TOWARDS THE GENERATION OF STRUCTURED SCIENTIFIC VECTOR GRAPHICS WITH LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We address the challenge of automatically visualizing scientific explanations. While prior work has explored large language model (LLM)-based vector graphic generation, existing approaches often overlook structural correctness, a key requirement for valid scientific diagrams. To achieve structurally correct generation, we make three key contributions. First, we introduce SSVG-Bench, a novel benchmark for evaluating the generation of Structured Scientific Vector Graphics. Unlike conventional visual similarity metrics, SSVG-Bench employs task-specific structural analysis for accurate evaluation, and it supports three vector formats: TikZ, SVG, and EPS. Second, we conduct an extensive benchmarking and analysis, revealing key findings such as the crucial role of LLM reasoning in ensuring structural validity. Third, we propose LLM-Oriented Orchestration Prompting (LOOP), a new prompting method that leverages LLMs' reasoning potential by combining familiar subtasks. Experiments demonstrate substantial improvements over existing prompting techniques, suggesting promising directions for scientific diagram generation. We will release our code and benchmark upon acceptance.

1 INTRODUCTION

In this paper, we address the challenge of automatically visualizing scientific explanations. Because scientific explanations are often abstract and complex, they can be difficult to grasp from text alone. To promote intuitive understanding, visualization is widely used in contexts such as science textbooks and research papers. These observations suggest that automated visualization could have a significant impact.

Recent research on scientific diagram generation has investigated vector graphics generation with large language models (LLMs) (Belouadi et al., 2024a;b; Zhang et al., 2025; Belouadi et al., 2025). As vector graphics encode visual content in structured text form, they can be directly produced by LLMs. Given that LLMs are capable of capturing scientific concepts in depth and encoding complex constraints, they are particularly promising for this task.

However, existing methods have overlooked a crucial aspect of scientific diagrams: structural correctness. To illustrate its importance, we present a physics scenario in Figure 1. Here, the visualization must strictly satisfy structural constraints: the object should remain in contact with the inclined plane, and three arrows must be shown, one vertical to the ground, one perpendicular to the plane, and one parallel to the plane. An existing method, namely a fine-tuned LLM for generating vector code (Belouadi et al., 2025), fails to meet these constraints. Although it prioritizes visual plausibility, the lack of structural enforcement ultimately leads to invalid scientific diagrams.

Towards the generation of scientific vector graphics with structural correctness, we make three main contributions. Our first contribution is a new benchmark for the generation of Structured Scientific Vector Graphics, named SSVG-Bench. It targets two foundational domains: plane geometry and molecular structure. The plane geometry task involves generating geometric figures from textual descriptions of theorems or construction methods, while the molecular structure task requires correctly generating a molecule's structure from its IUPAC name, which encodes structural information. These domains are representative of broader applications: the ability to generate plane geometry structures is fundamental for physics illustrations, engineering diagrams, and architec-

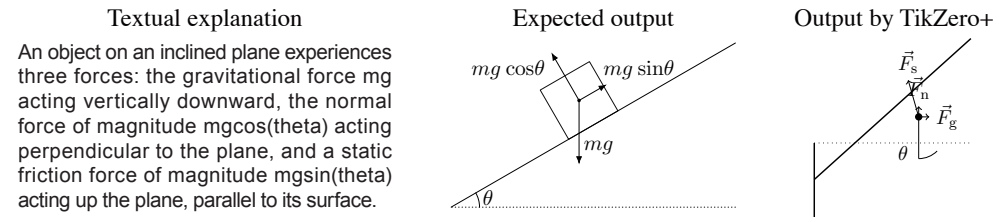


Figure 1: An example highlighting the importance of structural correctness: the object should remain in contact with the inclined plane, and three arrows must be shown, one vertical to the ground, one perpendicular to the plane, and one parallel to the plane. TikZero+ (Belouadi et al., 2025), a fine-tuned LLM for generating vector code, does not meet these structural constraints.

tural blueprints, while generating molecular structures requires correctly producing graph structures, which are important in various fields such as algorithm flowcharts, circuit designs, and biological pathways. The most significant feature of SSVG-Bench is its evaluation method. Conventional visual similarity metrics are insufficient for assessing structural correctness, as even minor visual changes can cause structural inconsistencies. To address this, we provide task-specific Python scripts that analyze the structure of the generated outputs, enabling accurate evaluation of structural correctness. In addition, SSVG-Bench supports three vector formats: TikZ, SVG, and EPS, which allows us to examine performance across formats.

Our second contribution is a comprehensive benchmarking and analysis of existing models using SSVG-Bench, which yields several key findings. First, LLLs fine-tuned on existing vector graphic generation datasets to produce vector code rarely generate structurally valid vector graphics. Second, we show that the reasoning capabilities of LLMs are essential for generating structurally correct vector graphics. Finally, although prior work has mainly focused on the TikZ format, our results demonstrate that the SVG format is better suited for LLM reasoning.

As our third contribution, we propose a new prompting technique, LLM-Oriented Orchestration Prompting (LOOP), to further enhance the reasoning capabilities of LLMs. Recent LLMs are explicitly trained to perform step-by-step reasoning and can solve complex tasks, but it has been reported that they struggle with tasks not encountered during training (Shojaee et al., 2025; Malek et al., 2025). Since LLMs are not explicitly trained to generate vector graphics from scientific explanations, they cannot fully utilize their reasoning potential when the task is presented in its original form. To fully leverage their reasoning abilities, we design a prompt that explicitly guides LLMs to perform LLM-friendly subtasks such as information extraction and relationship extraction, which enables them to generate vector graphics with correct structure. Our experiments demonstrate that LOOP achieves substantially better performance than existing prompting methods.

Our contributions can be summarized as follows.

- Dataset: We introduce SSVG-Bench, a new benchmark for structured scientific vector graphics generation, including scripts that verify structural correctness.
- Benchmarking and analysis: Our analysis shows that previous fine-tuned models cannot produce structurally correct graphics, that LLM reasoning capabilities are essential for ensuring structural correctness, and that the SVG format is well-suited for such reasoning.
- Method: We propose LOOP, a new prompting method that enhances LLM reasoning by solving LLM-friendly sub-problems step by step.

2 Related Works

Scientific vector graphic generation benchmarks. Vector graphics are gaining attention as an image format well-suited for LLMs, as they are represented in text and can be directly input or output by LLMs without requiring a vision adapter. Several benchmarks have been developed for general vector graphics generation, such as SVGEditBench (Nishina & Matsui, 2024), SVG Taxonomy (Xu & Wall, 2024), VGBench (Zou et al., 2024), and SGP-Bench (Qiu et al., 2025). Additionally, several benchmarks for visualizing scientific data have been developed, such as MatPlotBench (Yang et al., 2024), PandasPlotBench (Galimzyanov et al., 2025), and ChartMimic (Yang et al., 2025).

Table 1: Comparison of previous benchmarks with our SSVG-Bench.

Benchmark	# Evaluation data	Evaluation method	Vector format
DaTikZ v1	1,000	Visual and code similarity, Human evaluation	TikZ
DaTikZ v2	1,000	Visual and code similarity, Human evaluation	TikZ
DaTikZ v3	1,000	Visual and code similarity, Human evaluation	TikZ
ScImage	404	Human evaluation	TikZ
DiagramGenBenchmark	470	Visual and code similarity, Human evaluation	TikZ, DOT
SSVG-Bench (ours)	**1,230**	**Structural analysis scripts**	**TikZ, SVG, EPS**

Motivated by this trend, some benchmarks have also been created specifically for scientific vector graphics generation. DaTikZ v1 (Belouadi et al., 2024a) collects TikZ code and corresponding captions from sources such as arXiv papers. DaTikZ v2 (Belouadi et al., 2024b) collects pairs of hand-drawn sketches and TikZ code to evaluate the performance of sketch-to-TikZ conversion. DaTikZ v3 (Belouadi et al., 2025) further extends DaTikZ v1 and v2. ScImage (Zhang et al., 2025) employs synthetic data to analyze scientific vector generation in terms of attributes, numbers, and spatial dimensions. DiagramGenBenchmark (Wei et al., 2025) provides diagram structures in TikZ as well as graph structures in the DOT language.

However, these benchmarks generally do not focus on the structural correctness of the generated graphics. We present a comparison of these benchmarks with our SSVG-Bench in Table 1. The most significant feature of SSVG-Bench is its evaluation method. Visual similarity-based and code similarity-based metrics are insufficient for determining whether the structure of a generated graphic truly reflects the intended structure. Human evaluations, while informative, are not scalable and are subject to variability and inconsistency across evaluators. To address these issues, we provide task-specific Python scripts that analyze the structure of the generated output and determine its correctness. Our evaluation framework offers precise, objective, and consistent assessments of performance. In addition, SSVG-Bench supports three vector formats: TikZ, SVG, and EPS, which allows us to examine performance across formats.

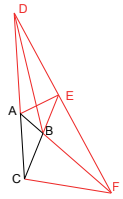
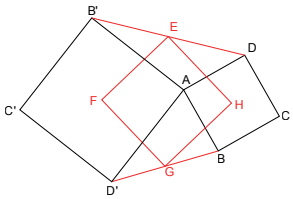
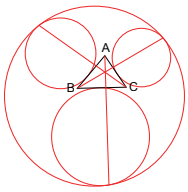
Scientific vector graphic generation methods. Based on the benchmarks, several scientific vector graphic generation methods have been proposed. AutomaTikZ (Belouadi et al., 2024a) is designed for TikZ generation by fine-tuning Llama (Touvron et al., 2023) to output TikZ code from captions. It leverages CLIP features (Radford et al., 2021), derived from captions, to further improve visual alignment. DeTikZify (Belouadi et al., 2024b) converts hand-drawn sketches into TikZ code by combining a vision encoder (SigLIP (Zhai et al., 2023)) with an LLM such as Llama. TikZero (Belouadi et al., 2025) addresses the scarcity of paired caption-TikZ data by leveraging readily available captioned raster images for training. DiagramAgent (Wei et al., 2025) enables the creation of complex diagrams by coordinating multiple agents. Despite their innovations, these models are primarily trained to predict output code and are not designed to guarantee structural correctness.

LLM prompting methods. The reasoning capabilities of LLMs can be elicited through effective prompting. Seminal work on Chain-of-Thought (CoT) demonstrated that allowing models to generate intermediate reasoning steps dramatically improves multi-step reasoning (Wei et al., 2022). Follow-ups revealed that simply appending "Let's think step by step" can unlock zero-shot reasoning (Kojima et al., 2022), and that sampling multiple reasoning paths and selecting the most consistent answer ("self-consistency") further boosts accuracy (Wang et al., 2023b). Beyond linear reasoning, researchers decomposed problems via least-to-most prompting (Zhou et al., 2023), introduced plan-first then execute strategies such as Plan-and-Solve (Wang et al., 2023a), and proposed Step-Back prompting, which encourages the model to abstract away from the immediate problem and reason at a higher conceptual level before providing a solution (Zheng et al., 2024). In this paper, we introduce a novel prompting method designed for scientific vector graphic generation.

