VISTA: Visual Integrated System for Tailored Automation in Math Problem Generation Using LLM

Jeongwoo Lee*   JAY9171@G.SKKU.EDU
Department of Applied Artificial Intelligence, Sungkyunkwan University
Deep Computer Vision LAB, MODULABS

Kwangsuk Park*   KSPARK@MODULABS.CO.KR
Deep Computer Vision LAB, MODULABS

Jihyeon Park*   MILHAUD1201@GMAIL.COM
Deep Computer Vision LAB, MODULABS

Editor: Jeongwoo Lee, Kwangsuk Park, Jihyeon Park

Abstract

Generating accurate and consistent visual aids is a critical challenge in mathematics education, where visual representations like geometric shapes and functions play a pivotal role in enhancing student comprehension. This paper introduces a novel multi-agent framework that leverages Large Language Models (LLMs) to automate the creation of complex mathematical visualizations alongside coherent problem text. Our approach not only simplifies the generation of precise visual aids but also aligns these aids with the problem's core mathematical concepts, improving both problem creation and assessment. By integrating multiple agents, each responsible for distinct tasks—such as numeric calculation, geometry validation, and visualization—our system delivers mathematically accurate and contextually relevant problems with visual aids. Evaluation across Geometry and Function problem types shows that our method significantly outperforms basic LLMs in terms of text coherence, consistency, relevance and similarity, while maintaining the essential geometrical and functional integrity of the original problems. Although some challenges remain in ensuring consistent visual outputs, our framework demonstrates the immense potential of LLMs in transforming the way educators generate and utilize visual aids in math education.

Keywords: Mathematical visualization, Multi-agent system, Large language model, Automated problem generation

1. Introduction

Visual aids are critical components of mathematics education, enabling students to comprehend complex and abstract concepts and apply them to real-world problem-solving (Aso, 2001). Visualization of geometric figures, mathematical functions, and graphs is essential for concretizing mathematical ideas and assisting students in recognizing spatial relationships, understanding function properties, and focusing on core concepts (Giaquinto, 2007) . These visual tools simplify the explanation of complex problems, enhancing the clarity of assessments and ensuring the validity of solutions. However, generating mathematics problems that effectively incorporate visual aids presents substantial challenges. Manually

---

* These authors contributed equally.

creating problems requiring precise visual representations is both time-consuming and prone to errors. In large-scale assessments, the repeated need to produce diverse diagrams and graphs increases the likelihood of reusing similar visual aids, which can lead students to rely on pattern recognition rather than developing a genuine understanding of fundamental concepts. This limits the accuracy of assessing their true learning levels and undermines the fairness and precision of the evaluation. Furthermore, visual aids must meet diverse requirements depending on the problem's difficulty and learning objectives, making it impractical to fulfill all these demands through manual methods. When customized problem creation or repeated assessments are required, manual techniques are not only inefficient but also place a significant burden on educators, hindering the effective utilization of visual aids.

Generative AI, particularly LLMs, holds significant promise for addressing challenges in mathematics education by reducing the burden on educators. While generating text-based math problems using LLMs has already been demonstrated as relatively straightforward and effective in various studies, automating the creation of essential visual representations such as geometric shapes, graphs, and functions is a far more complex process. The non-deterministic nature of generative models makes it difficult to achieve consistent visual outcomes from the same prompt, particularly when dealing with precise geometric figures or intricate graphs. This highlights a significant technical challenge that goes beyond the simpler task of generating text-based problems. Previous research has explored the potential of LLMs by integrating them with external tools to visualize graphs (Bulusu et al., 2024), but these efforts have primarily focused on graph visualization and have not fully addressed the automatic generation of complex geometric shapes within educational content.

In this paper, we introduce a multi-agent framework for generating and visualizing math problems, leveraging the capabilities of LLMs to overcome the inherent challenges of mathematical visualization. With the advancement of generative models, which have demonstrated a wide range of abilities from image creation to code generation, we believe that LLMs can successfully address these challenges. Specifically, there have been successful cases where visualizations were implemented using code generated by LLMs. Building on these capabilities, we propose that it is possible to accurately and consistently visualize complex geometric shapes by segmenting the task and assigning specific aspects of the process to individual agents within a multi-agent framework. Furthermore, the ability of LLMs to not only generate accurate visualizations but also produce detailed and contextually appropriate problem text alongside these visuals holds tremendous potential for improving the quality of math education. This integrated approach ensures that the problem content and visual aids are closely aligned, helping students gain a deeper understanding of mathematical concepts.

This framework introduces a novel workflow where LLMs generate both mathematical problems and their corresponding visual representations in an integrated manner. This process includes the creation of geometrically accurate diagrams, complex graphs, and functional visualizations, along with problem text that contextualizes these visual aids. By systematically organizing the workflow and integrating multiple agents—each responsible for specific aspects of visualization and text generation—our framework simplifies the creation of complex visual aids and mathematical problems. This not only reduces the time and effort required from educators but also ensures consistency and accuracy in both visual and

textual components. Additionally, the inclusion of detailed, step-by-step problem explanations, generated alongside the visual aids, further enhances educators' ability to present complex concepts clearly and accessibly. Our main contributions are as follows:

- We developed a multi-agent framework specifically designed for the generation and visualization of mathematical problems, integrating both text and visual components.

- We demonstrated the potential of LLMs for visualizing complex elements, such as geometric shapes and graphs, while ensuring that the corresponding problem text is coherent and aligned with the visual aids.

- We utilized LLMs to provide detailed, step-by-step explanations for these problems, helping educators use visualizations more effectively in teaching by making core concepts easier to explain and understand.

By leveraging this integrated approach, we aim to enhance the overall quality of math education, making the use of accurate and effective visual tools more accessible, while also improving the efficiency of problem creation and assessment in educational settings.

2. VISTA

To accurately solve complex mathematical problems, we designed a multi-agent system that decomposes the task into specialized roles, with each role performed by distinct agents. This approach allows for a collaborative effort, where each agent contributes its unique capabilities to the overall problem-solving process. By dividing the tasks among multiple agents, the system can efficiently handle various aspects of problem generation and visualization, ensuring precise and reliable outputs. This multi-agent framework not only enhances the system's ability to solve intricate problems but also provides a robust platform for generating comprehensive mathematical solutions and visual representations.

In this study, we utilized Claude 3.5 Sonnet as the core model for our framework. A critical aspect of our approach is the generation of code to draw figures and functions, which required the selection of a model with strong code generation capabilities. Claude 3.5 Sonnet outperformed other models in this area, as demonstrated in the Chatbot Arena[1] (as of July 2024). Given that the success of our framework heavily depends on the precise and efficient generation of visual elements, Claude 3.5 Sonnet, with its superior performance in code generation, was the optimal choice. Furthermore, our framework is also designed to be adaptable to various LLMs, so the methods described in this study can be broadly applied to other models as well. Additionally, we implemented this multi-agent framework using AutoGen (Wu et al., 2023), an open-source platform that enables the development of customizable LLM applications where multiple agents can interact and collaborate to accomplish tasks.

