open-source benchmark that targets this gap: 954 self-contained olympiad geometry problems, with a 297-problem hard subset, each paired with its solution and a human-authored, high-fidelity diagram in renderable Asymptote code, together with a suite of text-, code-, image-, VLM-, and constraint-based metrics for what we term *diagrammatic reasoning*. Evaluating current foundation models reveals a pronounced gap between solving and drawing: their diagrams are markedly less faithful, with an average compile success rate of only 36.14%. Strong mathematical reasoning, we find, does not imply the ability to construct accurate geometric diagrams. Our benchmark and codebase will be open-sourced upon acceptance.

1 Introduction

Recently, foundation models have achieved exceptional scores on math olympiad problems, so much so that these problems are now used as a general proxy for a model's mathematical reasoning. However, many olympiad problems rely on more than textual reasoning. Geometry is a prominent example, as progress frequently depends on an accurate, representative diagram, complete with the right auxiliary constructions and the correct incidences between points, lines, and circles. Solving the problem and drawing the figure it rests on are not obviously the same skill.

Our experiments show that they are not. Despite their textual problem-solving ability, foundation models exhibit a clear discrepancy between solve accuracy and faithful diagram generation, and this gap persists even when a model is handed the full reference solution and asked only to render a consistent figure. Since geometric construction and visualization are central to solving geometry problems, we define *diagrammatic reasoning* as a model's ability to produce a geometrically consistent, high-fidelity diagram from a problem, and treat it as a capability distinct from problem solving. Measuring it matters in practice: a model that reasons correctly but draws an inconsistent figure cannot be trusted to support geometric work, to verify its own constructions, or to feed reliable diagrams to downstream agents.

A growing collection of benchmarks evaluates the mathematical and geometric problem-solving abilities of foundation models, including OlympiadBench, MathVista, MathVerse, and We-Math. These efforts measure whether a model reaches the correct answer or follows a valid reasoning trace, but they consume diagrams as input or treat them as incidental. To our knowledge, none isolate the distinct ability to construct the diagram itself, so a model can score well while holding an inconsistent or impossible picture of the configuration.

We address this gap with an open-source benchmark of 954 self-contained olympiad geometry problems, including a 297-problem hard subset, each paired with its full solution and a human-authored, high-fidelity ground-truth diagram available as both an image and renderable code. We require models to express diagrams as Asymptote code, since its native geometric primitives

make it better suited to precise compass-and-straightedge constructions than general-purpose plotting libraries, while keeping the task in the model's native text modality and yielding figures that are precise, modifiable, and easy to compare against ground truth diagrams. That the gap persists in this code setting indicates it is not an artifact of weak multimodal generation; even when confined to text, models struggle to extract and correctly represent geometric constraints. We pair the dataset with a suite of text-, code-, image-, VLM- and constraint-based metrics for diagrammatic reasoning.

Evaluating current foundation models confirms a pronounced gap between solving and drawing. Their diagrams are markedly less faithful than their solve accuracy would suggest, with an average compile success rate of only 36.14% (Table 1). Our main contributions are as follows:

1. We define and standardize *diagrammatic reasoning*, the ability to generate a geometrically consistent, high-fidelity diagram, as a capability distinct from problem solving.
2. We release an open-source dataset of 954 olympiad geometry problems, each with a full solution and a human-made, high-fidelity diagram available as both an image and renderable Asymptote code, together with a suite of text-, code-, image-, VLM-, and constraint-based metrics for evaluation.
3. We benchmark state-of-the-art foundation models and show that strong mathematical reasoning does not imply the ability to construct accurate geometric diagrams.

2 Related Work

Geometric diagram generation. Approaches range from manual tools such as GeoGebra (Hohenwarter, 2007) to code-based generation with libraries like matplotlib (Hunter, 2007) or specialized languages such as Asymptote (Hammerlindl et al., 2026), and to formal geometry languages paired with constraint solvers (Lu et al., 2021; Wang et al., 2025; Zhang et al., 2025). The field has recently shifted toward learning-based generation (Wang et al., 2025; Zhang et al., 2025; Cheng et al., 2025): diffusion models produce high-fidelity images (Ho et al., 2020; Rombach et al., 2022) but tend to sacrifice the precision geometry demands, motivating multi-modal large language model (MLLM) approaches (Lu et al., 2024; Cheng et al., 2025). GeoUni (Cheng et al., 2025), for instance, trains a unified model with GRPO (Shao et al., 2024) and several geometry-specific rewards, but MLLM-based approaches require training a dedicated MLLM where a text- and code-based approach could suffice. Foundation models such as GPT-5.4 (OpenAI, 2026a) and Claude Sonnet 4.6 (Anthropic, 2026) can already emit matplotlib or Asymptote code that renders into a diagram without any dedicated training; whether those diagrams are geometrically faithful has not, to our knowledge, been systematically evaluated. This is the gap our work targets.