3 SSVG-BENCH

To evaluate whether LLMs can generate vector graphics with structural correctness, we introduce SSVG-Bench. SSVG-Bench covers two key topics: 1) plane geometry and 2) molecular structure. For each topic, we also develop an automatic evaluation framework. Figure 2 presents some ex-

Plane geometry

Vector graphic			
Input text	The exterior angle bisector in A intersects the extended side BC in E, the exterior angle bisector in B intersects the extended side AC in D and the exterior angle bisector in C intersects the extended side AB in F. The three points of intersection between the exterior angle bisectors and the extended triangle sides D, E, F are collinear, that is they lie on a common line.	To state the theorem, suppose that ABCD and AB'C'D' are two squares with common vertex A. Let E and G be the midpoints of B'D and D'B respectively, and let F and H be the centers of the two squares. Then the theorem states that the quadrilateral EFGH is a square as well.	The Apollonius point of a triangle is defined as follows. Let △ABC be any given triangle. Let the excircles of △ABC opposite to the vertices A, B, C be EA, EB, EC respectively. Let E be the circle which touches the three excircles EA, EB, EC such that the three excircles are within E. Let A', B', C' be the points of contact of the circle E with the three excircles. The lines AA', BB', CC' are concurrent. The point of concurrence is the Apollonius point of △ABC.

Molecular structure

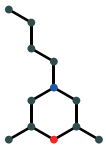
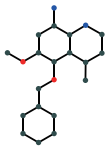
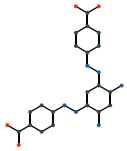
Vector graphic			
Input text	the molecular structure of the compound with the IUPAC name 4-butyl-2,6-dimethylmorpholine	the molecular structure of the compound with the IUPAC name 6-methoxy-4-methyl-5-phenyl methoxyquinolin-8-amine	the molecular structure of the compound with the IUPAC name 4-[[2,4-diamino-5-[(4-carboxyphenyl) diazenyl]phenyl]diazenyl]benzoic acid

Figure 2: Examples in SSVG-Bench.[1]

amples, and Table 2 summarizes respective statistics. We provide a detailed explanation of these components in the following sections.

3.1 Plane Geometry

This task involves generating visualizations of plane geometric figures from textual descriptions, translating explanations of theorems or constructions into precise visual representations. To succeed, LLMs must accurately interpret spatial relationships such as "intersection", "tangent", and "perpendicular", as well as uniquely determined constructions like "angle bisectors" and "excircles." This task can evaluate visualization capabilities that are important in a variety of applications, including physics illustrations, engineering diagrams, and architectural blueprints.

Table 2: Statistics of SSVG-Bench. The number of elements is counted based on the SVG.

Plane geometry	
Total number of input texts	110
Average number of elements per vector graphic	
- line	3.96
- circle	2.77
- ellipse	0.03
- polygon	1.65
- polyline	0.01
Molecular structure	
Total number of input texts	300
Average number of elements per vector graphic	
- line	32.45
- circle	30.38

We curated a dataset by collecting paired textual descriptions and corresponding images related to plane geometry from Wikipedia. All images used are freely available for modification and redistribution. Most of them are in SVG, a vector graphic format. We cleaned the images using Adobe Illustrator by removing elements unrelated to the text. When only raster images were available, we manually recreated the visuals in vector format. In total, we compiled 110 text-vector graphic pairs.

[1]The input texts and vector graphics are from (Wikipedia contributors, 2025b;f; 2024; Kmhkmh, 2019a; 2015; Krishnachandranvn, 2012; National Center for Biotechnology Information, 2025f;l;g)

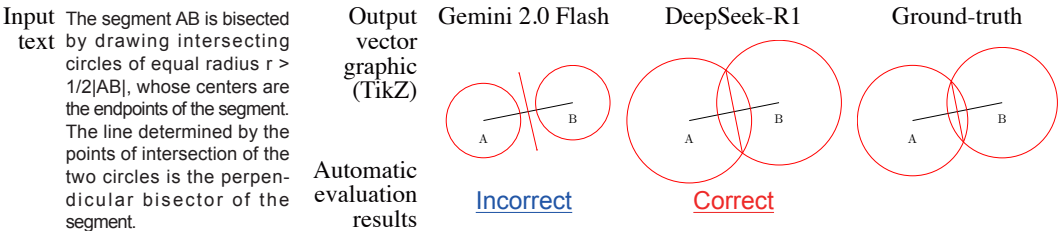


Figure 3: An example of the automatic evaluation results for Pattern 2 in the plane geometry visualization task, where the correct object is not uniquely determined. Our Python-based automatic evaluation framework checks whether the radius of each circle is greater than half the length of segment AB, enabling appropriate assessment.[2]

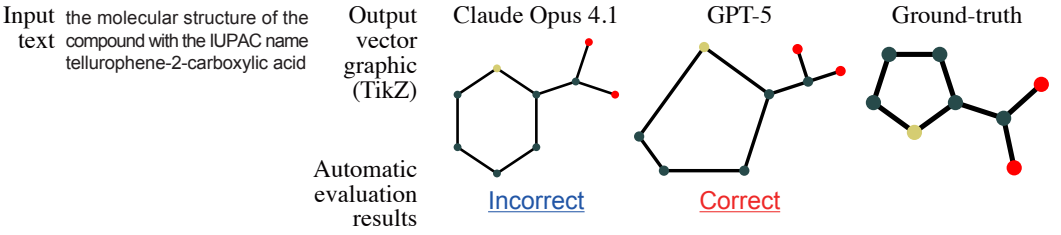


Figure 4: An example of the automatic evaluation results in the molecular structure visualization task. Correctness can be determined based on graph analysis, even when the images differ significantly in visual appearance.[3]

The automatic evaluation falls into two distinct patterns. In Pattern 1, the correct output can be uniquely determined. We provide some parts of the vector graphics as input to the LLM. As shown in Figure 2, the elements depicted in black are given as input, and the LLM is expected to generate the red elements. These red elements are uniquely determined by the black elements. We developed a Python script to analyze the vector data and assess correctness by checking whether the red components are present in the LLM's output.

In Pattern 2, the correct output cannot be uniquely determined, even when the black elements are provided. For example, in the case illustrated in Figure 3, any circle with a radius greater than half the length of the given line segment is considered acceptable. For such cases, we implemented case-specific Python logic to evaluate correctness based on the textual input, allowing for variation in valid outputs. In Figure 3, the output of Gemini 2.0 Flash is judged as incorrect because the radius of the circle is shorter than half the length of segment AB. Although the output of DeepSeek-R1 differs from the predefined ground-truth radius, it is judged as correct since their circle radii exceed half the length of segment AB.

For simplicity, we exclude text layout from the evaluation process.

3.2 Molecular Structure

In this task, the input is an IUPAC name that describes the molecular structure, and the goal is to generate the corresponding molecular structure. The IUPAC name encodes the structural information of a molecule, and by interpreting it, the molecular structure can be reconstructed. For example, the IUPAC name shown on the left side of Figure 2 is "4-butyl-2,6-dimethylmorpholine," which indicates that a butyl group is attached to the nitrogen atom at the 4-position (shown in blue) of the morpholine ring (a six-membered ring), and that methyl groups are attached at the 2- and 6-positions. In this way, the molecular structure can be accurately restored from the IUPAC name. It is essential to correctly identify the types and numbers of atoms involved, with particular importance placed on the graph structure formed by atomic bonds. This task can evaluate the graph structure vi-

[2]The input text and the ground-truth are from (Wikipedia contributors, 2025c) and (Ag2gaeh, 2021).

[3]The molecular structure data is from (National Center for Biotechnology Information, 2025p).

sualization capabilities of LLMs, which are important in various fields such as algorithm flowcharts, circuit design, and biological pathways.

We obtained pairs of IUPAC names and molecular structures from PubChem[4]. The structural data is stored in JSON format, and we developed a Python script to convert this information into vector graphics automatically. Using this script, we generated ground-truth data. We collected 300 molecules in total, with 50 examples each for molecules with fewer than 20 elements (atoms plus bonds), 21-40, 41-60, 61-80, 81-100, and more than 100 elements.

To enable automatic evaluation, we implemented a Python-based evaluation tool. The generated vector graphic is converted into a molecular graph, where nodes represent atoms and edges represent bonds. We then check for "graph isomorphism" between the generated structure and the ground-truth to automatically assess correctness. Graph isomorphism refers to the problem of determining whether two graphs are structurally identical, meaning their nodes and edges can be matched one-to-one while preserving connectivity. We use the NetworkX library to solve this problem. To simplify the task, we do not consider bond order. We present an example of automatic evaluation results in Figure 4. Although the output of GPT-5 appears visually different from the ground-truth, it is considered correct based on graph-theoretic equivalence.

3.3 Multiple Vector Formats

There are various types of vector graphic formats. To analyze performance differences across formats, we use three types: TikZ, SVG, and EPS. For the plane geometry task, since the vector graphics collected from Wikipedia are in SVG format, we developed Python scripts to automatically convert SVG to TikZ and EPS, thereby generating ground-truth data. For the molecular structure task, we generate vector graphics in each format directly from molecular structure data stored in JSON files, using custom Python scripts to automate the process. When evaluating LLMs, we add instructions to the prompt to generate output in a specific format. This approach allows us to explore which vector format is most suitable for LLMs. If the syntax is incorrect, it will result in a compilation error (for TikZ and EPS) or a parsing error (for SVG). In such cases, the output is considered incorrect. Therefore, LLMs must strictly adhere to the syntax of each format. Our SSVG-Bench dataset consists of two tasks and three vector formats, comprising a total of 1,230 text-vector graphic pairs.

4 Benchmarking and Analysis

Using SSVG-Bench, we evaluate whether recent models can generate structurally correct vector graphics. Our experiments utilize two fine-tuned models: AutomaTikZ (Belouadi et al., 2024a) and TikZero+ (Belouadi et al., 2025). As these models are trained to generate TikZ, we evaluate them exclusively on TikZ. We also evaluate recent general-purpose LLMs, including DeepSeek-V3, R1, V3.1 (DeepSeek, 2025), Claude Opus 4.1 (Anthropic, 2025), Gemini 2.0 Flash, 2.5 Flash, 2.5 Pro (Google, 2025), o4-mini, GPT-4.1, and GPT-5 (OpenAI, 2025). The detailed prompts used for evaluation are provided in the Appendix. The overall performance is shown in Table 4, with a detailed analysis provided below.

Limitations of fine-tuned models. Our benchmarking with SSVG-Bench reveals that models fine-tuned to generate TikZ code (AutomaTikZ and TikZero+) seldom produce structurally valid outputs. This highlights a limitation: simply training to predict TikZ code from captions is insufficient to generate correct scientific figures.