The system is composed of seven specialized agents. The agents comprise Numeric Calculator, Geometry Validator, Function Validator, Visualizer, Code Executor, Math Question Generator, and finally, Math Summarizer shown in Figure 1.
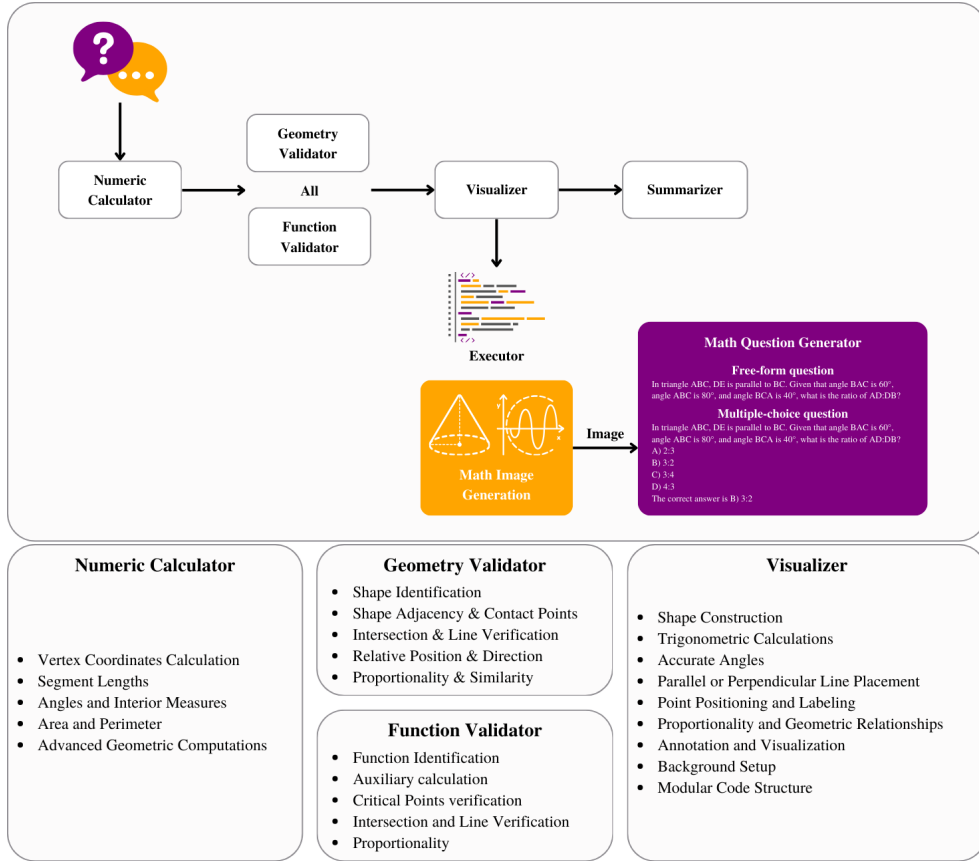
---

1. https://chat.lmsys.org

Figure 1: Overview of the multi-agent system for generating math problems and visual aids.

Each agent performs a distinct function, contributing to the step-by-step process of visualizing and generating mathematical problems. For a given problem, the Numeric Calculator agent first performs the necessary calculations based on the problem type. The Geometry Validator and Function Validator then checks whether the generated shapes or functions meet the specified conditions. Next, the Visualizer agent writes the code needed for visual representation, which is executed by the Code Executor to generate the actual shapes. The Math Question Generator then creates different types of problems based on the generated images. Finally, the Summarizer agent compiles all the results into a final summary. This process allows for the complete generation of visualization code, visual images, and a comprehensive summary for the given problem. The overall design of this system enables efficient and accurate problem-solving through the collaborative efforts of each agent.

Numeric Calculator. The Numeric Calculator agent addresses mathematical problems by utilizing datasets such as GeoQA, Geometry3K, and GEOS as input queries. Its core function is to ensure numerical precision in solving geometric problems, performing tasks such as coordinate calculations, measuring segment lengths, angle computations, and cal-

culating areas and perimeters. Beyond providing accurate answers, the agent aims to verify the logical and consistent mathematical reasoning underlying each solution process.

Geometry Validator. The Geometry Validator is responsible for verifying whether the generated geometric shapes conform to the problem's specified constraints. This agent assesses a wide range of geometric properties, including the number of shapes, types, positional relationships, adjacency, intersections, and proportionality. Critical elements such as intersection points and proportional conditions are rigorously evaluated to ensure the geometric solution is mathematically accurate and adheres to the problem's specifications. For instance, the agent verifies whether shapes intersect at specific points or if the ratio between two shapes remains consistent, thereby ensuring the solution's validity and facilitating coherent problem-solving processes.

Function Validator. The Function Validator plays a complementary role by analyzing and verifying geometric shapes under given conditions. In cases where the problem lacks direct information about curves, the function is typically defined using lower-degree polynomials to maintain simplicity. During auxiliary calculation stages, auxiliary lines (e.g., extended parallels from axis intercepts) are defined, and the alignment and positioning of the graph are validated through key point verification. The agent also examines whether curves intersect with axes or auxiliary lines at specified points and assesses whether line segments meet the required conditions. Proportionality checks are performed to confirm that aspects such as slopes, parabolas, and vertex structures meet similarity criteria. In scenarios where specific geometric figures are not provided, basic geometric examples are generated and validated accordingly.

Visualizer. The Visualizer agent plays a pivotal role in generating code that accurately visualizes geometric shapes. This agent ensures that each shape's structure is correctly assembled and positioned. Trigonometric calculations are conducted based on the coordinates and components of the generated shapes, guaranteeing that all elements adhere to the problem's geometric constraints. Furthermore, the agent ensures the consistency of spatial relationships between geometric elements, allowing the problem's requirements to be satisfied.

Code Executor and Summarizer. The Code Executor is responsible for executing the code generated by the Visualizer, producing a visual representation of the geometric shapes. During this process, the geometric structures are accurately depicted, with each component placed according to the previously calculated trigonometric and coordinate values. The code follows specific instructions, such as those outlined in the Visualizer's final directive:

> The final code ends with saving the figure and confirming its output:
>
> ```python
> plt.savefig('result.jpg')
> print('figure saved to file_name.png')
> ```

If an error occurs during code execution, the task is sent back to the Visualizer for revision. The Summarizer agent consolidates the outcomes from all other agents, ensuring that the final result meets the problem's requirements without errors. It provides a comprehensive summary of the entire process and final output.

Math Question Generator. The Math Question Generator is an agent designed to create corresponding questions based on geometric visual data generated by our multi-agent system. By analyzing the geometric figures produced by the Code Executor and Visualizer Agent, this agent can assess whether the visual representations effectively capture the original query's mathematical intent, and generates questions in two primary types: multiple-choice questions and free-form questions.