Geometry metrics and evaluation. Prior benchmarks typically measure whether a model reaches the correct answer or produces a valid reasoning trace (Chen et al., 2021; Lu et al., 2021; Chen et al., 2022; Lu et al., 2024), but not whether it conceptualizes the geometric structure correctly; a model can reach the right answer while holding an inconsistent diagrammatic representation (Zhang et al., 2024; Qiao et al., 2025). Comparing a generated diagram against a ground-truth construction instead requires fidelity metrics. For images, learned perceptual distances such as LPIPS (Zhang et al., 2018) capture visual similarity; for code, text-overlap metrics such as BLEU (Papineni et al., 2002) and chrF++ (Popović, 2017) count shared n-grams and CodeBLEU (Ren et al., 2020) adds syntactic and semantic signals, though all compare token distributions rather than geometric behavior. Using an LLM or vision-language model as a judge (Zheng et al., 2023; Chen et al., 2024) assesses geometric reasoning more directly, which we adopt alongside the metrics above.

Geometry datasets. Existing datasets focus on problem solving rather than diagram fidelity. GeoQA (Chen et al., 2021) and Geometry3K (Lu et al., 2021) pair problems with diagrams and formal annotations for answer prediction; UniGeo (Chen et al., 2022) unifies calculation and proof; and MathVista (Lu et al., 2024), MathVerse (Zhang et al., 2024), and We-Math (Qiao et al., 2025) target general visual mathematical reasoning. Separately, scientific vector-graphics benchmarks show that code-based diagrams can serve as evaluation targets. For example, AutomaTikZ (Belouadi et al., 2024a), DeTikZify (Belouadi et al., 2024b), Text2arch (Garg et al., 2026) and vTikZ (Reux et al., 2025) all focus on general diagrams

rather than theorem-driven geometry, which we address.

3 Dataset

Problem sources. We source problems from a range of mathematical olympiad exams, primarily the American Mathematics Competition (AMC), American Invitational Mathematics Examination (AIME), USA (Junior) Mathematical Olympiad (USAMO/USAJMO) (Mathematical Association of America, 2026a,b), and the International Mathematical Olympiad (IMO) (International Mathematical Olympiad, 2026), web-scraping problems dating back to the 1980s and their online solutions (Art of Problem Solving, 2026). We keep only solutions whose diagrams are given as human-authored Asymptote code, which we treat as ground truth. We then prompt GPT-5.4-mini (OpenAI, 2026b) to, per problem, (1) decide whether it is geometric, (2) decide whether it is self-contained, and (3) summarize the reference solution's approach. This yields 954 problems, each paired with a human-authored Asymptote diagram, and a 297-problem hard subset. An overview of the curation and evaluation pipeline is shown in Figure 1, and a full dataset breakdown appears in Appendix A.

Self-containment. We call a problem self-contained if its solution is reproducible from the statement alone, and exclude any problem that depends on an input image. This keeps the benchmark accessible to text-only models and removes a source of error propagation, since a misread input diagram would corrupt a model's reasoning from the outset.

Difficulty filtering. We prompt GPT-5.4-mini to rate each problem's difficulty from 1 to 10 and retain those scoring 7 or above, yielding the hard subset of 297 problems challenging enough to be discriminative, and comparable in size to other evaluation benchmarks.

4 Methodology

Models. We evaluate Claude Sonnet 4.6, Gemini 3.1 Flash-lite, GPT-5.4-mini, Kimi K2.5, and GLM 5.1 on their ability to generate high-quality, code-based diagrams.

Experiments. We run two settings per model, differing in how much guidance the model receives toward our ground-truth solution. We initially considered providing only the problem statement, but since the problems admit many valid solution paths, a model may diverge from our ground truth and have its diagram unfairly penalized against a single reference; we therefore add a second setting in which the model is anchored to the intended construction via the full reference solution. In the with-solution setting, we provide the statement together with the full reference solution and prompt the model to render a consistent diagram, directly measuring whether it can produce a diagram faithful to a given solution. In the without-solution setting, we provide only the problem statement and prompt the model to both reason through the solution and draw it, jointly evaluating textual and geometric reasoning. Prompts are given in Appendix B.

Metrics. We combine text-, code-, image-, and VLM-based metrics that together capture code similarity and final diagram fidelity; full details are in Appendix C. *Keyword Match F1* cross-references key Asymptote functions between generated and ground-truth code, and a family of overlap metrics: BLEU (Papineni et al., 2002), chrF++ (Popović, 2017), CodeBLEU (Ren et al., 2020), and ROUGE-L (Lin, 2004) measure surface similarity; these are cheap but reflect shared boilerplate rather than geometric relations. For visual fidelity, we use *LPIPS* (Zhang et al., 2018) (lower is better). We use *GPT-5.4-mini as a vision-language judge*, scoring each diagram's geometric similarity to the ground truth, and report a *Compile Success Rate*, the fraction of generated Asymptote programs that render without error.

5 Results

Table 1 reports all metrics across models and prompt settings.