Importance of reasoning. To evaluate the effectiveness of reasoning, we consider models where reasoning can be toggled on and off, and we report results for both configurations in Table 4. The models compared are as follows: DeepSeek-V3.1 vs. DeepSeek-V3.1 reasoning, Claude Opus 4.1 vs. Claude Opus 4.1 thinking, Gemini 2.5 Flash vs. Gemini 2.5 Flash reasoning, and GPT-5 Chat vs. GPT-5. Table 5 compares the averages of models with reasoning disabled and enabled. Enabling reasoning significantly improves performance. These results demonstrate that enabling reasoning plays a crucial role in generating structured vector graphics.

[4]https://pubchem.ncbi.nlm.nih.gov/

Table 4: Accuracies on SSVG-Bench (%). The fill colors correspond to the values.

Model	Plane geometry			Molecular structure			Average
	TikZ	SVG	EPS	TikZ	SVG	EPS	
Fine-tuned models							
AutomaTikZ	-	0.0	-	-	0.0	-	0.0
TikZero+	-	0.9	-	-	0.0	-	0.2
Non-reasoning models							
DeepSeek-V3	10.0	5.5	7.3	5.3	3.3	3.0	4.9
DeepSeek-V3.1	11.8	6.4	9.1	6.3	3.7	3.3	5.7
Claude Opus 4.1	14.5	12.7	20.9	24.3	26.0	16.0	20.5
Gemini 2.0 Flash	7.3	5.5	1.8	6.0	3.7	0.7	3.8
Gemini 2.5 Flash	12.7	9.1	5.5	22.7	11.3	14.3	14.2
GPT-4.1	10.9	10.0	14.5	19.0	15.0	13.7	14.8
GPT-5 Chat	12.7	10.0	7.3	16.0	14.3	11.0	12.8
Reasoning models							
DeepSeek-R1	28.2	40.9	39.1	18.3	20.0	19.7	23.8
DeepSeek-V3.1 reasoning	23.6	39.1	27.3	31.0	7.3	20.7	22.4
Claude Opus 4.1 thinking	20.0	23.6	17.3	26.7	27.7	23.3	24.4
Gemini 2.5 Flash reasoning	30.0	55.5	41.8	32.0	39.3	34.7	37.2
Gemini 2.5 Pro	50.0	62.7	56.4	41.3	**63.3**	**57.3**	54.6
o4-mini	48.2	62.7	55.5	33.3	42.7	39.0	42.9
GPT-5	**54.5**	**75.5**	**66.4**	**52.3**	55.7	49.7	**56.0**

Table 5: Comparison of averages for models with reasoning enabled vs. disabled. DeepSeek-V3.1, Claude Opus 4.1, Gemini 2.5 Flash, and GPT-5 are considered.

Model	Plane geometry			Molecular structure			Average
	TikZ	SVG	EPS	TikZ	SVG	EPS	
Reasoning disabled	13.0	9.5	10.7	17.3	13.8	11.2	13.3
Reasoning enabled	**32.0**	**48.4**	**38.2**	**35.5**	**32.5**	**32.1**	**35.0**

Table 3: Google search hits for format-specific keywords (September 2025).

	TikZ	SVG	EPS
Keyword	"\documentclass[tikz]"	"</svg>"	"showpage" and "moveto"
# Hits	154K	422M	23.3K

Impact of vector format. Focusing on the two best-performing models, Gemini 2.5 Pro and GPT-5, we observe that their performance on SVG is the highest, surpassing their performance on TikZ and EPS. This represents a novel finding, as prior benchmarks have primarily focused on TikZ. One possible explanation is the scale of resources used for training. Existing research (Zhu et al., 2024) has demonstrated that the reasoning capabilities of LLMs tend to be weaker in low-resource languages (e.g., Bengali or Thai) compared to high-resource languages (e.g., English). Similarly, it is possible that LLMs are not well-suited for reasoning with TikZ and EPS, because they may be considered "low-resource languages." To test this hypothesis, it would be necessary to examine the training data, but the datasets used to train Gemini 2.5 Pro and GPT-5 remain unspecified. We instead query Google with format-specific keywords and record the number of hits, since much of the training data for LLMs is derived from internet sources. Table 3 presents the number of hits obtained from searches using format-specific keywords. Compared to TikZ and EPS, SVG yielded a much higher number of hits, suggesting that SVG constitutes a high-resource format.

5 LLM-Oriented Orchestration Prompting (LOOP)

Through our analysis, we find that LLM reasoning plays a crucial role. Building on this finding, we propose a method to enhance their reasoning capabilities. Previous research has shown that carefully crafted prompts can significantly improve LLM reasoning, even in zero-shot settings. For instance,

Table 6: Comparison between our LOOP and other zero-shot prompting methods.

Model	Plane geometry			Molecular structure			Average
	TikZ	SVG	EPS	TikZ	SVG	EPS	
Gemini 2.5 Pro	50.0	62.7	56.4	41.3	63.3	57.3	54.6
w/ Zero-shot CoT	39.1	66.4	61.8	**47.7**	63.0	58.7	56.3
w/ Plan-and-Solve	39.1	69.1	**66.4**	41.3	58.7	55.0	53.4
w/ Step-Back	33.6	64.5	59.1	40.7	54.3	56.3	51.0
w/ LOOP (ours)	**65.5**	**80.9**	62.7	**47.7**	**64.7**	**67.7**	**62.6**
GPT-5	54.5	75.5	66.4	52.3	55.7	49.7	56.0
w/ Zero-shot CoT	58.2	**80.0**	75.5	53.0	52.0	49.3	56.7
w/ Plan-and-Solve	61.8	77.3	70.9	52.3	50.3	50.7	56.2
w/ Step-Back	55.5	75.5	72.7	50.7	51.7	48.3	55.0
w/ LOOP (ours)	**70.0**	**80.0**	**77.3**	**55.0**	**57.3**	**54.3**	**61.0**

zero-shot CoT prompting (Kojima et al., 2022), which simply instructs the model with "Let's think step by step," has been shown to improve performance. In this work, we introduce a novel zero-shot prompting method, termed LOOP. While recent LLMs are explicitly trained for step-by-step reasoning and can solve complex tasks, their performance often degrades on tasks outside their training distribution (Shojaee et al., 2025; Malek et al., 2025). Because LLMs are not inherently trained to generate vector graphics from scientific explanations, their reasoning potential remains underutilized. The core idea of LOOP is to instruct LLMs to generate vector graphics by orchestrating LLM-familiar tasks. Specifically, we use the following tasks: 1) information extraction, 2) relationship extraction, 3) mathematical reasoning, and 4) code generation. The first three tasks provide the information necessary for visualization, while the final task produces the vector graphics. Information and relationship extraction are long-standing tasks in the field of natural language processing, whereas mathematical reasoning and code generation are areas where recent LLMs have made significant progress. By orchestrating these familiar tasks, LOOP aims to accelerate and enhance LLM reasoning capabilities.

Specifically, for the plane geometry visual task, we use the following prompt:

"Let's think step by step, following this workflow: 1. Information extraction: describe the necessary elements. 2. Relationship extraction: describe their relationships. 3. Mathematical reasoning: compute the attributes of each element so that they satisfy those relationships. 4. Code generation: generate the TikZ."

For the molecular structure visualization task, we exclude mathematical reasoning, as it is not necessary. We use the following prompt:

"Let's think step by step, following this workflow: 1. Information extraction: describe the functional groups and substituents present in the IUPAC name. 2. Relationship extraction: describe how these groups are connected. 3. Code generation: generate the TikZ."

The word *"TikZ"* is replaced with the target vector format.

5.1 Evaluation

We evaluate the performance of LOOP by applying it to the two best-performing models: Gemini 2.5 Pro and GPT-5. For comparison, we use the following three zero-shot prompting techniques:

- Zero-shot CoT Prompting (Kojima et al., 2022): *"Let's think step by step."*
- Plan-and-Solve Prompting (Wang et al., 2023a): *"Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step."*
- Step-Back Prompting (Zheng et al., 2024): *"Let's think step by step, following this workflow: 1. Step back and pose higher-level, abstract questions. 2. Answer those questions. 3. Generate the TikZ."*

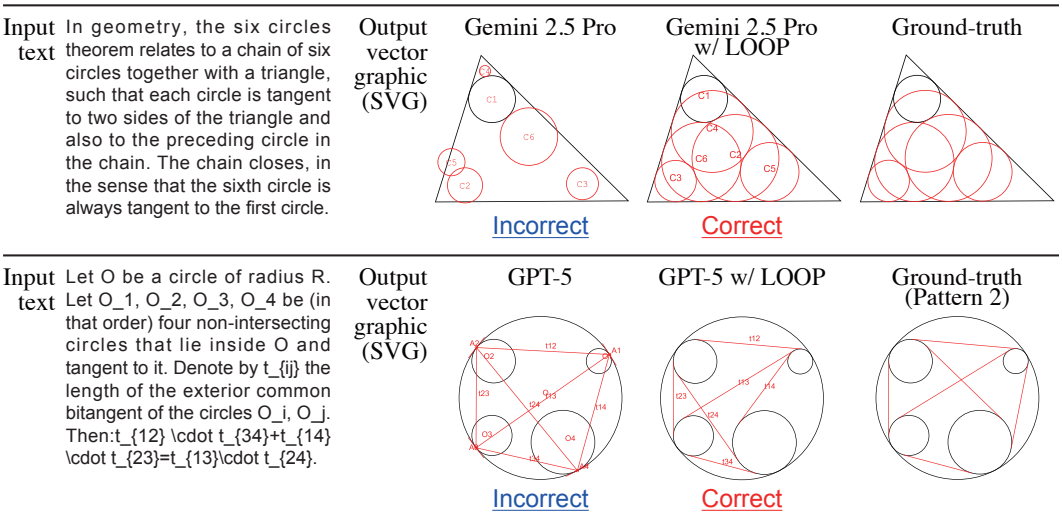


Figure 5: Examples demonstrating improvements from our prompting method on the plane geometry SVG generation task. In the bottom example, since there are two possible exterior common bitangents for each pair of circles, either line is considered correct.[5]

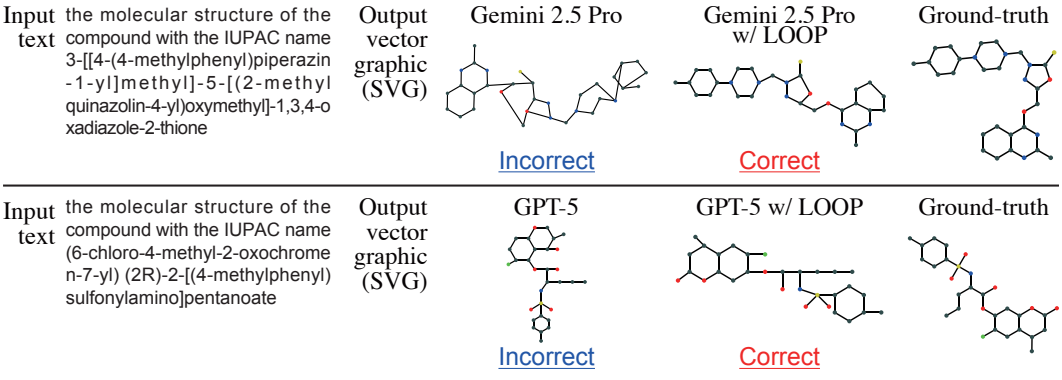


Figure 6: Examples demonstrating improvements from our prompting method on the molecular structure SVG generation task.[6]

The results are shown in Table 6. While some methods degrade performance, the proposed method provides the highest performance gain. This result clearly demonstrates that the proposed method can efficiently leverage the potential of LLMs.

