MagicGeo: Training-Free Text-Guided Geometric Diagram Generation

Anonymous ACL submission

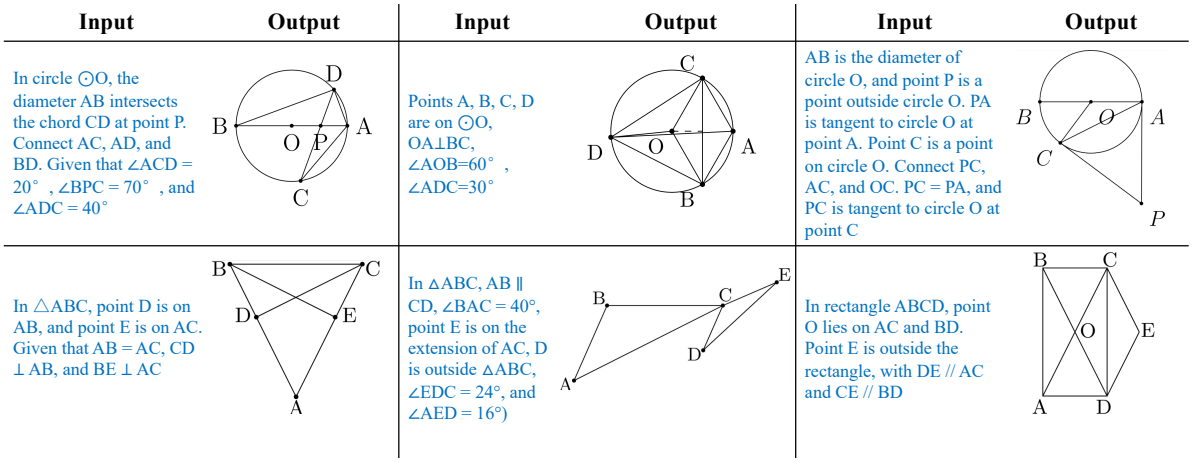


Figure 1: *MagicGeo* has the capability to generate accurate complex geometric diagrams from natural language.

Abstract

Geometric diagrams are critical in conveying mathematical and scientific concepts, yet traditional diagram generation methods are often manual and resource-intensive. While text-to-image generation has made strides in photorealistic imagery, creating accurate geometric diagrams remains a challenge due to the need for precise spatial relationships and the scarcity of geometry-specific datasets. This paper presents MagicGeo, a training-free framework for generating geometric diagrams from textual descriptions. MagicGeo formulates the diagram generation process as a coordinate optimization problem, ensuring geometric correctness through a formal language solver, and then employs coordinate-aware generation. The framework leverages the strong language translation capability of large language models, while formal mathematical solving ensures geometric correctness. We further introduce MagicGeoBench, a benchmark dataset of 220 geometric diagram descriptions, and demonstrate that MagicGeo outperforms current methods in both qualitative and quantitative evaluations. This work provides a scalable, accurate solution for automated diagram generation, with significant implications for educational and academic applications.

1 Introduction

"A picture is worth a thousand words" is a widely recognized proverb in literature. Specifically, diagram, as a form of picture, is essential in conveying information and have long been utilized across fields such as science and engineering. Extensive research (Larkin and Simon, 1987; Stenning and Oberlander, 1995) demonstrates that diagrams often outperform text in solving determinate problems. Prominent figures like Einstein and Hadamard have famously asserted that they do not "think in words" (Larkin and Simon, 1987). Furthermore, Stenning and Oberlander (1995) argues that text permits expression of ambiguity in the way that diagrams cannot easily accommodate. This paper focuses on the task of converting descriptions into structured diagrams, with particular emphasis on geometric diagrams, which play a critical role in mathematics and science. This task serves as a foundational step toward advancing diagram generation for scientific textbooks.

Traditional geometric diagram construction is closely associated with a suite of graphic drawing tools, such as Cinderella (Yu et al., 2015), Geometry Expert (Chou et al., 1996), Z+Z Su-

per Sketchpad (ZHANG et al., 2007), and WinGCLC (Janičić and Trajković, 2003; Szirmay-Kalos, 2003). These tools offer interactive platforms for drawing geometric figures. However, they are burdened by the need for manual input, which is both time-consuming and resource-intensive. This paper presents the development of an automatic, text-guided geometric diagram generation system, eliminating the manual effort typically involved. Such a system holds significant potential for streamlining diagram creation, offering considerable utility in the preparation of educational resources.