Solving performance does not predict diagram fidelity. Across every model, diagram-fidelity scores fall well short of what the same models' problem-solving accuracy would suggest. The text- and code-based metrics remain low (BLEU between 13.9 and 19.1, CodeBLEU between 12.4 and 17.7), and the VLM critic, our most direct measure of geometric faithfulness, never exceeds 53.2 on a 0-100 scale. A model that solves a problem correctly can still produce a diagram sharing little geometric structure with the reference, confirming that diagrammatic reasoning is a capability distinct from textual problem solving rather than a

Table 1: Experimental results (mean score scaled to 100). LPIPS is lower-is-better. VLM Critic raw scores were on a 0-10 scale and have been linearly rescaled to 0-100 for consistency. “With” denotes that the model was given the full reference solution as context; “Without" denotes that the model was given only the problem statement.

Method	GPT-5.4-mini		Claude Sonnet 4.6		Gemini 3.1 Flash-lite		Kimi K2.5		GLM 5.1	
	With	Without	With	Without	With	Without	With	Without	With	Without
Text/Code-Based Metrics										
↑ Keyword Match F1	55.4	51.4	55.9	52.8	47.4	42.1	49.4	47.9	54.6	51.5
↑ BLEU	18.5	16.8	15.7	15.3	16.6	13.9	19.1	16.4	17.8	16.3
↑ chrF++	35.7	32.9	34.7	32.4	30.5	27.2	33.8	30.1	35.1	31.9
↑ CodeBLEU	17.1	15.1	15.1	14.3	14.8	12.4	17.7	15.4	16.4	15.0
↑ ROUGE-L	22.4	21.6	20.2	20.2	24.8	23.4	24.9	24.0	22.6	21.9
Image-Based Metrics										
↓ LPIPS	54.0	53.7	50.6	51.3	52.6	52.7	47.9	49.5	50.8	53.3
VLM-as-a-Judge										
↑ VLM Critic	37.9	46.4	49.5	51.0	42.6	45.2	50.8	53.2	44.2	45.0
Compilation Metrics										
↑ Compile Success Rate	21.7	20.8	27.6	41.3	59.2	59.3	30.9	22.0	40.0	38.6

by-product of it.

Rendering more is not rendering better. Compile success and geometric fidelity are only weakly aligned. Gemini 3.1 Flash-lite compiles most reliably (≈59% in both settings) yet scores below the across-run mean on the VLM critic, whereas Kimi K2.5 and Claude Sonnet 4.6 earn the highest judge scores while compiling far less often (Figure 9). Compile rate alone is therefore a poor proxy for diagrammatic quality, since a model can produce renderable code while still drawing the wrong figure.

Effect of providing the reference solution. Supplying the full reference solution improves surface-level code similarity but does little for rendered fidelity (Figure 10). Every text- and code-similarity metric improves modestly and consistently with the solution, indicating that models reuse recognizable structure from the provided text, while the rendered-image metrics change little and often negatively. The solution context shifts the generated code toward the reference without making the diagram more geometrically correct, placing the bottleneck in construction rather than in understanding the solution.

Comparison with solving baselines. Due to compute constraints we did not re-run the models on the problem-solving task; instead we report each model's published accuracy on AIME 2025, a competition that contributes a large share of our dataset. The evaluated models are near-saturated there: the GPT-5 line reports up to 94.6% without tools and a perfect 100% with code execution (OpenAI, 2026a), and the Claude Sonnet line roughly 87% without tools and 100% with tools (Anthropic, 2026), with comparable figures across the other models. Placing these solving accuracies beside Table 1 exposes a large and consistent gap: models that solve the underlying problems almost perfectly produce diagrams whose VLM-critic fidelity remains below 54/100 and whose programs compile only 36.14% of the time on average.

6 Conclusion

We study diagrammatic reasoning in foundation models. While these models excel at mathematical reasoning, we show that they fail to generate diagrams consistent with textual solutions, even when those solutions are provided to them, a skill that is highly pertinent to problems requiring geometric reasoning and awareness. To measure it, we introduce a high-quality benchmark of olympiad geometry problems, each paired with a human-authored solution and a ground-truth diagram in renderable Asymptote code, together with a suite of text-, code-, image-, and VLM-based metrics. Benchmarking current foundation models on this dataset provides, to our knowledge, the first concrete evaluation of their diagrammatic reasoning, and reveals a pronounced gap between solving and drawing: generated diagrams achieve an average compile success rate of just 36.14%, far below the models' problem-solving ability. These results highlight a continuing need to bring foundation models' diagrammatic reasoning and geometric awareness up to par with their textual reasoning.

Limitations

Due to compute constraints and high API costs associated with model inference we did not run the evaluated models on the problem-solving task ourselves, and instead drew baseline accuracies (such as AIME 2025 performance) from published system cards. For the same reason, we generated only a single diagram per problem from each model rather than sampling multiple times; because generation is stochastic, our reported compile-success and fidelity scores reflect one sample per problem and may vary across runs. Our automatic metrics are not perfect proxies for geometric correctness: text- and code-overlap scores reward shared Asymptote boilerplate, and metrics such as SymPy-based symbolic match, could improve evaluation reliability. Our most direct fidelity measure relies on a VLM judge, which inherits the judge model's own visual and geometric limitations and may misjudge borderline diagrams.