We present examples in Figures 5 and 6 showing that LOOP enables the generation of structurally correct vector graphics. Without LOOP, LLMs fail to accurately produce complex structures. LOOP encourages deeper reasoning, which leads to the generation of structurally correct vector graphics.

6 Conclusion

In this paper, we tackled the problem of scientific vector graphics generation using LLMs. Specifically, aiming for structurally correct vector graphics generation, we made three contributions. First, we introduced a new benchmark that assesses the structural correctness of generated graphics using structural analysis scripts. Second, we conducted a comprehensive benchmarking study and provided detailed analyses based on this benchmark. Third, we proposed a novel prompting technique that accelerates LLM inference and significantly improves performance.

[5]The input texts and the ground-truth vector graphics are from (Wikipedia contributors, 2022; 2025e; Rocchini, 2010; Kmhkmh, 2018).

[6]The molecular structure data is from (National Center for Biotechnology Information, 2025j;o).

REFERENCES

Ag2gaeh. Construction of a perpendicular line segment bisector. https://commons.wikimedia.org/wiki/File:Mittelsenkr-ab-konstr-e.svg, 2021. Licensed under CC BY-SA 4.0; modified by the authors.

Anthropic. Models overview, 2025. https://docs.claude.com/en/docs/about-claude/models/overview.

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

Jonas Belouadi, Simone Ponzetto, and Steffen Eger. Detikzify: Synthesizing graphics programs for scientific figures and sketches with tikz. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:85074–85108, 2024b.

Jonas Belouadi, Eddy Ilg, Margret Keuper, Hideki Tanaka, Masao Utiyama, Raj Dabre, Steffen Eger, and Simone Paolo Ponzetto. Tikzero: Zero-shot text-guided graphics program synthesis. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Braindrain0000. Graphic representing medial triangles gfdl. https://commons.wikimedia.org/wiki/File:Medial_Triangle.svg, 2006. Licensed under CC BY/SA 3.0; modified by the authors.

DeepSeek. Models & pricing, 2025. https://api-docs.deepseek.com/quick_start/pricing.

Timur Galimzyanov, Sergey Titov, Yaroslav Golubev, and Egor Bogomolov. Drawing pandas: A benchmark for llms in generating plotting code. In *IEEE/ACM International Conference on Mining Software Repositories (MSR)*, pp. 503–507, 2025.

Google. Gemini models, 2025. https://ai.google.dev/gemini-api/docs/models.

Gustavb. Butterfly theorem. https://commons.wikimedia.org/wiki/File:Butterfly_theorem.svg, 2006. Licensed under CC BY/SA 3.0; modified by the authors.

Inductiveload. Diagram to shown the construction of the incenter (blue, i), the incircle (blue), the excentres (orange, ja,jb,jc and the excircles (orange). https://commons.wikimedia.org/wiki/File:Incircle_and_Excircles.svg, 2007a. Released into the public domain under CC0 1.0; modified by the authors.

Inductiveload. Extouch_triangle_and_nagel_point.svg — diagram of the extouch triangle and nagel point illustrating splitters in a triangle. https://commons.wikimedia.org/wiki/File:Extouch_Triangle_and_Nagel_Point.svg, 2007b. Released into the public domain under CC0 1.0; modified by the authors.

Inductiveload. Diagram to shown the construction of the intouch, or contact, triangle (red) and the gergonne point (green) of a triangle (black). the blue circle is the incircle, and the blue point, i, is the incentre of the original triangle. https://commons.wikimedia.org/wiki/File:Intouch_Triangle_and_Gergonne_Point.svg, 2007c. Released into the public domain under CC0 1.0; modified by the authors.

Ixnay. An animated gif of the construction of a bisection (with ruler and compass). https://en.wikipedia.org/wiki/File:Bisection_construction.gif, 2007. Released into the public domain under CC0 1.0; modified by the authors.

Kmhkmh. hadwiger-finsler theorem about squares. https://commons.wikimedia.org/wiki/File:Hadwiger_finsler_theorem.svg, 2015. Licensed under CC BY 4.0; modified by the authors.

Kmhkmh. Antiparallel symmedian.svg — diagram showing symmedian and antiparallel segments. https://commons.wikimedia.org/wiki/File:Lemoine_punkt.svg, 2016. Licensed under CC BY 4.0; modified by the authors.

Kmhkmh. Casey new1a.svg. https://commons.wikimedia.org/wiki/File:Casey_new1a.svg, 2018. Licensed under CC BY 4.0; modified by the authors.

Kmhkmh. exterior angle bisectors of a triange. https://commons.wikimedia.org/wiki/File:Aussenwinkelhalbierende2.svg, 2019a. Licensed under CC BY 4.0; modified by the authors.

Kmhkmh. tangential triangle of reference triangle. https://commons.wikimedia.org/wiki/File:Tangential_triangle.svg, 2019b. Licensed under CC BY 4.0; modified by the authors.

Kmhkmh. Japanese theorem for cyclic quadrilaterals (correct version) — diagram. https://commons.wikimedia.org/wiki/File:Japanese_theorem_2_correct_version_a.svg, 2024. Licensed under CC BY 4.0; modified by the authors.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:22199–22213, 2022.

Krishnachandranvn. The construction of the apollonius point. https://commons.wikimedia.org/wiki/File:Apollonius_point.svg, 2012. Released into the public domain under CC0 1.0; modified by the authors.

Alan Malek, Jiawei Ge, Nevena Lazic, Chi Jin, András György, and Csaba Szepesvári. Frontier llms still struggle with simple reasoning tasks. *arXiv preprint arXiv:2507.07313*, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 100329, 2,5-dihydroxy-3,6-bis[1-(2-methylbut-3-en-2-yl)indol-3-yl]cyclohexa-2,5-diene-1,4-dione, 2025a. URL https://pubchem.ncbi.nlm.nih.gov/compound/100329. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 179317, N-[5-chloro-4-(trifluoromethyl)-1,3-thiazol-2-yl]-N-(trideuteriomethyl)-3,5-bis(trifluoromethyl)benzamide, 2025b. URL https://pubchem.ncbi.nlm.nih.gov/compound/179317. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 21011, 4-chloro-1-methyl-5-nitroimidazole, 2025c. URL https://pubchem.ncbi.nlm.nih.gov/compound/21011. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 23521, N-heptylacridin-9-amine, 2025d. URL https://pubchem.ncbi.nlm.nih.gov/compound/23521. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 243518, 3-bromobut-3-en-2-amine, 2025e. URL https://pubchem.ncbi.nlm.nih.gov/compound/243518. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 251192, 4-Butyl-2,6-dimethylmorpholine, 2025f. URL https://pubchem.ncbi.nlm.nih.gov/compound/251192. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 257753, 4-[[2,4-diamino-5-[(4-carboxyphenyl)diazenyl]phenyl]diazenyl]benzoic acid, 2025g. URL https://pubchem.ncbi.nlm.nih.gov/compound/257753. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 272448, methyl 4-[3-acetamido-4,5-diacetyloxy-6-(acetyloxymethyl)oxan-2-yl]oxy-1-(2,4-dinitrophenyl)pyrrolidine-2-carboxylate, 2025h. URL https://pubchem.ncbi.nlm.nih.gov/compound/272448. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 319843, ethyl 3-iminobutanoate, 2025i. URL https://pubchem.ncbi.nlm.nih.gov/compound/319843. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 57105, 3-[[4-(4-methylphenyl)piperazin-1-yl]methyl]-5-[(2-methylquinazolin-4-yl)oxymethyl]-1,3,4-oxadiazole-2-thione, 2025j. URL https://pubchem.ncbi.nlm.nih.gov/compound/57105. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 574024, 5-ethylcyclopentene-1-carboxylic acid, 2025k. URL https://pubchem.ncbi.nlm.nih.gov/compound/574024. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 609005, 6-methoxy-4-methyl-5-phenylmethoxyquinolin-8-amine, 2025l. URL https://pubchem.ncbi.nlm.nih.gov/compound/609005. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 70260, 1,2,5-trimethylpyrrole, 2025m. URL https://pubchem.ncbi.nlm.nih.gov/compound/70260. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 784953, N-(2-methylsulfinylethyl)acetamide, 2025n. URL https://pubchem.ncbi.nlm.nih.gov/compound/784953. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 982106, (6-chloro-4-methyl-2-oxochromen-7-yl) (2R)-2-[(4-methylphenyl)sulfonylamino]pentanoate, 2025o. URL https://pubchem.ncbi.nlm.nih.gov/compound/982106. Retrieved September 10, 2025.

National Center for Biotechnology Information. PubChem Compound Summary for CID 141986, 2-Tellurophenecarboxylic acid, 2025p. URL https://pubchem.ncbi.nlm.nih.gov/compound/141986. Retrieved September 10, 2025.

Kunato Nishina and Yusuke Matsui. Svgeditbench: A benchmark dataset for quantitative assessment of llm's svg editing capabilities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 8142–8147, 2024.

OpenAI. Models, 2025. https://platform.openai.com/docs/models.

PegasusRoe. image for projection formula in trigonometry. https://commons.wikimedia.org/wiki/File:Projection_formula_(3).png, 2007. Released into the public domain under CC0 1.0; modified by the authors.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Zhen Liu, Tim Z Xiao, Katherine M Collins, Joshua B Tenenbaum, Adrian Weller, Michael J Black, and Bernhard Schölkopf. Can large language models understand symbolic graphics programs? In *International Conference on Learning Representations (ICLR)*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pp. 8748–8763, 2021.

Claudio Rocchini. Isogonal conjugate.svg — illustration of isogonal conjugacy in a triangle. https://commons.wikimedia.org/wiki/File:Isogonal_Conjugate.svg, 2008. Licensed under CC BY 3.0; modified by the authors.

Claudio Rocchini. Six circles theorem: examples of some configuration. https://commons.wikimedia.org/wiki/File:Six_circles_theorem.svg, 2010. Licensed under CC BY/SA 3.0; modified by the authors.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2609–2634, 2023a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:24824–24837, 2022.

Jingxuan Wei, Cheng Tan, Qi Chen, Gaowei Wu, Siyuan Li, Zhangyang Gao, Linzhuang Sun, Bihui Yu, and Ruifeng Guo. From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13315–13325, 2025.