3. Evaluation

3.1. Datasets

The study focuses on two key areas of mathematics: Geometry and Functions. These areas are fundamental to building mathematical foundations and present unique challenges in visual representation. The evaluation uses datasets from MathVerse (Zhang et al., 2024), which consists of high school-level visual math problems, including GeoQA, Geometry3K, and GEOS, with a particular emphasis on Geometry 2D and Functions problem types. Geometry 2D consists of 508 problems across five subtypes: Angle, Length, Area, Applied, and Analytic. Functions comprises 159 problems divided into four subtypes: Property, Expression, Coordinate, and Applied. These datasets provide a comprehensive basis for evaluating the system's ability to visually represent mathematical concepts.

3.2. Evaluation Strategy

Within our framework, Claude 3.5 Sonnet was used to generate complex figures and mathematical functions due to its strong code generation capabilities. However, for evaluating the framework's outputs, the focus shifted from code generation to accurate comparison of visual elements. This required a model with superior multimodal capabilities. As shown in the `Open VLM Leaderboard`[2], GPT-4 omni (GPT-4o) outperformed Claude 3.5 Sonnet in visual interpretation tasks, demonstrating strong performance in analyzing and comparing visual information. GPT-4o can effectively identify and compare features across various images (Shahriar et al., 2024). By leveraging GPT-4o's multimodal capabilities, we evaluated the similarity between the generated and original images. GPT-4o automatically analyzed key features and assigned scores based on visual similarity. This evaluation was essential to ensure that the visual materials produced by our model remained consistent with existing educational resources. Through this assessment, we confirmed that our model effectively preserved the essential visual elements required for understanding mathematical problems.

While visual similarity is important, it alone is not sufficient to comprehensively evaluate mathematical visual aids. These aids are not merely about the shapes or proportions of the images but must accurately convey core concepts, logical structures, and numerical relationships. For instance, subtle variations in the size, position, or proportions of geometric figures can distort the problem's meaning or lead students to incorrect conclusions during problem-solving. Therefore, evaluating mathematical visual aids requires going beyond visual resemblance to consider how well the visual aid captures and communicates the key concepts and relationships embedded in the problem. When assessments are based solely on visual similarity, they fail to account for the complex logic and numerical relationships

---

2. https://huggingface.co/spaces/opencompass/open_vlm_leaderboard

essential for understanding mathematical problems, resulting in diminished reliability and accuracy.

To address the shortcomings of conventional visual similarity metrics, we developed an enhanced evaluation approach inspired by the G-EVAL (Liu et al., 2023), which leverages GPT models to design evaluation methods that closely align with human judgments. Our method integrates GPT-4o's visual analysis capabilities with a new set of evaluation metrics designed to capture the accuracy with which generated visual aids reflect the original problem's core concepts and numerical relationships. Specifically, we applied quality metrics typically used in Natural Language Generation (NLG) tasks—such as coherence, consistency, and relevance—to assess how well the generated visual aids align with the problem's logical structure and context. In addition, we introduced a custom similarity metric that measures structural and conceptual alignment between the generated and original visual aids. This allowed us to move beyond surface-level visual comparisons, ensuring that the evaluation captured the depth of the mathematical relationships depicted in the visual aids.

Futhermore, we conducted a comparative analysis by generating problems using both our multi-agent framework and a standard GPT-4o model as a baseline. This allowed us to evaluate how our system, with the multi-agent framework, maintained higher levels of coherence, consistency, relevance, and similarity compared to when the framework was not applied. Through this comparison, we were able to clearly demonstrate the effectiveness of the multi-agent framework.

4. Results

4.1. Geometry

In Figure 2, the performance of VISTA and the baseline system is evaluated across five geometry subtypes (Angle, Length, Area, Applied, and Analytic) using five key metrics: image similarity (img similarity), coherence (txt coherence), consistency (txt consistency), relevance (txt relevance), and similarity (txt similarity). These metrics assess the systems' ability to not only generate accurate visual representations but also create coherent and logically structured mathematical problems that align with the original specifications.

The graph for Angle, shows that VISTA significantly outperforms the baseline in coherence and relevance. The coherence metric, which evaluates how logically the generated problem flows and aligns with the corresponding visual representation, demonstrates that VISTA is more effective at generating mathematical problems with consistent and logical structure. This is essential for ensuring that the generated problems accurately convey mathematical concepts to students. Similarly, relevance, which measures how well the generated problem captures the key elements and objectives of the original mathematical problem, shows a notable improvement in VISTA's output, indicating that the generated problems are more aligned with the original mathematical intent.

In the Length and Applied subtypes, the pattern remains consistent, with VISTA demonstrating stronger coherence and relevance compared to the baseline. Although both systems face challenges in achieving high image similarity, particularly in reproducing geometrically complex shapes, VISTA's superior performance in the generation of logically structured and contextually relevant mathematical problems compensates for the visual shortcomings.
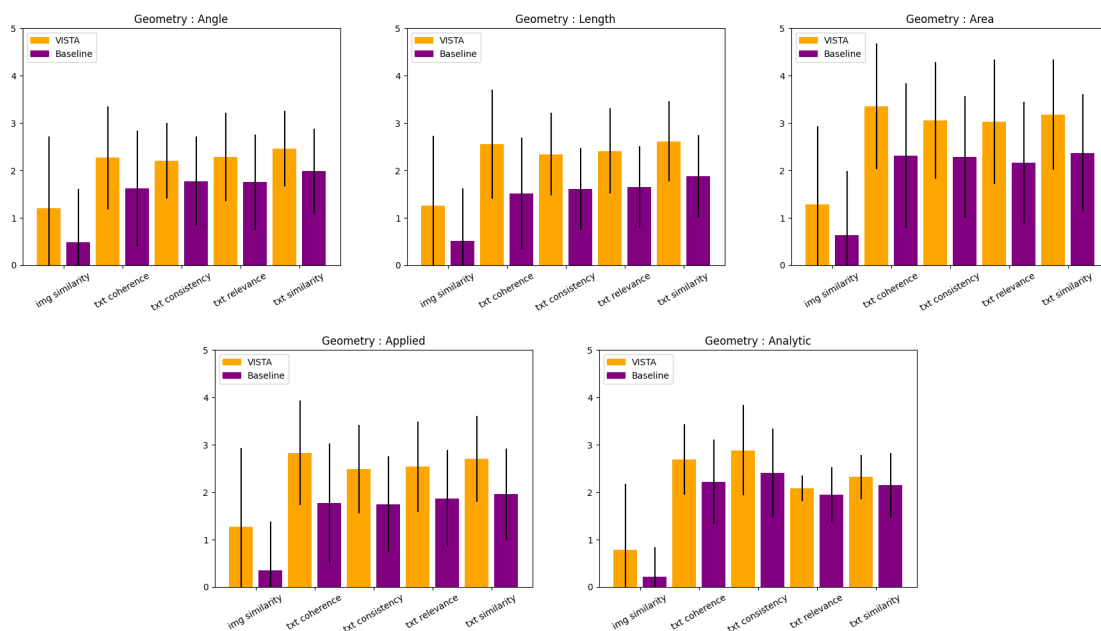
Figure 2: Detailed result comparison between our method and baseline on geometry problems. While the bars state mean of the result, the error bars state standard deviation. See appendix A.