References

Anthropic. 2026. Introducing Claude Sonnet 4.6. Accessed 2026-05-01.

Art of Problem Solving. 2026. AoPS Wiki: Competition problems and solutions. Accessed 2026-05-01.

Jonas Belouadi, Anne Lauscher, and Steffen Eger. 2024a. Automatikz: Text-guided synthesis of scientific vector graphics with tikz. In *The Twelfth International Conference on Learning Representations*.

Jonas Belouadi, Simone Paolo Ponzetto, and Steffen Eger. 2024b. Detikzify: Synthesizing graphics programs for scientific figures and sketches with tikz. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. In *Forty-first International Conference on Machine Learning*.

Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. 2022. UniGeo: Unifying geometry logical reasoning via reformulating mathematical expression. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3313–3323, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric Xing, and Liang Lin. 2021. GeoQA: A geometric question answering benchmark towards multimodal numerical reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 513–523, Online. Association for Computational Linguistics.

Jo-Ku Cheng, Zeren Zhang, Ran Chen, Jingyang Deng, Ziran Qin, and Jinwen Ma. 2025. Geouni: A unified model for generating geometry diagrams, problems and problem solutions. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 3057–3066. ACM.

Shivank Garg, Sankalp Mittal, and Manish Gupta. 2026. Text2arch: A dataset for generating scientific architecture diagrams from natural language descriptions. In *The Fourteenth International Conference on Learning Representations*.

Andy Hammerlindl, John Bowman, and Tom Prince. 2026. *Asymptote: The Vector Graphics Language*. Version 3.10-32.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Preprint*, arXiv:2006.11239.

Markus Hohenwarter. 2007. Dynamic mathematics with geogebra. *Journal of Online Mathematics and its Applications*, 7.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

International Mathematical Olympiad. 2026. Problems. Accessed 2026-05-01.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6774–6786, Online. Association for Computational Linguistics.

Mathematical Association of America. 2026a. American mathematics competitions. Accessed 2026-05-01.

Mathematical Association of America. 2026b. MAA invitational competitions. Accessed 2026-05-01.

OpenAI. 2026a. Introducing GPT-5.4. Accessed 2026-05-01.

OpenAI. 2026b. OpenAI API models. Accessed 2026-05-01.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Runqi Qiao, Qiuna Tan, Guanting Dong, MinhuiWu MinhuiWu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma GongQue, Shanglin Lei, YiFan Zhang, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Xiao Zong, Yida Xu, Peiqing Yang, Zhimin Bao, Muxi Diao, Chen Li, and Honggang Zhang. 2025. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20023–20070, Vienna, Austria. Association for Computational Linguistics.

Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. 2020. Codebleu: a method for automatic evaluation of code synthesis. *Preprint*, arXiv:2009.10297.

Charly Reux, Mathieu Acher, Djamel Eddine Khelladi, Clément Quinton, and Olivier Barais. 2025. Llm code customization with visual results: A benchmark on tikz. In *Proceedings of the 29th International Conference on Evaluation and Assessment in Software Engineering*, EASE '25, page 1086–1096. ACM.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. *Preprint*, arXiv:2112.10752.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Junxiao Wang, Ting Zhang, Heng Yu, Jingdong Wang, and Hua Huang. 2025. Magicgeo: Training-free text-guided geometric diagram generation. *Preprint*, arXiv:2502.13855.

Chengrui Zhang, Maizhen Ning, Tianyi Liu, Zihao Zhou, Jie Sun, Qiufeng Wang, and Kaizhu Huang. 2025. Geosdf: Plane geometry diagram synthesis via signed distance field. *Preprint*, arXiv:2506.13492.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, Peng Gao, and Hongsheng Li. 2024. *MATHVERSE: Does Your Multi-modal LLM Truly See the Diagrams in Visual Math Problems?*, page 169–186. Springer Nature Switzerland.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Dataset Details

A full breakdown of our dataset according to the original problem source and difficulty score is as shown in Table 2:

B Prompts

For our evaluations of foundation models, we use the system prompt (shown in Fig 5).

We then follow up with the following user prompt for evaluations where we provide the solution (shown in Fig 6).

For evaluations without a solution, we use this user prompt instead (shown in Fig 7)

For our VLM-as-a-judge metric, we use GPT-5.4-mini as our judge. We used the following prompt and appended the two rendered images (the ground truth image and the generated image) to create our full prompt (shown in Fig 8).