Wikipedia contributors. Six circles theorem. `https://en.wikipedia.org/wiki/Six_circles_theorem`, 2022. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Apollonius point. `https://en.wikipedia.org/wiki/Apollonius_point`, 2024. [Online; accessed 10-September-2025; licensed under CC BY-/SA 4.0].

Wikipedia contributors. Altitude (triangle). `https://en.wikipedia.org/wiki/Altitude_(triangle)`, 2025a. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Angle bisector theorem. `https://en.wikipedia.org/wiki/Angle_bisector_theorem`, 2025b. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Bisection. `https://en.wikipedia.org/wiki/Bisection`, 2025c. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Butterfly theorem. `https://en.wikipedia.org/wiki/Butterfly_theorem`, 2025d. [Online; accessed 10-September-2025; licensed under CC BY-/SA 4.0].

Wikipedia contributors. Casey's theorem. `https://en.wikipedia.org/wiki/Casey%27s_theorem`, 2025e. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Finsler-hadwiger theorem. `https://en.wikipedia.org/wiki/Finsler%E2%80%93Hadwiger_theorem`, 2025f. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Incircle and excircles. `https://en.wikipedia.org/wiki/Incircle_and_excircles`, 2025g. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Isogonal conjugate. `https://en.wikipedia.org/wiki/Isogonal_conjugate`, 2025h. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Japanese theorem for cyclic quadrilaterals. https://en.wikipedia.org/wiki/Japanese_theorem_for_cyclic_quadrilaterals, 2025i. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Medial triangle. https://en.wikipedia.org/wiki/Medial_triangle, 2025j. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Splitter (geometry). https://en.wikipedia.org/wiki/Splitter_%28geometry%29, 2025k. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Symmedian. https://en.wikipedia.org/wiki/Symmedian, 2025l. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Wikipedia contributors. Tangential triangle. https://en.wikipedia.org/wiki/Tangential_triangle, 2025m. [Online; accessed 10-September-2025; licensed under CC BY/SA 4.0].

Zhongzheng Xu and Emily Wall. Exploring the capability of llms in performing low-level visual analytic tasks on svg data visualizations. In *IEEE Visualization and Visual Analytics (VIS)*, pp. 126–130, 2024.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran XU, Xinyu Zhu, Siheng Li, Yuxiang Zhang, et al. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Zhiyu Yang, Zihan Zhou, Shuo Wang, Xin Cong, Xu Han, Yukun Yan, Zhenghao Liu, Zhixing Tan, Pengyuan Liu, Dong Yu, et al. Matplotagent: Method and evaluation for llm-based agentic scientific data visualization. In *Findings of the Association for Computational Linguistics (ACL)*, pp. 11789–11804, 2024.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11975–11986, 2023.

Leixin Zhang, Steffen Eger, Yinjie Cheng, Weihe Zhai, Jonas Belouadi, Christoph Leiter, Simone Paolo Ponzetto, Fahimeh Moafian, and Zhixue Zhao. Scimage: How good are multimodal large language models at scientific text-to-image generation? In *International Conference on Learning Representations (ICLR)*, 2025.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Wenhao Zhu, Shujian Huang, Fei Yuan, Shuaijie She, Jiajun Chen, and Alexandra Birch. Question translation training for better multilingual reasoning. In *Findings of the Association for Computational Linguistics (ACL)*, pp. 8411–8423, 2024.

Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. Vgbench: A comprehensive benchmark of vector graphics understanding and generation for large language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3647–3659, 2024.

A THE USE OF LARGE LANGUAGE MODELS (LLMS)

We used ChatGPT when writing our paper, for translation purposes.

B DETAILED PROMPTS USED IN OUR EXPERIMENTS

Figures 7, 8, and 9 show examples of prompts used in the plane geometry task. We first present a textual description, followed by input elements (the black elements in Figure 2).

Figures 10, 11, and 12 show examples of prompts used in the molecular structure task. We first provide the IUPAC name, followed by a set of instructions. In particular, we include color specifications so that the types of atoms can be identified by their color. We also present an example using chlorobenzene, which is a relatively simple molecule.

[Explanation]The exterior angle bisector in A intersects the extended side BC in E, the exterior angle bisector in B intersects the extended side AC in D and the exterior angle bisector in C intersects the extended side AB in F.

The three points of intersection between the exterior angle bisectors and the extended triangle sides D, E, F are collinear, that is they lie on a common line.
[TikZ]\documentclass{standalone}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}[x=1pt,y=1pt]
\draw (38.23,-264.41) -- (65.98,-195.89) -- (32.16,-166.40) -- cycle;
\node at (23.54,-167.59) {A};
\node at (70.26,-197.89) {B};
\node at (28.79,-270.30) {C};
\end{tikzpicture}
\end{document}
Please visualize the explanation by adding elements to the TikZ diagram.
– Set the color of any added elements to red.
– Do not modify any elements that are originally present in the TikZ.
– Do not use any animations.

Figure 7: An example prompt from the TikZ generation task on plane geometry.

[Explanation]The exterior angle bisector in A intersects the extended side BC in E, the exterior angle bisector in B intersects the extended side AC in D and the exterior angle bisector in C intersects the extended side AB in F.

The three points of intersection between the exterior angle bisectors and the extended triangle sides D, E, F are collinear, that is they lie on a common line.
[SVG]<svg xmlns="http://www.w3.org/2000/svg"
xmlns:ev="http://www.w3.org/2001/xml-events" version="1.1"
viewBox="0 0 300 300">
<style>
.input_object {
  fill: none;
  stroke: black;
  stroke-width: 1;
}
.input_text {
  fill: black;
  stroke: none;
  font-size: 12px;
}
.output_object {
  fill: none;
  stroke: red;
  stroke-width: 1;
}
.output_text {
  fill: red;
  stroke: none;
  font-size: 12px;
}
</style>
<polygon class="input_object" points="38.23232323232324
264.4065656565657 65.9848484848485 195.8901515151515
32.15909090909092 166.40151515151516"/>
<text class="input_text" x="23.541666666666668"
y="167.58838383838383">A</text>
<text class="input_text" x="70.2588383838384"
y="197.89141414141415">B</text>
<text class="input_text" x="28.787878787878793"
y="270.30303030303037">C</text>
</svg>
Please visualize the explanation by adding elements to the SVG diagram.
- Assign class="output_text" to any added text elements, and class="output_object" to all other added elements.
- Do not modify any elements that are originally present in the SVG.
– Do not use any animations.

Figure 8: An example prompt from the SVG generation task on plane geometry.

[Explanation]The exterior angle bisector in A intersects the extended side BC in E, the exterior angle bisector in B intersects the extended side AC in D and the exterior angle bisector in C intersects the extended side AB in F.

The three points of intersection between the exterior angle bisectors and the extended triangle sides D, E, F are collinear, that is they lie on a common line.
[EPS]%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 300 300
1 setlinewidth
0 0 0 setrgbcolor
newpath
38.23232323232324 35.593434343434296 moveto
65.9848484848485 104.1098484848485 lineto
32.15909090909092 133.59848484848484 lineto
closepath
stroke
/Helvetica findfont 12 scalefont setfont
0 0 0 setrgbcolor
newpath
23.541666666666668 132.41161616161617 moveto
(A) show
/Helvetica findfont 12 scalefont setfont
0 0 0 setrgbcolor
newpath
70.2588383838384 102.10858585858585 moveto
(B) show
/Helvetica findfont 12 scalefont setfont
0 0 0 setrgbcolor
newpath
28.787878787878793 29.69696969696963 moveto
(C) show
showpage
Please visualize the explanation by adding elements to the EPS diagram.
– Set the color of any added elements to red.
– Do not modify any elements that are originally present in the EPS.
– Do not use any animations.

Figure 9: An example prompt from the EPS generation task on plane geometry.

Please create a TikZ file that visualizes the molecular structure of the compound with the IUPAC name 4-butyl-2,6-dimethylmorpholine. Represent each atom as a circle, using colors to indicate atom types. The color mapping for each atom type is provided below, although not all listed types may be present in the molecule. Omit hydrogen atoms from the visualization. Depict bonds between atoms as lines, using a single line for each bond regardless of bond order.

H: #638c8c, B: #2AD52A, C: #274A4A, N: #0000FF, O: #FF0000, F: #D52092, Si: #D59E13, P: #D58600, S: #D5D500, Cl: #2AD52A, Br: #D58639, Te: #D5CD72, I: #FF00FF, Eu: #00CCD5, Lu: #00CCD5, Os: #838C8C, U: #00CCD5

As a reference, an example TikZ visualization of the compound chlorobenzene is provided below.
```
\documentclass[tikz]{standalone}
\definecolor{274A4A}{HTML}{274A4A}
\definecolor{2AD52A}{HTML}{2AD52A}
\begin{document}
\begin{tikzpicture}[x=1pt,y=1pt]
  \draw[line width=1pt] (20.00,6.22) -- (28.66,1.22);
  \draw[line width=1pt] (45.98,1.22) -- (45.98,-8.78);
  \draw[line width=1pt] (45.98,1.22) -- (37.32,6.22);
  \draw[line width=1pt] (45.98,-8.78) -- (37.32,-13.78);
  \draw[line width=1pt] (37.32,6.22) -- (28.66,1.22);
  \draw[line width=1pt] (37.32,-13.78) -- (28.66,-8.78);
  \draw[line width=1pt] (28.66,1.22) -- (28.66,-8.78);
  \filldraw[fill=2AD52A, draw=none] (20.00,6.22) circle (1.5pt);
  \filldraw[fill=274A4A, draw=none] (45.98,1.22) circle (1.5pt);
  \filldraw[fill=274A4A, draw=none] (45.98,-8.78) circle (1.5pt);
  \filldraw[fill=274A4A, draw=none] (37.32,6.22) circle (1.5pt);
  \filldraw[fill=274A4A, draw=none] (37.32,-13.78) circle (1.5pt);
  \filldraw[fill=274A4A, draw=none] (28.66,1.22) circle (1.5pt);
  \filldraw[fill=274A4A, draw=none] (28.66,-8.78) circle (1.5pt);
\end{tikzpicture}
\end{document}
```

Figure 10: An example prompt from the TikZ generation task on molecular structure.

Please create an SVG file that visualizes the molecular structure of the compound with the IUPAC name 4-butyl-2,6-dimethylmorpholine. Represent each atom as a circle, using colors to indicate atom types. The color mapping for each atom type is provided below, although not all listed types may be present in the molecule. Omit hydrogen atoms from the visualization. Depict bonds between atoms as lines, using a single line for each bond regardless of bond order.