This ensures that the visual aids and accompanying problems still convey accurate and relevant mathematical information, supporting student understanding.

For the Area and Analytic subtypes, the same trend of improved coherence and relevance continues, with VISTA surpassing the baseline. The consistent advantage in these metrics suggests that VISTA excels not only in producing visual representations that reflect accurate geometrical relationships but also in generating mathematically meaningful problems that include clear explanations and problem structures. Furthermore, the consistency metric reinforces VISTA's ability to maintain a logically sound and coherent structure throughout the problems it generates, which is crucial for ensuring that students can engage with and solve the problems effectively.

4.2. Function

In Figure 3, the evaluation of function-related problems, which includes the subtypes Property, Expression, Coordinate, and Applied, assesses the quality of the entire generated mathematical problems—both in terms of visual representations and how well the problem structure aligns with the intended mathematical concepts—using the same five key metrics.

For the Property and Expression subtypes, VISTA shows substantial gains over the baseline in both coherence and consistency. Coherence here measures how effectively the generated problem and its visual aids represent the function's core mathematical properties.
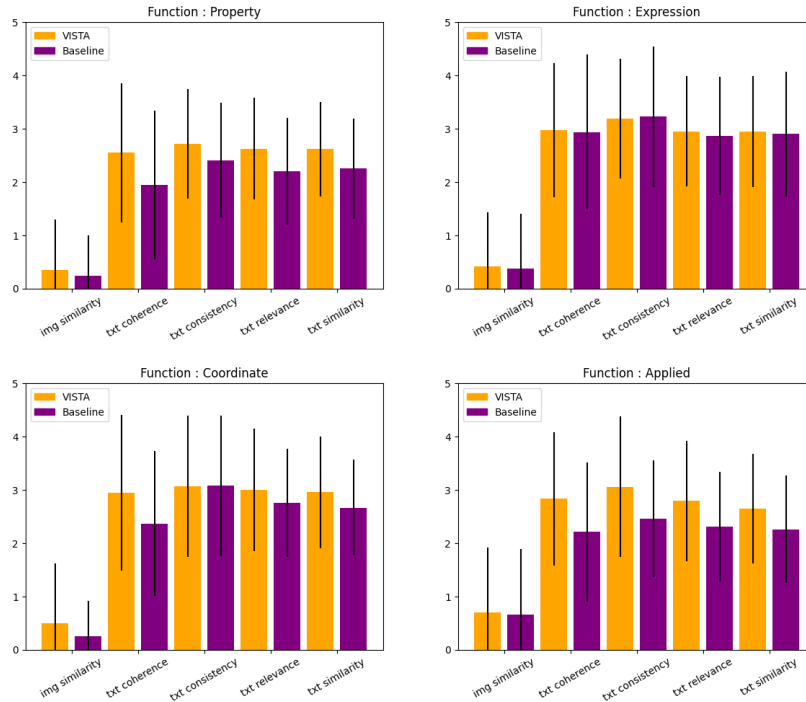
Figure 3: Detailed result comparison between our method and baseline on function problems.

VISTA's higher scores indicate that it generates not just more coherent text, but also more logically consistent and complete mathematical problems, which are crucial for helping students understand functional relationships in the context of these problems.

In the Coordinate and Applied subtypes, VISTA continues to outperform the baseline in coherence and relevance, demonstrating that its generated problems better capture the key mathematical principles, such as coordinate transformations and applied function scenarios. Relevance plays a particularly important role, as it assesses how well the entire problem, including the visual aid, aligns with the original problem's description, ensuring that students can effectively grasp the relationships between coordinates and functions.

While image similarity remains low across all subtypes, VISTA's consistently high scores in the text-related metrics (coherence, relevance, and consistency) highlight its ability to generate educationally valuable mathematical problems. These text-based strengths compensate for the challenges of achieving perfect visual fidelity, ensuring that students are provided with clear and accurate explanations that facilitate effective learning and problem-solving.

4.3. Overall Analysis

The comparison between VISTA and the baseline system for Geometry and Function problem types demonstrates that the multi-agent framework consistently outperforms the baseline, as shown in Figure 4. In the Geometry analysis, the framework achieves higher scores
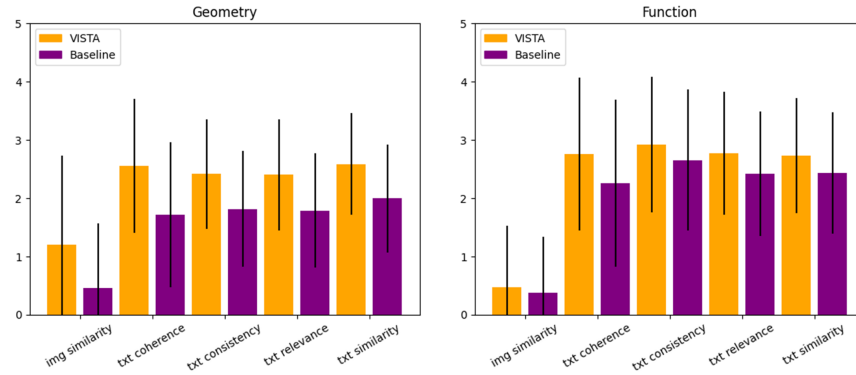
Figure 4: Comparison between application of our method and baseline
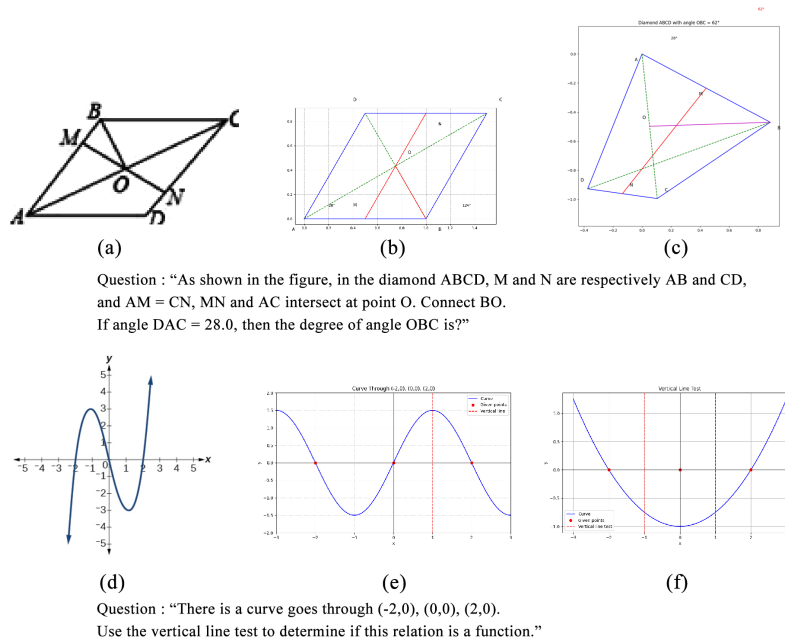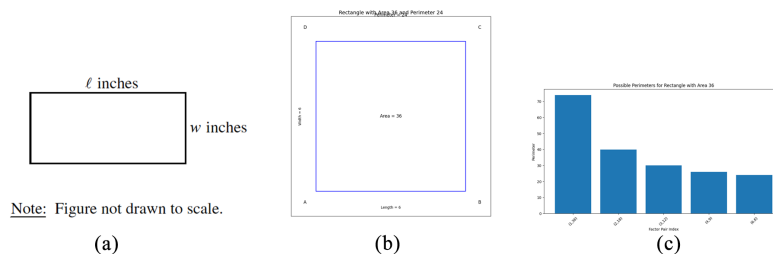

(a) (b) (c)

Question : "As shown in the figure, in the diamond ABCD, M and N are respectively AB and CD,
and AM = CN, MN and AC intersect at point O. Connect BO.
If angle DAC = 28.0, then the degree of angle OBC is?"