C Metrics

Text- and Code-Based Matching Metrics We use a Keyword Match F1 score that cross-references key Asymptote functions present in the generated code against those in the ground truth. Alongside it we report a family of overlap metrics: BLEU, chrF++, CodeBLEU, and ROUGE-L. At its core, BLEU measures n-gram precision according to:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \quad (1)$$

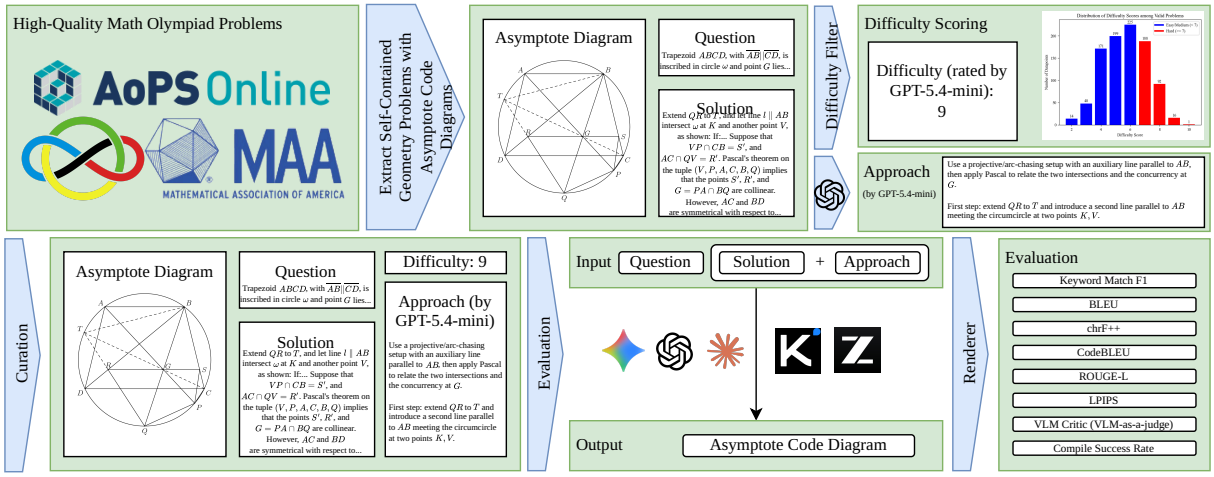

Figure 1: Overview of the Data Curation and Evaluation Processes. We web-scrape high-quality mathematical olympiad problems, namely from national and international competitions with well-documented solutions. We then extract human-made Asymptote code to generate the diagrams corresponding to each solution. To create our dataset, we filter our data, eliminating problems that are not self-contained or not geometric. We then prompt GPT-5.4-mini to generate a difficulty score and a summary of each solution's approach for further analysis. Using a difficulty cutoff, we curate a subset of harder geometry problems. We evaluate foundation models on a range of metrics with two different experimental settings: 1) a baseline with solely the problem statement and 2) including the full reference solution and approach.

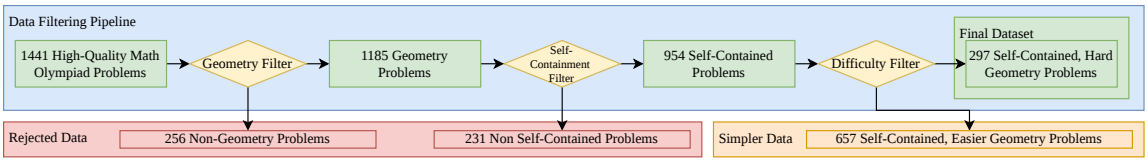

Figure 2: Overview of the Problems Filtered during the Dataset Curation Process. We web-scraped a total of 1441 raw datapoints, before filtering out the problems that were not geometric and not self-contained.

where w_n are the weights and BP is the brevity penalty,

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{2}$$

and p_n is the modified precision for n-grams of size n,

$$p_n = \frac{\sum_{\text{ngram}} \min\big(\mathrm{Count}_{\text{cand}}(\text{ngram}), \mathrm{Count}_{\text{ref}}(\text{ngram})\big)}{\sum_{\text{ngram}} \mathrm{Count}_{\text{cand}}(\text{ngram})}. \tag{3}$$

chrF++ measures character n-gram precision and recall, CodeBLEU augments n-gram overlap with syntactic and semantic program signals, and ROUGE-L measures longest-common-subsequence overlap rather than n-gram overlap. We chose this family for its ability to identify direct matches between generated and ground-truth code; the metrics are cheap and give a rough indication of how close a model's output is to the reference. They are limited, however, in that they cannot capture the geometric relations between elements: a high score may reflect only shared boilerplate, while a correct diagram built from different Asymptote functions may be penalized. Since Asymptote code is highly formulaic and reuses many of the same functions, we retain these metrics as coarse indicators of consistency rather than measures of geometric correctness.

Image-Based Metrics For visual fidelity, we use Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018), which compares deep feature representations of the rendered diagrams. It is computed as:

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \left\| w_l \odot \left(\hat{\phi}^{(x)}_{l,hw} - \hat{\phi}^{(x_0)}_{l,hw} \right) \right\|_2^2 \tag{4}$$

where $\hat{\phi}^{(x)}_{l,hw}$ is the normalized activation tensor from layer l of image x at spatial position (h, w), H_l and W_l are the height and width of the feature map at layer l, and w_l is a learned weight. Lower values indicate greater similarity.