Recent advancements in text-to-image generation have achieved notable progress in synthesizing photorealistic images (Cao et al., 2024; Zhou and Shimada, 2023). However, these methods, trained on large datasets of natural image-text pairs, often struggle with diagram generation. Efforts to address this challenge include DiagrammerGPT (Zala et al., 2023), which proposes a two-stage framework using layout as an intermediary to enable spatial control, and AutomaTikZ (Belouadi et al., 2023), which leverages the TikZ graphic language to autonomously generate scientific figures from captions. Despite these advances, both approaches rely on supervised training data, which limits their generalizability. Furthermore, the scarcity of geometry-specific image-text pairs relative to general image-text corpora makes it difficult to learn the semantic and structural logic of geometric layouts directly from natural language inputs.

In this paper, we introduce MagicGeo, a framework for the automatic generation of text-to-geometric diagrams in a training-free manner, thereby sidestepping the need for paired geometry-text datasets. We focus on geometric diagram as it stands out due to its stringent precision requirement, that is, properties such as parallelism, orthogonality, and degree constraint must be rigorously maintained. Given that even minor inaccuracies are immediately noticeable, this task poses significant challenges within image generation.

Our key insight is that correctness hinges on the precise placement of points. Once the point locations are accurate, constructing the geometry becomes straightforward, such as connecting points with lines or drawing circles. Drawing inspiration from computational geometry methods used in geometry theorem provers (Wu, 2008), we model diagram generation as a set of polynomial equations based on point coordinates.

While large language models (LLMs) exhibit impressive capabilities in language understanding and reasoning, they are not inherently equipped to solve complex multi-constraint tasks (Kambhampati et al., 2024). As a result, directly using LLMs to solve for point coordinates leads to errors and hallucinations. Instead, we turn to leverage LLM's strengths in translation to convert geometry texts into key formal information. This information is then used to formulate an optimization problem, which is solved algorithmically to ensure that the geometric constraints are satisfied.

To this end, MagicGeo operates in three distinct stages: 1) Autoformalization with LLM: LLMs interpret the geometry description and translate it into an optimization problem, defining a set of constraints with respect to the point coordinates. 2) Solving with Verification: Computational geometry principles are applied to search for one solution that satisfies all constraints; if no solution is found, the system reverts to the autoformalization step to re-extract the necessary information. 3) Coordinate-aware generation: We employ point coordinates to generate TikZ language, which serves as an intermediary representation for the creation of the corresponding geometric diagram.

To advance the evaluation of text-to-geometric diagram generation and promote further research, we present MagicGeoBench, a real-world dataset containing 220 plane geometry descriptions sourced from middle school math exams. Empirical results demonstrate that MagicGeo significantly outperforms state-of-the-art baselines, both qualitatively and quantitatively. Figure 1 illustrates several examples. We also explore its potential for diagram editing, showcasing how the diagrams can be tailored to user preferences, thereby enhancing practical utility. While our experiments focus on plane geometry, the underlying methodology is highly extensible to other geometric branches, such as analytical and solid geometry. Our current goal is to demonstrate the efficacy of the propose concept, which we believe will foster broader exploration and inspire further innovation in the field.

In summary, our key contributions are:

- We propose a novel perspective that frames geometric diagram generation as a well-defined optimization problem, enhancing its tractability within the zero-shot capabilities of LLMs.
- We present MagicGeo, a training-free framework for high-quality geometric diagram generation. Integrating LLMs with formal solvers

for diagram generation, MagicGeo achieves both generalizability and correctness.

- We introduce a test benchmark to foster research in this area. Empirically, MagicGeo delivers highly accurate geometric diagrams, surpassing the performance of the baseline models, without requiring training data.