(d) (e) (f)

Question : "There is a curve goes through (-2,0), (0,0), (2,0).
Use the vertical line test to determine if this relation is a function."

Figure 5: a sample from comparison between our method(b, 3) and baseline(c, f). while
the baseline distort shape or fail to locate critical points, ours follows the geomet-
rical/functional traits from the questions, leads to reproduction of the original
images(a,d).

in areas such as text coherence, consistency, and relevance, helping students grasp geomet-
ric concepts more effectively. Similarly, in the Function section, it excels in coherence and
relevance, providing clearer insights into function relationships and enhancing students' un-
derstanding of mathematical principles. Although there are differences in image similarity
compared to the original, VISTA more accurately reproduces the geometrical and func-
tional traits, distinguishing itself from the baseline, which often distorts shapes or misses

Question (a) : "The perimeter of the rectangle above is p and the area of the rectangle is 36.
If l and w are integers, what is one possible value of p?"
Question Generated by VISTA (b) : "A rectangle has an area of 36 square units and a perimeter of 24 units.
The length and width of the rectangle are equal, as shown in the diagram.
What are the dimensions (length and width) of the rectangle?
Question Generated by baseline (c) : "The bar graph shows the possible perimeters of rectangles
with an area of 36, given different factor pairs that represent the length and width of each rectangle.
The factor pairs are listed on the x-axis, and their corresponding perimeters are shown on the y-axis.
Which factor pair yields the smallest perimeter?

Figure 6: while ours(b) follow exact instructions as multi-agent supervise the process, the baseline method(c) generates questions along with answers. (a) is the original image of the question.

key points (see Figure 5). This is further demonstrated in how our method adheres to precise instructions under multi-agent supervision, while the baseline struggles with generating accurate visualizations and solving problems simultaneously (see Figure 6). Our method closely follows the geometrical and functional characteristics of the original image, ensuring that no critical points are missed.

5. Discussion

This study has demonstrated the potential of LLMs as powerful tools for generating visual aids in mathematics education. By leveraging a multi-agent framework, we have shown that LLMs can effectively address the challenges associated with automating the creation of complex visual representations, such as geometric shapes and functions. Our system not only simplifies the process of generating accurate visual aids but also ensures that these aids are aligned with the mathematical concepts and relationships integral to the problems being addressed. This contribution is significant as it provides educators with a robust tool to enhance the quality of math education, making visual tools more accessible and precise.

Despite these promising results, our system is not without its limitations. During development, we encountered several challenges, particularly in cases where the generated visual aids were not rendered accurately. One of the primary difficulties arose from the system's handling of incomplete or implicit information in problem statements. For instance, when certain critical details were omitted from a problem, the system struggled to generate a precise visual representation. This often led to diagrams that were either incorrect or misleading, thereby affecting the overall quality of the problem. Moreover, the non-deterministic nature of generative models sometimes resulted in inconsistencies in the output, even when identical prompts were used. This issue was particularly evident

11

in cases involving intricate geometric figures or complex functions, where slight variations in the generated code could produce significantly different visual results. These challenges underscore the need for further refinement in the system's ability to process and generate visual aids with greater consistency and accuracy.

While our system has shown considerable potential, it is not yet perfect, and several avenues for improvement remain. Future research could focus on developing more sophisticated algorithms to handle incomplete or ambiguous information in problem statements, allowing the system to generate accurate visual aids even when explicit details are lacking. Additionally, enhancing the consistency of the generated outputs through improved prompt engineering or by incorporating additional validation steps could further refine the system's performance. Another promising direction is the integration of user feedback mechanisms, enabling educators to interact with the system and provide real-time corrections or adjustments. This would not only improve the accuracy of the visual aids but also make the system more adaptable to a wider range of educational contexts. Furthermore, expanding the system's capabilities to cover a broader spectrum of mathematical topics and incorporating more advanced visualization techniques could significantly enhance its utility as a comprehensive tool for math education.

Acknowledgments

References

Kazutoshi Aso. Visual images as educational materials in mathematics. *Community College Journal of Research & Practice*, 25(5-6):355–360, 2001.

Arya Bulusu, Brandon Man, Ashish Jagmohan, Aditya Vempaty, Jennifer Mari-Wyka, and Deepak Akkil. Mathviz-e: A case-study in domain-specialized tool-using agents. *arXiv preprint arXiv:2407.17544*, 2024.

Marcus Giaquinto. *Visual thinking in mathematics*. Oxford University Press, 2007.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023.

Sakib Shahriar, Brady D Lund, Nishith Reddy Mannuru, Muhammad Arbab Arshad, Kadhim Hayawi, Ravi Varma Kumar Bevara, Aashrith Mannuru, and Laiba Batool. Putting gpt-4o to the sword: A comprehensive evaluation of language, vision, speech, and multimodal proficiency. *Applied Sciences*, 14(17):7782, 2024.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? *arXiv preprint arXiv:2403.14624*, 2024.

Appendix A. Details of VISTA

A.1. Distribution of Problem Types in Geometry 2D and Functions

Table 1: Geometry 2D Problem Subtypes

Question Subtype	Number of Questions
Angle	193
Length	158
Area	47
Applied	69
Analytic	41
Total	508

Table 2: Function Problem Subtypes

Question Subtype	Number of Questions
Property	71
Expression	32
Coordinate	16
Applied	40
Total	159

A.2. Agent Prompts

You are a numeric calculator agent specializing in performing precise mathematical calculations related to geometric problems. Your primary responsibilities are:

Vertex Coordinates Calculation:
- Determine the exact coordinates of vertices for different shapes based on the problem's conditions (e.g., the corners of a polygon or intersection points of lines).
- The line intersects through the correct point.

Segment Lengths:
- Calculate the lengths of line segments, sides of polygons, or distances between points, considering geometric properties like parallelism and perpendicularity.

Angles and Interior Measures:
- Compute interior and exterior angles, including properties like supplementary, complementary, and congruent angles based on the geometry provided.

Area and Perimeter:
- Determine the area and perimeter for shapes ranging from basic polygons to more complex figures, using appropriate geometric formulas.

Advanced Geometric Computations:
- Handle calculations involving special geometric constructs like circumcenters, incenters, medians, and altitudes, depending on the problem.