Table 2: Final dataset breakdown by difficulty score.

Dataset	1	2	3	4	5	6	7	8	9	10	Total
AMC	0	14	29	91	95	84	51	15	0	0	379
AIME	0	0	13	63	83	116	105	52	8	0	440
USAJMO	0	0	0	2	5	6	3	5	0	0	21
USAMO	0	0	1	3	3	5	14	15	6	1	48
IMO	0	0	0	0	1	1	3	1	2	0	8
Others	0	0	5	12	12	16	9	4	0	0	58
Total	0	14	48	171	199	225	188	92	16	1	954

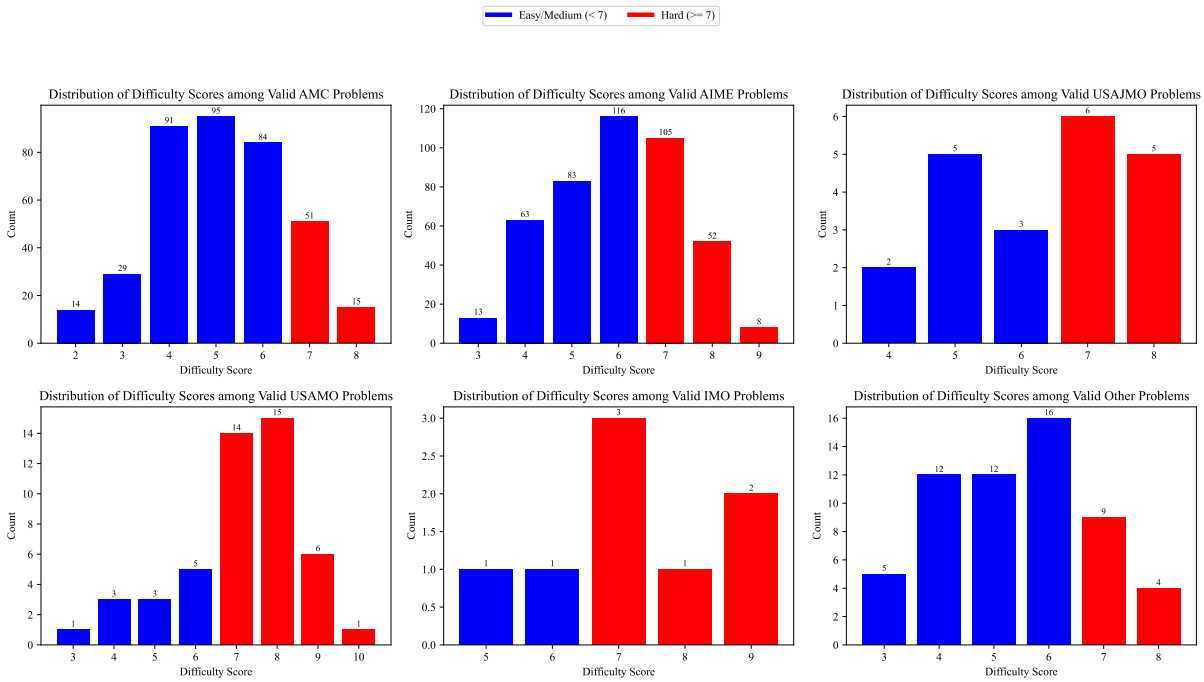

Figure 3: Breakdown of Problems by Difficulty Score.

VLM-Based Metrics We use GPT-5.4-mini as a vision-language judge, scoring each generated diagram against the ground truth on a scale from 0 to 10, which we rescale to 0–100 for consistency with our other metrics. The full prompt is given in Appendix B.

Compilation Metrics We report a Compile Success Rate, the fraction of generated Asymptote programs that render to an image without error. It captures only whether a model's output is usable, not whether the resulting diagram is geometrically correct, and so serves as a lower bound on diagram quality.

D Additional Results

We provide additional analysis figures referenced in the main text below.