H: #638c8c, B: #2AD52A, C: #274A4A, N: #0000FF, O: #FF0000, F: #D52092, Si: #D59E13, P: #D58600, S: #D5D500, Cl: #2AD52A, Br: #D58639, Te: #D5CD72, I: #FF00FF, Eu: #00CCD5, Lu: #00CCD5, Os: #838C8C, U: #00CCD5

As a reference, an example SVG visualization of the compound chlorobenzene is provided below.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="10 -29.976
67.479 59.646">
<line x1="20" y1="6.2239999999999999" x2="28.66" y2="1.224"
stroke="black" stroke-width="1" />
<line x1="45.980999999999995" y1="1.224"
x2="45.980999999999995" y2="-8.776" stroke="black"
stroke-width="1" />
<line x1="45.980999999999995" y1="1.224" x2="37.32"
y2="6.2239999999999999" stroke="black" stroke-width="1" />
<line x1="45.980999999999995" y1="-8.776" x2="37.32" y2="-13.776"
stroke="black" stroke-width="1" />
<line x1="37.32" y1="6.2239999999999999" x2="28.66" y2="1.224"
stroke="black" stroke-width="1" />
<line x1="37.32" y1="-13.776" x2="28.66" y2="-8.776" stroke="black"
stroke-width="1" />
<line x1="28.66" y1="1.224" x2="28.66" y2="-8.776" stroke="black"
stroke-width="1" />
<circle cx="20" cy="6.2239999999999999" r="1.5" fill="#2AD52A" />
<circle cx="45.980999999999995" cy="1.224" r="1.5" fill="#274A4A"
/>
<circle cx="45.980999999999995" cy="-8.776" r="1.5" fill="#274A4A"
/>
<circle cx="37.32" cy="6.2239999999999999" r="1.5" fill="#274A4A" />
<circle cx="37.32" cy="-13.776" r="1.5" fill="#274A4A" />
<circle cx="28.66" cy="1.224" r="1.5" fill="#274A4A" />
<circle cx="28.66" cy="-8.776" r="1.5" fill="#274A4A" />
</svg>
```

Figure 11: An example prompt from the SVG generation task on molecular structure.

Please create an EPS file that visualizes the molecular structure of the compound with the IUPAC name 4-butyl-2,6-dimethylmorpholine. Represent each atom as a circle, using colors to indicate atom types. The color mapping for each atom type is provided below, although not all listed types may be present in the molecule. Omit hydrogen atoms from the visualization. Depict bonds between atoms as lines, using a single line for each bond regardless of bond order.

H: [0.39, 0.55, 0.55], B: [0.16, 0.84, 0.16], C: [0.15, 0.29, 0.29], N: [0.0, 0.0, 1.0], O: [1.0, 0.0, 0.0], F: [0.84, 0.13, 0.57], Si: [0.84, 0.62, 0.07], P: [0.84, 0.53, 0.0], S: [0.84, 0.84, 0.0], Cl: [0.16, 0.84, 0.16], Br: [0.84, 0.53, 0.22], Te: [0.84, 0.8, 0.45], I: [1.0, 0.0, 1.0], Eu: [0.0, 0.8, 0.84], Lu: [0.0, 0.8, 0.84], Os: [0.51, 0.55, 0.55], U: [0.0, 0.8, 0.84]

As a reference, an example EPS visualization of the compound chlorobenzene is provided below.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 10 -29 77 29
%%Creator: EPS Generator
%%EndComments

1 setlinewidth
0 0 0 setrgbcolor

newpath
20 6.223999999999999 moveto
28.66 1.224 lineto
stroke

newpath
45.980999999999995 1.224 moveto
37.32 6.223999999999999 lineto
stroke

newpath
37.32 6.223999999999999 moveto
28.66 1.224 lineto
stroke

newpath
28.66 1.224 moveto
28.66 -8.776 lineto
stroke

newpath
28.66 -8.776 moveto
37.32 -13.776 lineto
stroke

newpath
37.32 -13.776 moveto
45.980999999999995 -8.776 lineto
stroke

newpath
45.980999999999995 -8.776 moveto
45.980999999999995 1.224 lineto
stroke

0.16 0.84 0.16 setrgbcolor
newpath
20 6.223999999999999 1.5 0 360 arc
closepath
fill

0.15 0.29 0.29 setrgbcolor
newpath
45.980999999999995 1.224 1.5 0 360 arc
closepath
fill

0.15 0.29 0.29 setrgbcolor
newpath
45.980999999999995 -8.776 1.5 0 360 arc
closepath
fill

0.15 0.29 0.29 setrgbcolor
newpath
37.32 6.223999999999999 1.5 0 360 arc
closepath
fill

0.15 0.29 0.29 setrgbcolor
newpath
37.32 -13.776 1.5 0 360 arc
closepath
fill

0.15 0.29 0.29 setrgbcolor
newpath
28.66 1.224 1.5 0 360 arc
closepath
fill

0.15 0.29 0.29 setrgbcolor
newpath
28.66 -8.776 1.5 0 360 arc
closepath
fill

showpage
```

Figure 12: An example prompt from the EPS generation task on molecular structure.

C Limitations of Our Automatic Evaluation Framework

Our automatic evaluation framework for the plane geometry visualization task assesses whether the necessary elements are present in the output, but it does not penalize the inclusion of unnecessary elements. In the example shown in Figure 13, the output from Gemini 2.5 Flash reasoning includes an irrelevant straight line, yet it is still considered correct. We do not penalize unnecessary elements because it is often non-trivial to determine whether an additional element is truly unnecessary. For instance, the output from Gemini 2.5 Flash reasoning in Figure 13 includes circles not anticipated in the ground-truth, but these represent intersections and the circle center, and they do not hinder the explanation.

Because in practical scenarios it is usually easier for humans to remove unnecessary elements than to create necessary ones from scratch, we do not currently view this limitation as a major issue. However, enabling the framework to identify and evaluate such extraneous content remains an important direction for future work.

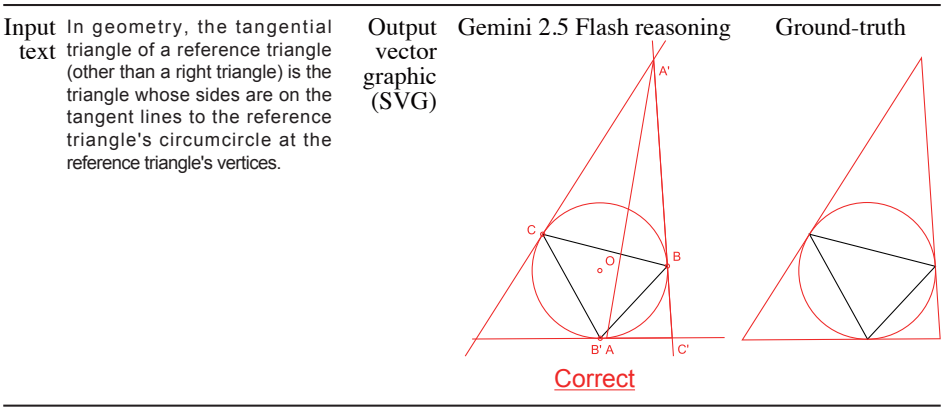


Figure 13: An example judged correct despite an unrelated line.[7]

[7]The input text and the ground-truth vector graphic are from (Wikipedia contributors, 2025m; Kmhkmh, 2019b).

D Additional Examples of Generated Vector Graphics

D.1 Examples Generated by Fine-tuned Models

Figure 14 shows examples generated by two fine-tuned models, AutomaTikZ (Belouadi et al., 2024a) and TikZero+ (Belouadi et al., 2025). The top example is the only case where TikZero+ produces a structurally correct vector graphic. However, in all other cases, the models fail to generate structurally correct vector graphics. In the plane geometry task, they cannot follow simple instructions such as "Set the color of any added elements to red" or "Do not modify any elements that are originally present in the TikZ." In the molecular structure task, the models generate entirely invalid vector graphics. These results reveal the limitations of fine-tuned models.

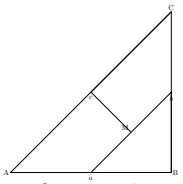
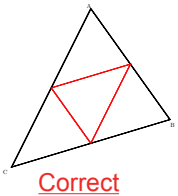
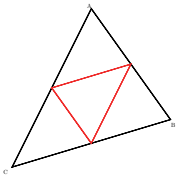
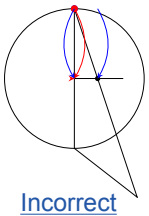
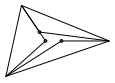
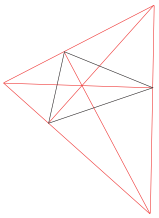
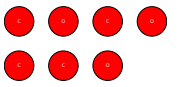
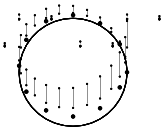
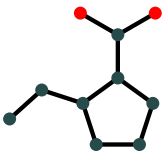
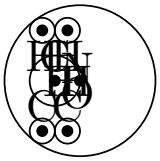
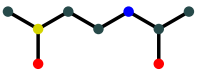
Input text	Output vector graphic (TikZ)	AutomaTikZ	TikZero+	Ground-truth
In Euclidean geometry, the medial triangle or midpoint triangle of a triangle △ABC is the triangle with vertices at the midpoints of the triangle's sides AB, AC, BC.		 Incorrect	 Correct	
The center of an excircle is the intersection of the internal bisector of one angle and the external bisectors of the other two. Every triangle has three excircle centers.		 Incorrect	 Incorrect	
the molecular structure of the compound with the IUPAC name 5-ethylcyclopentene-1-carboxylic acid		 Incorrect	 Incorrect	
the molecular structure of the compound with the IUPAC name N-(2-methylsulfinylethyl)acetamide		Incorrect	 Incorrect	

Figure 14: Examples generated by fine-tuned models.[8]

[8]The input text and the ground-truth vector graphic are from (Wikipedia contributors, 2025j;g; Braindrain0000, 2006; Inductiveload, 2007a). The molecular structure data is from (National Center for Biotechnology Information, 2025k;n).

D.2 Impact of Reasoning

We show in Figures 15 and 16 that enabling reasoning allows LLMs to output the correct structure. Without reasoning, they struggle to generate even simple structures.