2 Related Work

Text-to-Image Generation. Text-to-image generation (Zhang et al., 2023; Bie et al., 2024; Jia et al., 2024) has become a rapidly growing field in computer vision and machine learning. This progress traced back to the emergence of Generative Adversarial Networks (GANs) (Goodfellow et al., 2020), which paved the way for research focused on generating images from textual prompts (Reed et al., 2016; Tao et al., 2022; Xu et al., 2018; Zhang et al., 2021, 2017, 2018). Transformer-based autoregressive models (Ding et al., 2021; Gafni et al., 2022; Ramesh et al., 2021; Yu et al., 2022) have attracted significant attention due to their strong capabilities in modeling text-image alignment, as demonstrated by typical models such as DALL-E (Ramesh et al., 2021) and STAR (Ma et al., 2024). In parallel, diffusion models (Gu et al., 2022; Nichol et al., 2021; Ramesh et al., 2022; Rombach et al., 2022; Saharia et al., 2022) have emerged as a prominent type of generative model for image generation, achieved through the gradual introduction of noise in iterative steps. Notable examples include Imagen (Saharia et al., 2022) and others focus on improving compositionality, e.g., attribute binding (Chefer et al., 2023; Feng et al., 2022).

Although these approaches have advanced the generation of realistic scene imagery and propelled text-to-image generation into the spotlight of machine learning research, they struggle with tasks that demand precise control over complex structures and intricate relationships. This includes the generation of diagrams in fields like geometry, architecture, or other technical domains.

Text-to-Diagram Generation. Generating diagrams from text has long been an intriguing area of research and has recently garnered considerable attention, driven by the success of text-to-image generation. Early efforts (Ghosh et al., 2018; Shahbaz et al., 2011; Btoush and Hammad, 2015) primarily focused on generating entity-relationship diagrams, utilizing semantic heuristics to identify entities, attributes, and relationships from natural language specifications. With the rise of LLMs in various language generation tasks (Touvron et al., 2023a,b; OpenAI et al., 2024; Chung et al., 2024; Mann et al., 2020; Chowdhery et al., 2023), recent work has also leveraged LLMs to facilitate spatial control in diagram generation. These methods can be generally classified into two categories: layout-guided models and code-guided methods. Layout-guided approaches, exemplified by DiagrammerGPT (Zala et al., 2023), employ a two-stage framework that first leverages LLMs to plan layout, then applies layout-guided diffusion models. Code-guided methods, such as AutomaTikZ (Belouadi et al., 2023), fine-tune LLMs on large TikZ datasets to generate code for scientific vector graphics, while DiagramAgent (Wei et al., 2024) introduces a four-agent framework leveraging code for text-to-diagram generation and editing.

In geometric diagram generation, both existing approaches face significant limitations. First, image generators suffer from limited spatial fidelity (Gokhale et al., 2022; Chatterjee et al., 2024a,b), despite extensive research in the layout-to-image field (Li et al., 2023; Yang et al., 2023; Balaji et al., 2022; Singh et al., 2023; Couairon et al., 2023; Xie et al., 2023). This limitation prevents these methods from fulfilling precise geometric constraints. Second, code-guided models for diagram generation are restricted by the capabilities of text-to-code models (Roziere et al., 2023; Fried et al., 2022; Li et al., 2022; Hui et al., 2024; Guo et al., 2024), which rely on large, data-intensive datasets for effective performance.

In contrast, we propose a training-free method that avoids the need for supervised data, leveraging precise point coordinates to enforce stringent geometric constraints. Our approach shares similarities with Zhengyu and Xiuqin (2023), which also utilizes point coordinates, but diverges in three key aspects. 1) We leverage the zero-shot capabilities of LLMs to extract points and constraints, bypassing the labor-intensive process of building entity relationship extractors. 2) We introduce a self-verification module to correct LLM-extracted information when the optimization problem is unsolvable. 3) We leverage text-to-code LLMs for TikZ code generation, enabling richer textual insights such as point connections, a capability not fully explored in Zhengyu and Xiuqin (2023). Finally, empirical results demonstrate our system's ability of generating complex geometric diagrams.