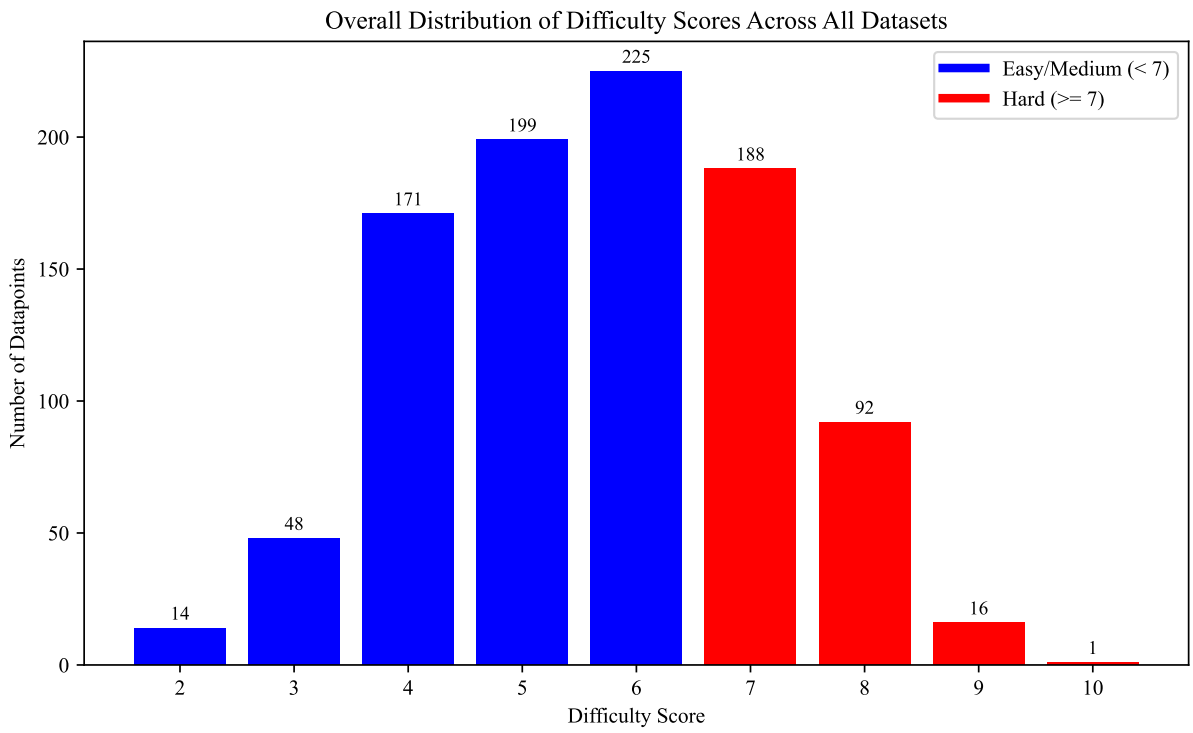

Figure 4: Breakdown of All Problems by Difficulty Score.

```
You are an expert data visualization programmer specializing in research-quality
diagrams using Asymptote.

Generate a complete, self-contained Asymptote script that creates a diagram based
on the user's request.

Rules:
1. Use Asymptote (asy) for all diagrams. Import packages as needed, e.g.:
   - import graph;   // for 2-D function/data plots
   - import stats;   // for histograms, bar charts
   - import three;   // for 3-D diagrams
2. Do NOT specify an output file inside the script. The output path is passed via
   the -o CLI flag.
3. Set canvas size with size(width, height) or size(300) at the top of the script.
4. Follow research publication standards:
   - Clear axis labels with units where applicable
   - Descriptive title using label() or xaxis()/yaxis() labels
   - Legible font sizes
   - Professional colors (use Asymptote named colors or rgb() values; avoid garish
     colors)
   - Include a legend when multiple series are present
5. Use synthetic/example data if the user does not provide specific data. Make it
   realistic and meaningful.
6. IMPORTANT: cos() and sin() take RADIANS. Use dir(degrees) for unit vectors at a
   given angle in degrees.
7. IMPORTANT: label() only accepts ONE pen argument. Combine pens with +:
   label("$x$", p, blue+fontsize(10pt)) not label("$x$", p, fontsize(10pt), blue).
8. Output ONLY the Asymptote code inside a single ```asy code block. No other text.
```

Figure 5: System prompt used for all foundation-model evaluations.

```
Create a diagram for:
{question}

Solution:
{solution}

**Approach to use:**
{approach}
```

Figure 6: User prompt for the with-solution setting.

```
Create a diagram for:

{question}
```

Figure 7: User prompt for the no-solution setting.

```
You are evaluating geometry diagram generation for a research benchmark.
You will see two rendered images:
1. The reference image rendered from the ground-truth Asymptote code.
2. The generated image rendered from a model's Asymptote code.

Judge geometric similarity, not artistic style. Focus on whether the generated
diagram contains the same geometric objects, labels, incidences, relative
layout, angle/parallel/perpendicular markings, and important construction
elements as the reference. Ignore minor rendering artifacts such as antialiasing,
line thickness, small font differences, and harmless scaling/cropping.

Return only JSON with these fields:
{
  "score_0_10": number,
  "summary": string,
  "missing_elements": list[string],
  "wrong_geometry": list[string],
  "extra_elements": list[string],
  "label_issues": list[string]
}

Scoring:
- 9-10: essentially the same geometry and labels.
- 7-8: mostly correct with minor missing or misplaced details.
- 5-6: recognizable core construction but several meaningful issues.
- 3-4: major geometry mismatch, but some relevant structure remains.
- 1-2: mostly wrong or unreadable.
- 0: blank, non-diagram, or completely unrelated.
```

Figure 8: Prompt for the VLM-as-a-judge metric (GPT-5.4-mini).