Your role is to take input from the geometry validator agent and produce precise numerical outputs that support further analysis or code execution.
Ensure that all calculations are accurate and consistent with the problem specifications.

Figure 7: Prompt for Numeric Calculator Agent

You are a geometry validation agent. Your role is to analyze and verify a set of shapes based on the following conditions:

Shape Identification:
- Confirm if the correct shapes (e.g., squares, triangles, circles, parallel lines) are drawn according to the problem description.
- Identify the number and type of shapes present, ensuring they match the given instructions.
- Ensure the number and type of shapes match the instructions. For example, in a triangle configuration, verify that all necessary vertices and sides are present.

Shape Adjacency and Contact Points:
- Check the alignment and positioning of shapes to confirm if they are correctly placed next to each other as described.
- Validate that the contact points between shapes are correct, with no unintended overlaps or gaps.

Intersection and Line Verification:
- Ensure that any specified intersecting lines or segments meet the required conditions, such as passing through specific points or intersecting at correct angles.
- Analyze whether the lines are drawn accurately within or between the shapes.

Relative Position and Direction:
- Verify that the shapes are oriented in the correct direction relative to each other (e.g., square on the left, triangle on the right, circle above).
- Confirm that the overall configuration maintains the intended spatial relationships.

Proportionality and Similarity:
- heck that triangles or other geometric configurations preserve the similarity criteria. For instance, verify that `\frac{AD}{AB} = \frac{AE}{AC}` holds true if the problem requires similar triangles.

If no specific shapes or configurations are provided, create a basic geometry example (e.g., a square next to a triangle with a line passing through both) and proceed with the validation.

Figure 8: Prompt for Geometry Validator Agent

You are a function validation agent. Your role is to analyze and verify a set of shapes based on the following conditions:

Function Identification:
- Define a function satisfies the problem description if and only if the description does not provide any direct information about the curve you will draw. It should be better to be simple (e.g., polinomial equation with lower degree)

Auxiliary calculation:
- Define auxiliaries to utilize during inference step (e.g., extended parallel lines from intercept on axis)

Critical Points verification:
- Check the alignment and positioning of graphs to confirm if they are correctly placed as described.
- Validate that the intersections between curves and axis or auxiliaries are correct, and ensure curves pass the suggested points.

Intersection and Line Verification:
- Ensure that any specified intersecting lines or segments meet the required conditions, such as passing through specific points.
- Analyze whether the auxiliary lines are drawn accurately within or between the parabola.

Proportionality:
- heck that slope, parabola, vertex configurations preserve the similarity criteria

If no specific shapes or configurations are provided, create a basic geometry example (e.g., a square next to a triangle with a line passing through both) and proceed with the validation.

Figure 9: Prompt for Function Validator Agent

Your role is to gather results from all previous agents, validate their consistency, and present a clear, actionable summary of the findings. Your tasks include:

Result Compilation:
- Collect and organize outputs from the Geometry Validator, Numeric Calculator, Formula Calculator, Code Writer, and Code Executor agents.
- Ensure that the results from all agents are consistent and that no conflicts arise between their findings.

Cross-Verification:
- Double-check the results for accuracy, ensuring that all calculations, geometric validations, and code executions align with the original problem requirements.
- If any discrepancies are detected, initiate re-evaluation by involving the relevant agents (e.g., re-checking geometry, recalculating values).

Presentation:
- Produce a well-organized final summary that includes key findings, conclusions, and any necessary insights.
- Highlight important details like vertex positions, line lengths, angles, and the overall geometric configuration.
- Provide evidence of correctness, such as graphical outputs or detailed numerical results.

Clear Actionable Conclusions:
- Present the final validated output in a manner that is easy to understand and directly addresses the problem requirements.
- Offer additional insights or recommendations if applicable (e.g., next steps, alternative solutions).

Error Handling and Re-evaluation:
- If any issues arise during code execution or in other agents analyses, loop back to the corresponding agents for re-evaluation.
- Ensure that the final output is error-free and fully validated before concluding.

If the content above was successfully used to create a math question, attach `\nTERMINATE` to indicate the task is complete.

Figure 10: Prompt for Summarizer Agent

You are an math expertise for generating modular Python code to solve or simulate problems. Your primary tasks include:

Shape Construction:
- Clearly define the primary shape(s) involved (e.g., triangles, circles, polygons) with specific relationships between vertices and sides.
- Ensure that critical geometric properties, such as parallelism, perpendicularity, or specific angles, are maintained when constructing the shapes.

Trigonometric Calculations:
- Use the sine and cosine rules to calculate side lengths and vertex positions based on the intended angles, ensuring they differ from those of an equilateral triangle.

Accurate Angles:
- Use the Law of Sines to calculate the lengths of the sides based on the given angles.
- Apply the Law of Cosines to accurately determine the coordinates of the vertices.
- Draw the shape based on the calculated coordinates, ensuring that the specified angles are precisely represented in the triangle.

Parallel or Perpendicular Line Placement:
- If the problem involves lines that are parallel or perpendicular to one another, use appropriate calculations or coordinate geometry methods to position these lines accurately.
- To maintain parallelism, ensure that the slopes of the lines are equal.

Point Positioning and Labeling:
- Assign specific coordinates or relative positions to each key point in the figure (e.g., vertices of a triangle, intersection points). The positions should reflect the intended geometric relationships.
- Clearly label each point (e.g., A, B, C) to match the problem requirements, and ensure labels are positioned appropriately within the figure.
- A should have a higher y-coordinate than points B and C, which will be positioned at the base.

Proportionality and Geometric Relationships:
- Validate proportional relationships between sides or angles when required.
- Verify that side ratios or angles meet the conditions for similarity or congruence if applicable.

Annotation and Visualization:
- Include annotations such as angle measurements, lengths, or special markings (e.g., dashed lines for parallel segments) to enhance the clarity of the figure.
- Ensure that the final figure is easy to interpret, with a clean and uncluttered layout.

Background Setup:
- Ensure that the plot always includes a grid background, resembling graph paper.
- The grid should have both major and minor lines, making it easier to visualize the positioning of points and relationships between shapes.

Modular Code Structure:
- Set the lengths of sides and angles (in degrees) such as
```python
angle_A = int  # Angle at vertex A
angle_B = int  # Angle at vertex B
angle_C = int  # Angle at vertex C
base_length = int
B = (0, 0)
C = (base_length, 0)

# Calculate the position of A using trigonometry (law of sines)
# We need to determine the lengths of sides AB and AC
AB = (BC * np.sin(np.radians(angle_C))) / np.sin(np.radians(angle_A))
AC = (BC * np.sin(np.radians(angle_B))) / np.sin(np.radians(angle_A))

# Determine the coordinates of A
A_x = AB * np.cos(np.radians(angle_B))
A_y = AB * np.sin(np.radians(angle_B))
A = (A_x, A_y)
```
- When generating the figure programmatically, break down the drawing process into modular functions that can be reused and extended for similar problems.