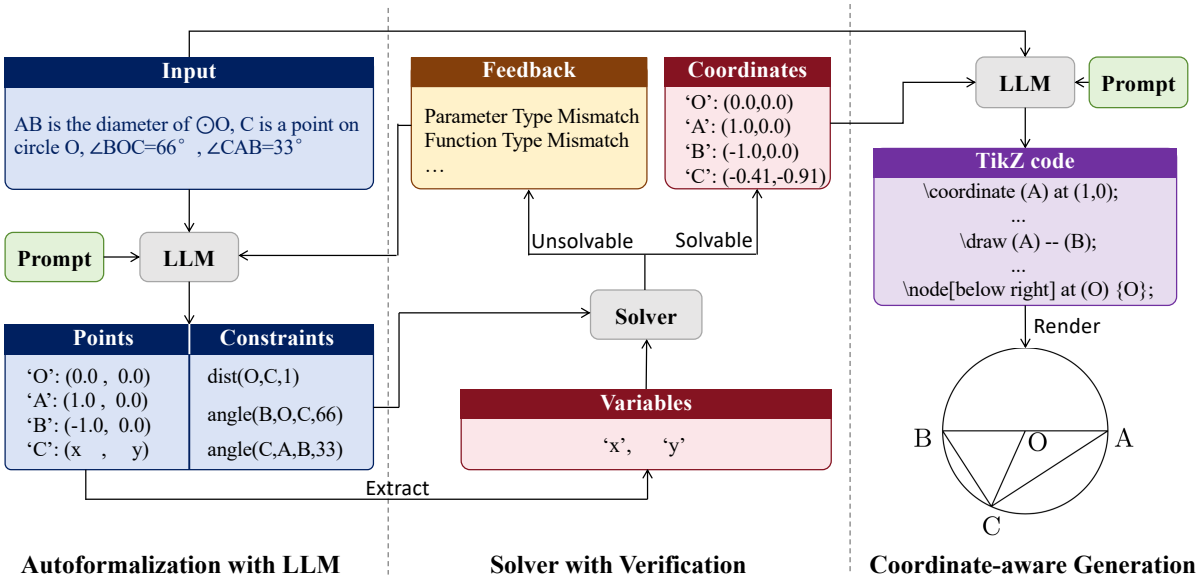


Figure 2: The overall framework of MagicGeo consists of three stages: Autoformalization with LLM, Solver with Verification, and Coordinate-aware Generation.

3 Method

In the task of text-to-geometric diagram generation, given a textual description T, the objective is to generate a corresponding geometric diagram D that adheres to the geometric constraints outlined in T. To realize this objective, we introduce MagicGeo, as depicted in Figure 2.

3.1 Autoformalization with LLM

We observe that a geometric diagram can be efficiently represented by the coordinates of points and the relationships between them. To formalize this process, we propose a specialized formal language that encapsulates the geometric structure through a set of points and associated constraints, defining their interrelationships. The objective of autoformalization is to convert natural language input, often ambiguous or imprecise, into a precise, unambiguous formal representation that accurately captures geometric relationships and configurations.

Building on the success that LLMs can translate between formal and informal mathematical statements to some extent (Wu et al., 2022), we investigate their potential to convert natural language mathematics into our customized formal language, suitable for the solver we introduce. By providing these models with a predefined prompt, we guide their generation, ensuring the output aligns with the requirements of the subsequent solver.

Specifically, we prompt the LLM to generate two key pieces of information: coordinates $Points$ represented by variables $Vars$ and the required geometric constraints $Cons$ based on these coordinates $Points$. Figure 2 shown an example of autoformalization with LLM. The prompt consists of a structured database containing a wide range of geometric constraints, along with corresponding instructions that elucidate their precise meanings. By leveraging this structured representation, the LLM interprets the prompt as a comprehensive reference manual, and processes user input in accordance with the specifications outlined in the manual, systematically translating the given descriptions into customized formal languages.