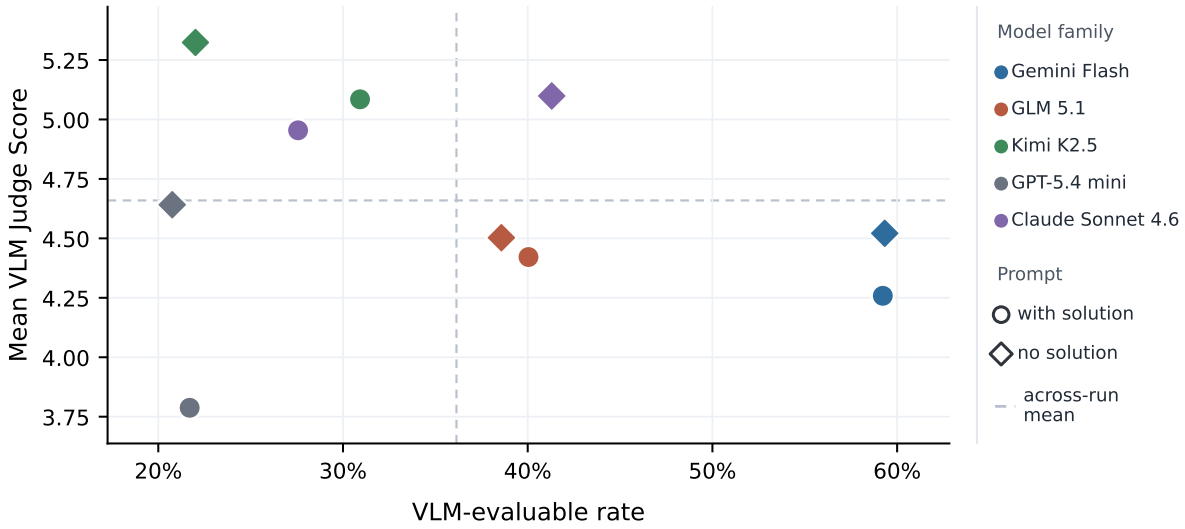

Figure 9: Mean VLM judge score against VLM-evaluable rate for each model and prompt setting (circle: with solution; diamond: no solution). Dashed lines mark the across-run means. Higher evaluable rates do not correspond to higher judge scores: the most reliably renderable models score below the mean on geometric fidelity, while the best-judged runs render less often.

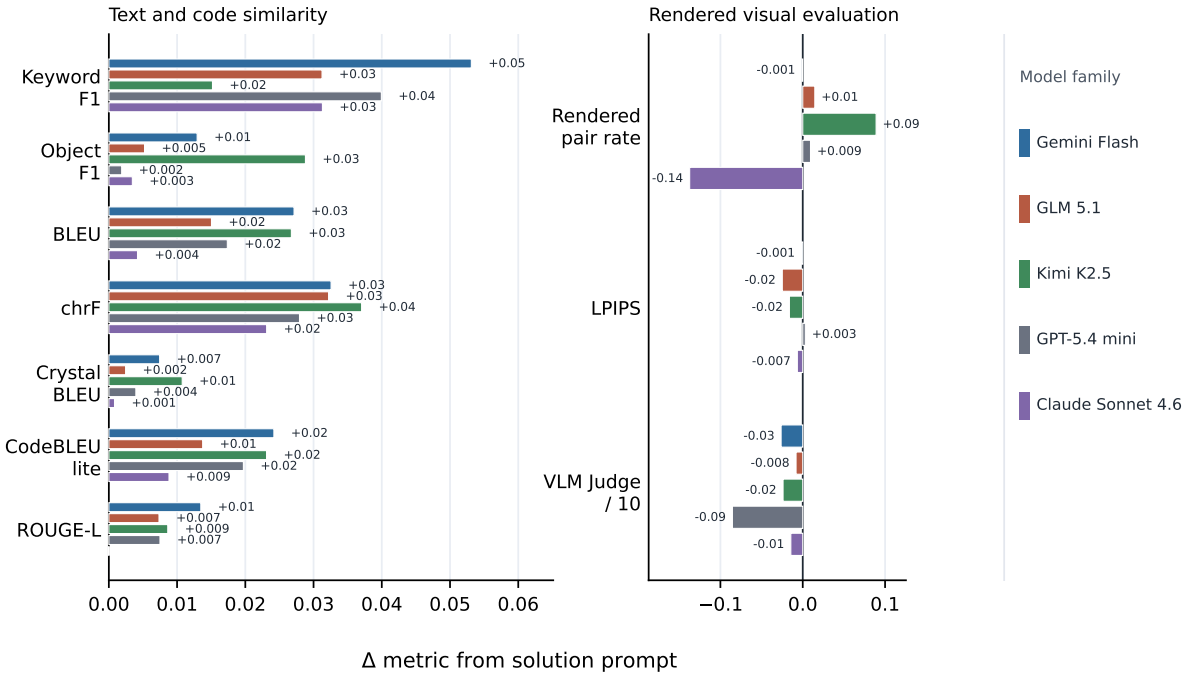

Figure 10: Change in each metric when the full reference solution is provided, relative to the no-solution setting (with minus without), per model. Left: text- and code-similarity metrics improve uniformly. Right: rendered-image metrics change little and often negatively, indicating that solution context raises code overlap without improving diagram fidelity.