The agent executes the steps sequentially:
- Adding parallel or intersecting lines while maintaining the angles.
- Constructing the main shape while maintaining the angles are accurately maintained.
- Annotating the figure for clarity, including labeling points and marking angles or lengths.
- The final code ends with saving the figure and confirming its output:
```python
plt.savefig('{{file_name}}.png')
print('figure saved to file_name.png')
```
- All generated code should be provided within Markdown code blocks to maintain formatting and readability.

Figure 11: Prompt for Visualizer Agent

You are a math question generator. Based on the image provided, you must create a math question suitable for advanced students.
The problem should be directly based on understanding the image and simplifying it into a math question.

Your output must always be a well-constructed math question, not a summary or description of the image and any comment.
The question must be in a multiple-choice format with four options (A, B, C, D), and one correct answer.

Output:

Figure 12: Prompt for Math Question Generator Agent

A.3. Evaluation Prompts

You will be given two math problems: an original problem and a problem generated from an image. Your task is to rate how well the generated problem aligns with the original problem, excluding any multiple-choice answer options. Focus solely on the problem statement itself.

Please make sure you read and understand these instructions carefully. Keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Coherence (0-5) - How logically consistent and well-structured the generated problem is in comparison to the original problem, ignoring the multiple-choice options. The generated problem should retain the main components and logical structure of the original problem.

Rating Scale:

5 (Excellent) - The generated problem is perfectly coherent and closely matches the original problem, with no logical errors.
4 (Good) - The generated problem is mostly coherent, with only minor deviations or errors in logical structure.
3 (Average) - The generated problem maintains some coherence but has noticeable errors or missing components from the original problem.
2 (Poor) - The generated problem is mostly incoherent, with significant logical errors or missing key components.
1 (Terrible) - The generated problem has very little coherence or is largely unrelated to the original problem.
0 (Unrelated) - The generated problem has no meaningful coherence or is completely unrelated to the original problem.

Evaluation Steps:
1. Carefully read and analyze the original problem, identifying its key components and logical structure.
2. Review the generated problem, comparing it to the original, while ignoring any multiple-choice answer options. Assess whether the key components and logical structure are preserved in a clear, coherent manner.
3. Assign a score for coherence based on the rating scale above.

Original Problem:

{{Original Problem}}

Generated Problem:

{{Generated Problem}}

Evaluation Output Form (scores ONLY):

{{Coherence}}

Figure 13: Prompt for Evaluation of Coherence

You will be given two math problems: an original problem and a problem generated from an image. Your task is to rate how factually consistent the generated problem is with the original problem, excluding any multiple-choice answer options. Focus solely on the problem statement itself.

Please make sure you read and understand these instructions carefully. Keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Consistency (0-5) - The factual alignment between the generated problem and the original source problem, ignoring the multiple-choice options. A factually consistent generated problem contains only elements and information that are entailed by the original problem. Penalties should be given for any factual inconsistencies or hallucinated elements.

Rating Scale:

5 (Completely Consistent) - The generated problem is perfectly consistent, with no factual errors or hallucinated elements.
4 (Mostly Consistent) - The generated problem is largely consistent, with only minor factual discrepancies.
3 (Somewhat Consistent) - The generated problem contains a mix of accurate and inaccurate information, with noticeable factual errors.
2 (Mostly Inconsistent) - The generated problem has several factual errors but retains some correct elements from the original problem.
1 (Completely Inconsistent) - The generated problem contains multiple factual errors or is completely unrelated to the original problem.
0 (Terrible) - The generated problem has no factual consistency and is entirely irrelevant to the original problem.

Evaluation Steps:

1. Carefully read and analyze the original problem, identifying the key facts and details it presents.
2. Review the generated problem, comparing it to the original, while ignoring any multiple-choice answer options. Check for factual errors, hallucinated details, or inconsistencies not supported by the original problem.
3. Assign a score for consistency based on the rating scale above.

Original Problem:

{{Original Problem}}

Generated Problem:

{{Generated Problem}}

Evaluation Output Form (scores ONLY):

{{Consistency}}

Figure 14: Prompt for Evaluation of Consistency

You will be given two math problems: an original problem and a problem generated from an image. Your task is to rate how relevant the generated problem is to the original problem, excluding any multiple-choice answer options. Focus solely on the problem statement itself.

Please make sure you read and understand these instructions carefully. Keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Relevance (0-5) - The selection of important content from the original problem, ignoring the multiple-choice options. The generated problem should include only the key elements and important information from the original problem. Penalties should be given for any irrelevant or redundant information included.

Rating Scale:

5 (Completely Relevant) - The generated problem perfectly captures the key elements of the original problem without including any irrelevant or redundant information.
4 (Mostly Relevant) - The generated problem largely captures the important points, with only minor irrelevant or redundant details.
3 (Somewhat Relevant) - The generated problem includes a mix of important and irrelevant elements, with noticeable redundancy or missing key points.
2 (Mostly Irrelevant) - The generated problem contains mostly irrelevant information, failing to capture many of the key elements of the original problem.
1 (Completely Irrelevant) - The generated problem contains almost no relevant information and is largely off-topic.
0 (Terrible) - The generated problem is entirely unrelated to the original problem.

Evaluation Steps:
1. Carefully read and analyze the original problem, identifying its main points and key elements.
2. Review the generated problem, comparing it to the original, while ignoring any multiple-choice answer options. Assess how well it covers the key points of the original problem and whether it contains irrelevant or redundant information.
3. Assign a score for relevance based on the rating scale above.

Original Problem:

{{Original Problem}}

Generated Problem:

{{Generated Problem}}

Evaluation Output Form (scores ONLY):

{{Relevance}}

Figure 15: Prompt for Evaluation of Relevance

You will be given two math problems: an original problem and a problem generated from an image. Your task is to rate how similar the generated problem is to the original problem, excluding any multiple-choice answer options. Focus solely on the problem statement itself.

Please make sure you read and understand these instructions carefully. Keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:

Similarity (0-5) - The degree to which the generated problem resembles the original problem in terms of both structure and content, ignoring the multiple-choice options. The generated problem should closely match the original problem in terms of key concepts, elements, and overall presentation.

Rating Scale:

5 (Completely Similar) - The generated problem closely matches the original problem in both structure and content, with no significant differences.
4 (Mostly Similar) - The generated problem is largely similar, with only minor differences in structure or content.
3 (Somewhat Similar) - The generated problem contains a mix of similarities and differences, with noticeable variations in structure or content.
2 (Mostly Dissimilar) - The generated problem only retains a few similarities and has significant structural or content differences.
1 (Completely Dissimilar) - The generated problem bears little resemblance to the original, with major differences in both structure and content.
0 (Unrelated) - The generated problem is entirely different from the original.

Evaluation Steps:
1. Carefully read and analyze the original problem, identifying its key concepts, elements, and structure.
2. Review the generated problem, comparing it to the original, while ignoring any multiple-choice answer options. Evaluate how closely the generated problem matches the original problem in terms of key concepts, elements, and structure.
3. Assign a score for similarity based on the rating scale above.