Surprisingly, we find that LLMs exhibit a decent proficiency in formalizing mathematical concepts in our scenario. Notably, the LLM demonstrates the ability to employ intricate reasoning to adapt and generalize beyond explicitly stated rules. This capability allows the model to infer implicit relationships and make logical extensions where necessary. For instance, if the input contains the phrase "triangle ABC is inscribed in circle O", the LLM recognizes that this implies the distances from O to points A, B, and C are equal to the radius of the circle. This inference is made despite the absence of explicit instructions in the manual, highlighting the model's capacity to apply intuitive geometric principles autonomously.

Furthermore, in our approach, we utilize the second phase, namely the solver, to rigorously verify the accuracy of the generated translation. In instances where the candidate autoformalization

fails to produce a valid solution, we incorporate the feedback derived from this failure into the process. Specifically, this feedback is treated as a new contextual input, which is then fed into the subsequent iterations of the generation process. This iterative refinement mechanism enables continuous improvement of the formalization output. Our results demonstrate that by including such a verification step within the framework, the autoformalization accuracy of LLMs is significantly enhanced.

3.2 Solver with Verification

Solver. We recognize the existence of numerous interactive theorem provers, such as Isabelle (Wenzel et al., 2008), Coq (Huet et al., 1997), HOL Light (Harrison, 1996; Srivas and Camilleri, 1996), and Lean (De Moura et al., 2015; Felty and Middeldorp, 2015). These systems function as specialized programming languages, allowing users to formalize statements and construct proofs, which are then automatically verified for correctness. However, these tools are inherently tailored for mathematical proof problems and thus ill-suited for numerical computation tasks. Additionally, when the solver fails, debugging is challenging due to its lack of interpretability, making it ineffective in guiding the conversational autoformalization process.

To address this, we develop a custom solver, utilizing the constraints of the formal language as function names and leveraging computational analytical geometry methods to examine the constraints and solve the coordinates. Specifically, in order to determine point coordinates, we first identify the relevant variables and extract them into a structured list. We then implement an iterative approach to traverse each variable, simultaneously validating geometric constraints through the derived function names. A precise solution for the coordinates is obtained once a value set is identified for the variables that satisfies all the constraints.

Verification. Verification plays a crucial role in bridging the Solver and Autoformalization processes, enabling the provision of immediate and actionable feedback for newly generated formalizations. By offering insights into the nature of errors, verification empowers LLMs to refine their understanding and improve the quality of subsequent formalization outputs. Our experimental analysis highlights two primary failure modes that often require the autoformalization phase to be restarted: (1) detection of non-compliant characters, where symbols or elements violate established syntax or formal language rules, and (2) errors in parameter specifications, including incorrect value assignments or misalignment of parameter numbers.

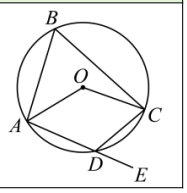
Origin text	Origin diagram
As shown in the figure, quadrilateral ABCD is an inscribed quadrilateral of circle ⊙O, and E is a point on the extension of AD. Given that ∠AOC = 128°, what is the value of ∠CDE?	
Modified text	
The quadrilateral ABCD is inscribed in circle O, with E being a point on the extension of AD. ∠AOC = 128°, and ∠CDE = 64°	

Figure 3: Illustrating an example of modifying the original text to include necessary information during MagicGeoBench construction.

3.3 Coordinate-aware Generation

While directly inputting precise coordinates and textual descriptions into generative models may seem intuitive, it often leads to disorganized visual elements (e.g., misaligned points and lines) that fail to faithfully represent the intended structure. To overcome this limitation, we introduce a more disciplined approach, employing TikZ as an intermediate representation, similar to AutomaTikZ (Belouadi et al., 2023). However differently, we capitalize on precise point coordinates to harness the zero-shot code generation capabilities of LLMs, eliminating the need for finetuning. This enables the generation of figures that not only maintain structural clarity but also exhibit high fidelity to the original textual descriptions.

4 Experiments

4.1 MagicGeoBench

To rigorously evaluate the performance of text-to-geometric diagram models, we introduce the MagicGeoBench Dataset, a meticulously curated collection of 220 plane geometry questions drawn from high school entrance examinations. In constructing this dataset, we retain the original text for self-contained questions. For questions where essential information is embedded in diagrams rather than explicitly stated in text, we augment the textual descriptions so that diagram can be generated solely from textual input. Figure 3 illustrates such an example. The evaluation dataset covers fundamental geometric shapes, and is systematically categorized into three groups: 70 questions on circles, 70 on triangles, and 80 on quadrangles.