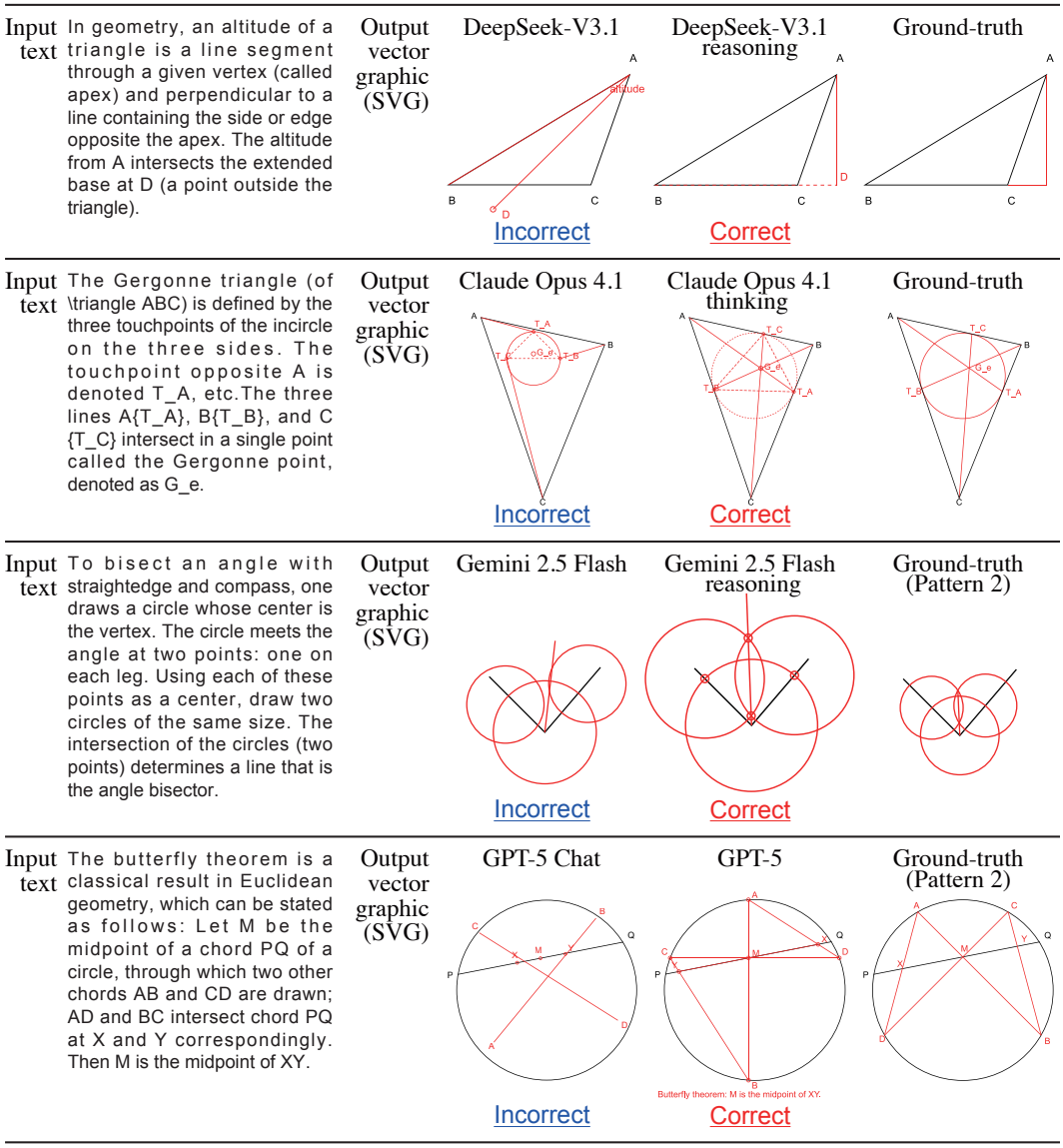


Figure 15: Examples where enabling reasoning allows LLMs to generate the correct structure in the plane geometry SVG geneartion task.[9]

[9]The input texts and the ground-truth vector graphics are from (Wikipedia contributors, 2025a;g;c;d; PegasusRoe, 2007; Inductiveload, 2007c; Ixnay, 2007; Gustavb, 2006).

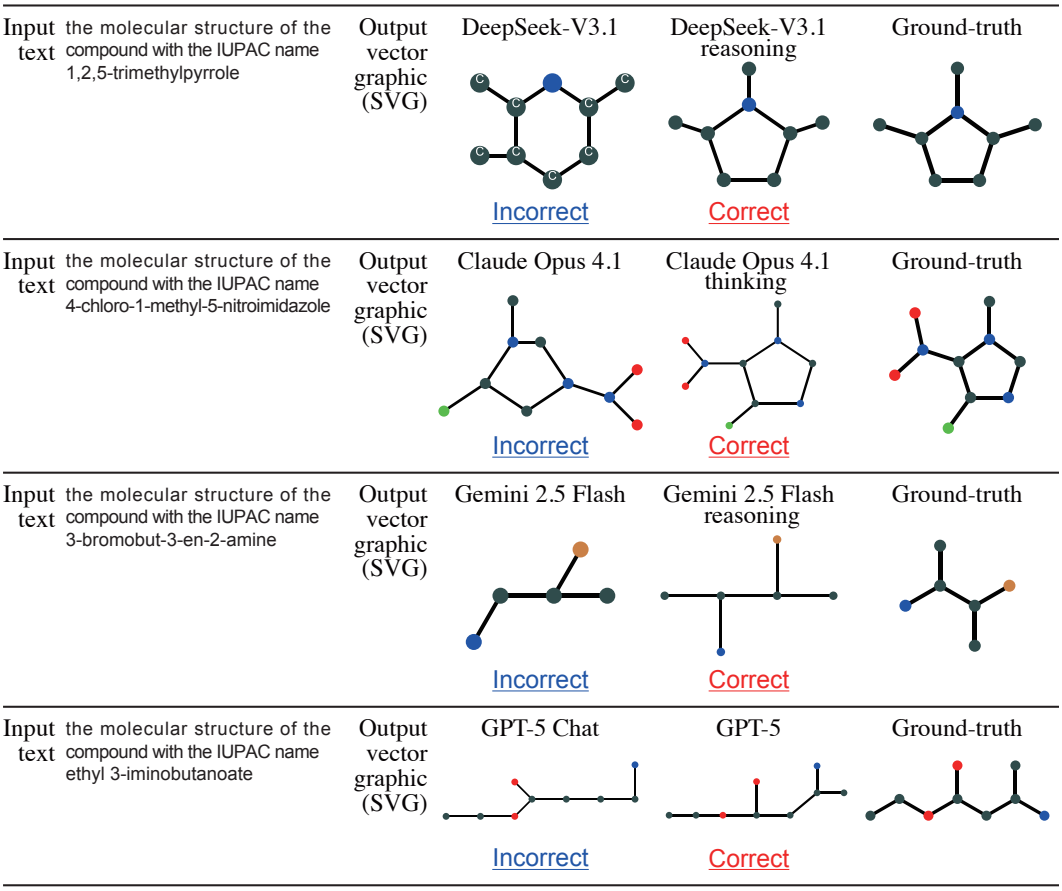


Figure 16: Examples where enabling reasoning allows LLMs to generate the correct structure in the molecular structure SVG geneartion task.[10]

[10]The molecular structure data is from (National Center for Biotechnology Information, 2025m;c;e;i).

D.3 Impact of Format

Figures 17 and 18 present examples where LLMs produce correct structures in SVG format but fail in TikZ and EPS formats. Although the input text is identical, the results clearly vary depending on the output format.

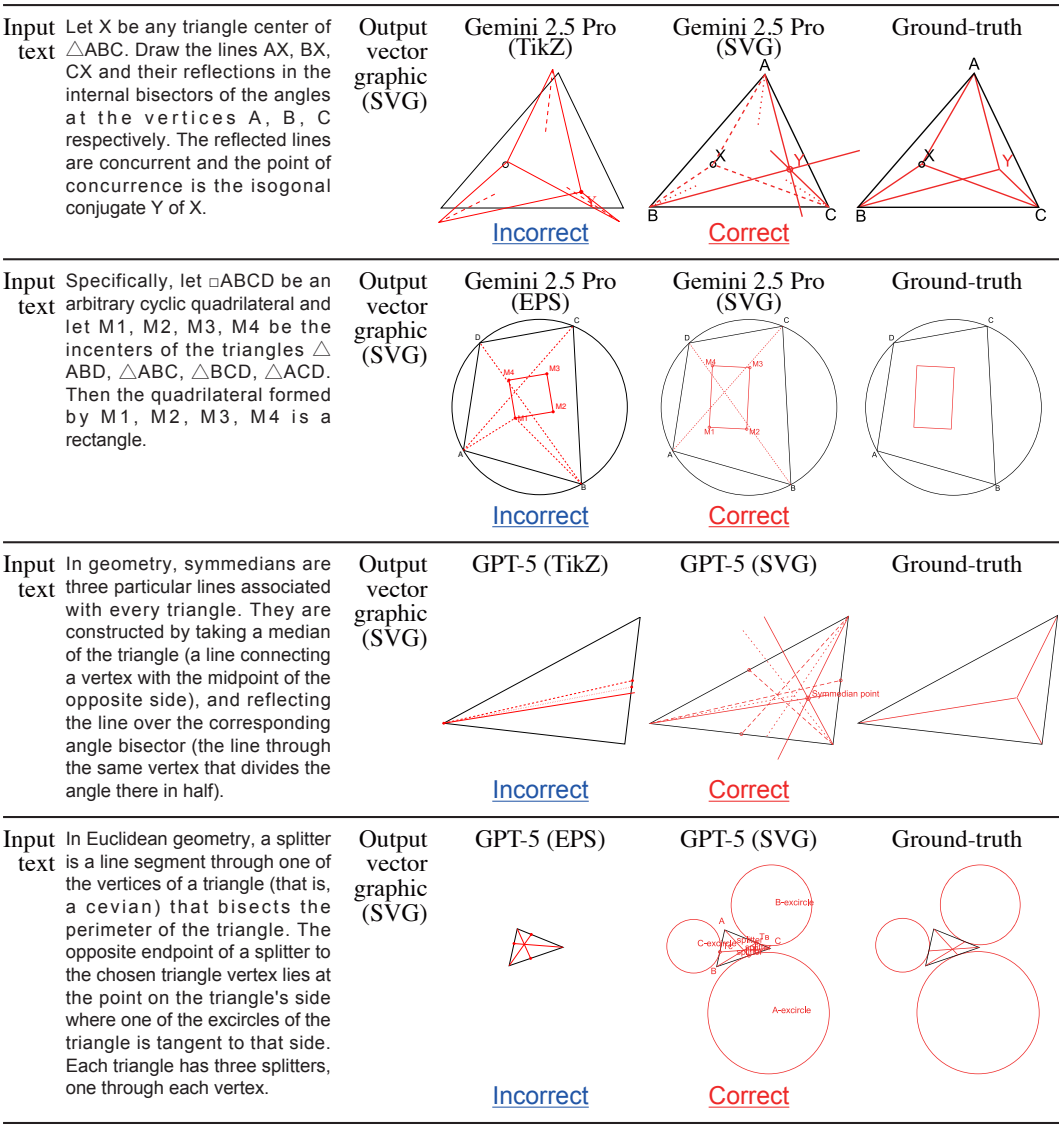


Figure 17: Examples where LLMs produce correct structures in SVG format but fail in TikZ and EPS formats in the plane geometry task.[11]

[11]The input texts and the ground-truth vector graphics are from (Wikipedia contributors, 2025h;i;l;k; Rocchini, 2008; Kmhkmh, 2024; 2016; Inductiveload, 2007b).