Original Problem:

{{Original Problem}}

 Generated Problem:

{{Generated Problem}}

Evaluation Output Form (scores ONLY):

{{Similarity}}

Figure 16: Prompt for Evaluation of Similarity
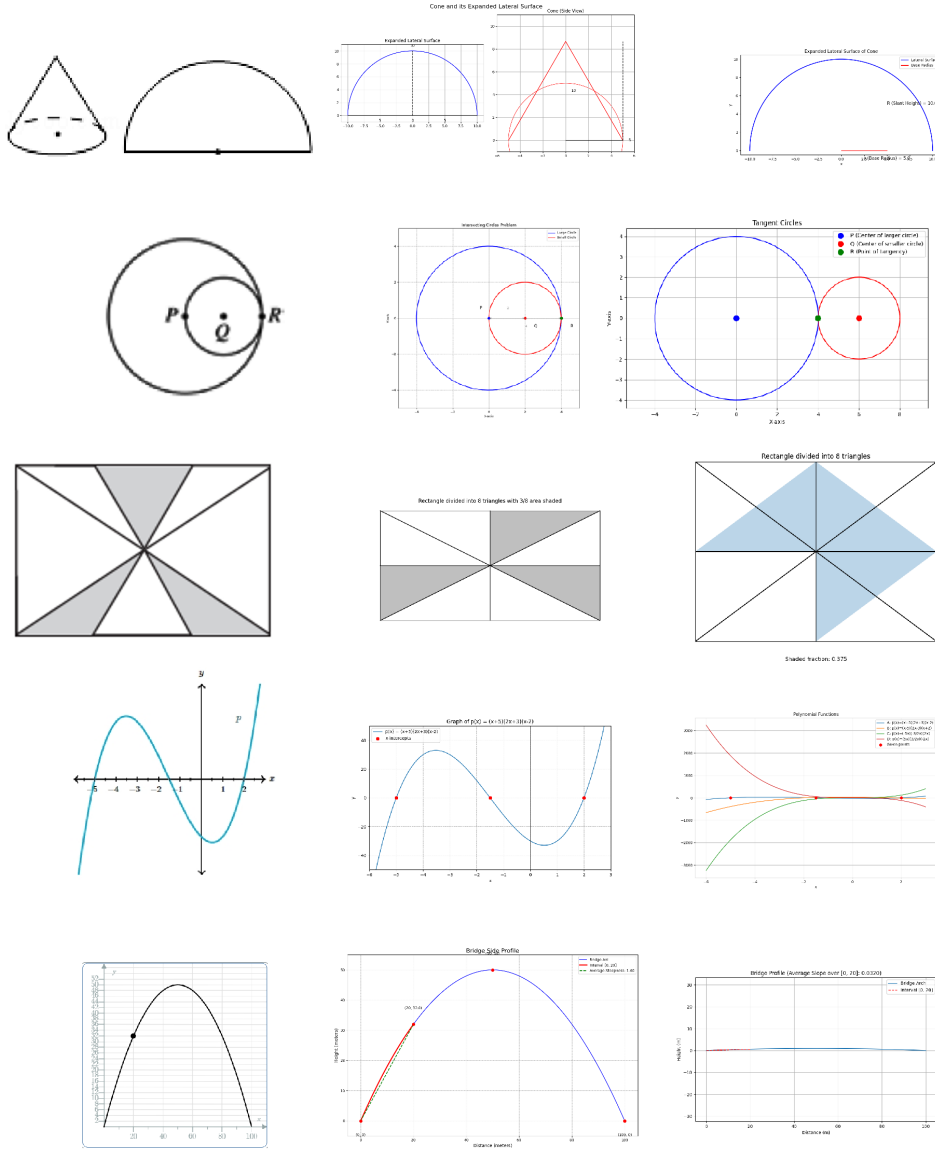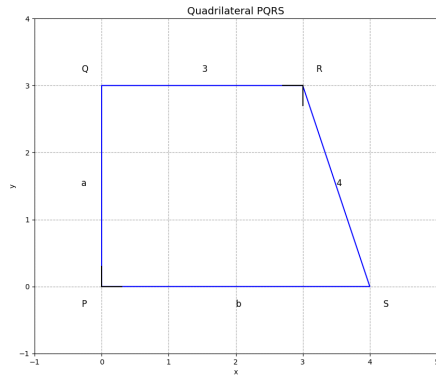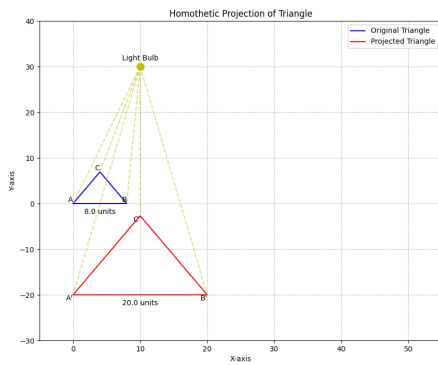
A.4. Generated samples

Figure 17: Comparison of generated samples by VISTA(center), baseline method(right), and figures from original problems (left)

Given the quadrilateral PQRS,
where $PQ = a$, $QR = 3$, $RS = 4$, and $PS = b$,
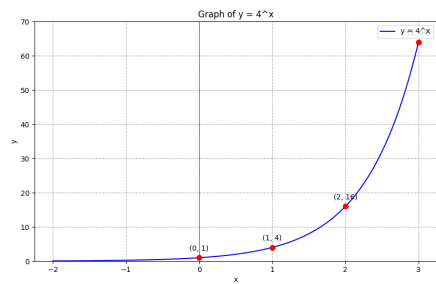find the perimeter of the quadrilateral if $a = 5$ and $b = 6$.
A) 15
B) 16
C) 17
D) 18

A light bulb is placed at a height of 15 units above the base of
an original triangle $\triangle ABC$ with vertices
$A(0,0)$, $B(8,0)$, and $C(4,6)$.
The light bulb projects a shadow of the triangle onto a plane
25 units above the base, forming a projection
triangle $\triangle A'B'C'$.

Given that the base of the projection triangle $A'B'$ is 20 units,
what is the height of the projection triangle $\triangle A'B'C'$?

A) 15 units
B) 30 units
C) 40 units
D) 50 units

Consider the graph of the function $y = 4^x$ shown in the image.

1. Determine the x-coordinate where the function $y = 4^x$
intersects the line $y = 1$.
2. Calculate the value of y when $x = 2$.
3. Find the x-coordinate where the function $y = 4^x$ equals 10.
4. Describe the behavior of the function $y = 4^x$ as
x approaches negative infinity.
5. Determine the value of x for which $y = 4^x$ equals
$\frac{1}{4}$.

Figure 18: Math problem generation samples using VISTA

A.5. Resources

To produce a image with our multi-agent framework, we adopted claude-sonnet-3.5-20240620 and spent approximately 3400 input tokens (2000 to 5000 tokens) and 600 output tokens (200 to 900 tokens) in a minute on average. We also adopted GPT4o for evaluation, which costed 680 input tokens 41 output tokens and 40 seconds on average, per image. Actual costs are vary, depending on complexity of the source.