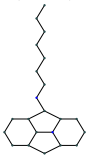
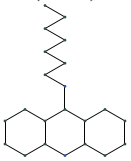
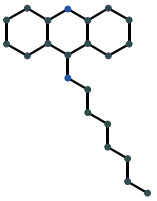
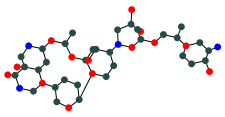
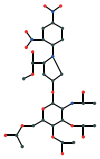
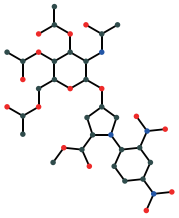
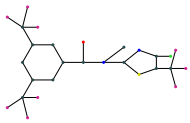
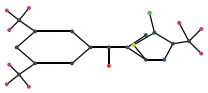
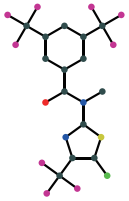
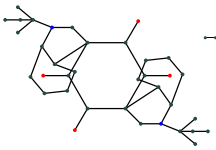
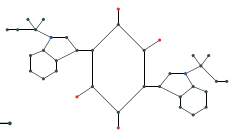
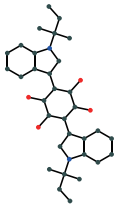
Input text	the molecular structure of the compound with the IUPAC name N-heptylacridin-9-amine	Output vector graphic (SVG)	Gemini 2.5 Pro (TikZ)  Incorrect	Gemini 2.5 Pro (SVG)  Correct	Ground-truth 
Input text	the molecular structure of the compound with the IUPAC name methyl 4-[3-acetamido-4,5-diacetyloxy-6-(acetyloxymethyl) oxan-2-yl]oxy-1-(2,4-dinitrophenyl) pyrrolidine-2-carboxylate	Output vector graphic (SVG)	Gemini 2.5 Pro (EPS)  Incorrect	Gemini 2.5 Pro (SVG)  Correct	Ground-truth 
Input text	the molecular structure of the compound with the IUPAC name N-[5-chloro-4-(trifluoromethyl) -1,3-thiazol-2-yl]-N-(trideuterio methyl)-3,5-bis(trifluoromethyl) benzamide	Output vector graphic (SVG)	GPT-5 (TikZ)  Incorrect	GPT-5 (SVG)  Correct	Ground-truth 
Input text	the molecular structure of the compound with the IUPAC name 2,5-dihydroxy-3,6-bis[1-(2-methylbut-3-en-2-yl)indol-3-yl] cyclohexa-2,5-diene-1,4-dione	Output vector graphic (SVG)	GPT-5 (EPS)  Incorrect	GPT-5 (SVG)  Correct	Ground-truth 

Figure 18: Examples where LLMs produce correct structures in SVG format but fail in TikZ and EPS formats in the molecular structure task.[12]

[12]The molecular structure data is from (National Center for Biotechnology Information, 2025d;h;b;a).

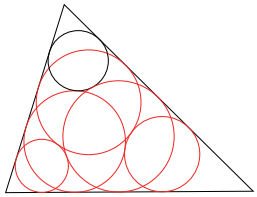
D.4 Comparison of Zero-shot Prompting Methods

We present results obtained by applying zero-shot prompting methods to Gemini 2.5 Pro in Figures 19, 20, 21, and 22. When using methods other than Zero-shot CoT, Gemini 2.5 Pro produces an explanation prior to generating vector graphics, and we include that explanation as well. In the case of Plan-and-Solve prompting, the output explanation "C3 will also be in corner B, tangent to C2" is incorrect, revealing that Gemini 2.5 Pro fails to capture the relationships. In the case of Step-Back prompting, the output explanation reveals that Gemini 2.5 Pro captures the necessary elements and relationships, but overlooks the importance of mathematical reasoning. In contrast, by applying our LOOP, Gemini 2.5 Pro captures the necessary elements and relationships and recognizes the importance of mathematical reasoning, enabling it to generate structurally correct vector graphics.

Input text and ground-truth

Input text: In geometry, the six circles theorem relates to a chain of six circles together with a triangle, such that each circle is tangent to two sides of the triangle and also to the preceding circle in the chain. The chain closes, in the sense that the sixth circle is always tangent to the first circle.

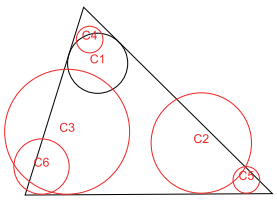
Ground-truth



Gemini 2.5 Pro w/ Zero-shot CoT

Output explanation: (There is no output explanation.)

Output vector graphic (SVG)



Incorrect

Figure 19: Example output generated by applying Zero-shot CoT prompting to Gemini 2.5 Pro.[13]

[13]The input text and the ground-truth vector graphic are from (Wikipedia contributors, 2022; Rocchini, 2010).

Gemini 2.5 Pro w/ Plan-and-Solve

Output explanation

An explanation of the six circles theorem has been provided, and it needs to be visualized by adding elements to the given SVG.

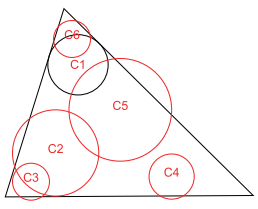
Plan:
1. **Deconstruct the Theorem**: The theorem describes a chain of six circles (C1 to C6) associated with a triangle.
   * There are three pairs of circles, with each pair nestled into one corner of the triangle.
   * Circles in the same corner are tangent to each other and to the two adjacent sides of the triangle.
   * The chain follows a sequence around the triangle's vertices: A -> B -> C -> A.
   * Let's say the given circle, C1, is in corner A.
   * The next circle, C2, will be in corner B and tangent to C1.
   * C3 will also be in corner B, tangent to C2.
   * C4 will be in corner C, tangent to C3.
   * C5 will also be in corner C, tangent to C4.
   * C6 will be in corner A, tangent to C5.
   * The theorem's conclusion is that C6 will also be tangent to the first circle, C1, thus closing the chain.

2. **Analyze the Input SVG**:
   * We have a triangle. Let's label its vertices A (top), B (bottom-left), and C (bottom-right).
   * We have one circle, which we will label C1. It is located in corner A.

3. **Add Visual Elements**:
   * Add the remaining five circles (C2, C3, C4, C5, C6) with the class `output_object`.
   * Place them according to the theorem's rules, ensuring they appear tangent to the correct triangle sides and preceding circles. The positions and radii will be estimated to create a visually plausible representation of the theorem.
   * C1 and C6 will be in corner A.
   * C2 and C3 will be in corner B.
   * C4 and C5 will be in corner C.
   * The crucial part is to draw the chain such that C6, which is tangent to C5, is also tangent to C1.
   * Add text labels "C1" through "C6" for each circle, assigning them the class `output_text`.

Execution:
The following SVG code adds the five circles and six labels to illustrate the theorem. The positions and sizes of the new circles have been chosen to create a clear and accurate visual representation of the tangency relationships described.

Output vector graphic (SVG)



Incorrect

Figure 20: Example output generated by applying Plan-and-Solve prompting to Gemini 2.5 Pro.

Gemini 2.5 Pro w/ Step-Back

Output explanation

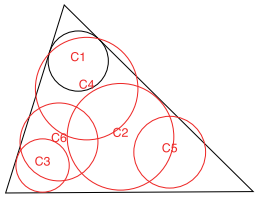
An explanation of the six circles theorem has been provided. This theorem describes a closed chain of six circles, where each circle is tangent to two sides of a triangle and the preceding circle in the chain. The provided SVG shows the triangle and the first circle of such a chain.

To visualize the theorem, we will add the remaining five circles to complete the chain.

1. **Circle C1:** The initial circle is provided, located in the top corner of the triangle.
2. **Circle C2:** We add the second circle, C2, in the bottom-right corner, making it tangent to the two sides of the triangle at that corner and also to circle C1.
3. **Circle C3:** The third circle, C3, is placed in the bottom-left corner, tangent to the two sides there and to circle C2.
4. **Circle C4:** The fourth circle, C4, is placed back in the top corner. It is tangent to the same two sides as C1, and also to circle C3.
5. **Circle C5:** The fifth circle, C5, is placed in the bottom-right corner, tangent to the sides and to circle C4.
6. **Circle C6:** The sixth and final circle, C6, is placed in the bottom-left corner, tangent to the sides and to circle C5.

The remarkable property of the theorem is that this chain always closes perfectly: circle C6 will not only be tangent to C5 but will also be tangent to the very first circle, C1, completing the loop. The visualization below illustrates this closed chain.

Output vector graphic (SVG)



Incorrect

Figure 21: Example output generated by applying Step-Back prompting to Gemini 2.5 Pro.

Gemini 2.5 Pro w/ LOOP (ours)

Output explanation

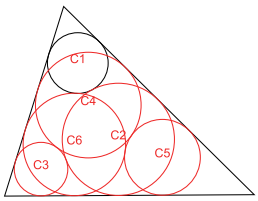
An elegant theorem in geometry, the six circles theorem, describes a fascinating property of a chain of circles within a triangle. Starting with a circle tangent to two sides of the triangle, a sequence of five more circles is constructed. Each new circle is tangent to two sides of the triangle and to the previous circle in the chain. The remarkable result is that this chain always forms a closed loop, with the sixth circle being perfectly tangent to the first one.

This visualization completes the chain described by the theorem, starting from the single provided circle.

1. **Information extraction**: The core components are a triangle and a chain of six circles. The initial SVG provides the triangle and the first circle (let's call it C1). C1 is tangent to two sides of the triangle at one vertex. The goal is to construct the remaining five circles, C2 through C6, following the theorem's rules.
2. **Relationship extraction**: The chain proceeds sequentially around the vertices of the triangle.
   * C1 is tangent to sides AB and AC (at vertex A).
   * C2 must be tangent to sides AC and BC (at vertex C) and also to C1.
   * C3 must be tangent to sides BC and AB (at vertex B) and also to C2.
   * C4 must be tangent to sides AB and AC (at vertex A) and also to C3.
   * C5 must be tangent to sides AC and BC (at vertex C) and also to C4.
   * C6 must be tangent to sides BC and AB (at vertex B) and also to C5.

   The theorem guarantees that the resulting C6 will also be tangent to the starting circle, C1.
3. **Mathematical reasoning**: The center of any circle tangent to two sides of the triangle must lie on the angle bisector of the corresponding vertex. The condition that the new circle must also be tangent to the previous circle provides a second constraint. Combining these two conditions leads to a solvable quadratic equation for the position of the new circle's center along the angle bisector. This process is applied iteratively to find the center and radius of each subsequent circle in the chain (C2, C3, C4, C5, and C6). The final calculation confirms the theorem: the distance between the centers of C6 and C1 is indeed equal to the sum of their radii, proving their tangency.
4. **Code generation**: Based on the calculated centers and radii, five `<circle>` elements are added to the SVG to represent C2 through C6. Additionally, `<text>` elements are added to label each circle from C1 to C6, clarifying the sequence of the chain.

Output vector graphic (SVG)



Correct

Figure 22: Example output generated by applying our LOOP to Gemini 2.5 Pro.