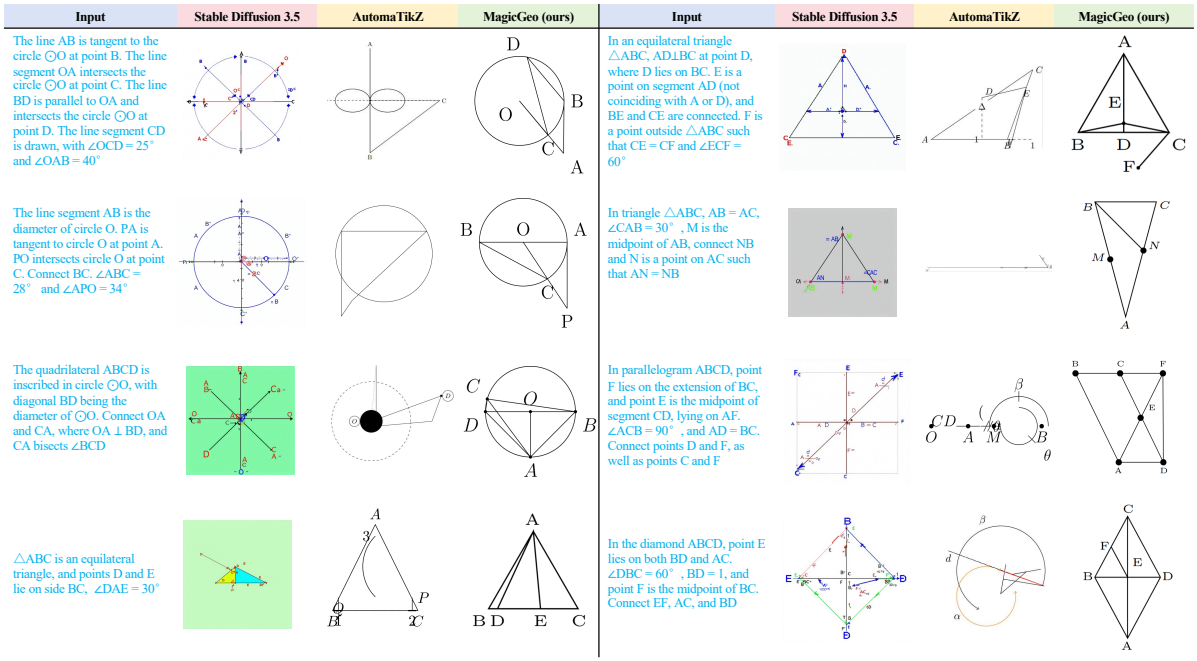


Figure 4: Qualitative comparison with other approaches. Our method generates results that rigorously adhere to geometric constraints while maintaining high perceptual quality.

Method	Img-Txt				Img-Img			
	Circle	Triangle	Quadrangle	Avg	Circle	Triangle	Quadrangle	Avg
SD3.5-Large (Sauer et al., 2024)	33.25	**34.72**	**34.02**	**33.99**	80.85	83.65	82.01	82.17
AutomaTikZ (Belouadi et al., 2023)	30.89	29.95	29.97	30.27	84.66	83.82	87.61	85.36
MagicGeo (ours)	**33.93**	31.89	32.13	32.65	**91.49**	**88.16**	**89.90**	**89.85**

Table 1: Quantitative comparison in terms of CLIP score. Higher values indicate better performance (↑).

4.2 Settings

Baselines. There is a lack of extensive research focusing on the automatic generation of geometric diagrams. Hence we compare our proposed approach with two established baselines. The first baseline, Stable Diffusion 3.5 (SD3.5) (Sauer et al., 2024), is a robust model known for its prowess in generating photorealistic images across a variety of domains. Its ability to synthesize high-quality, realistic images positions it as a strong competitor in the image generation space. The second baseline, AutomaTikZ (Belouadi et al., 2023), utilizes the TikZ language as an intermediate step for creating high-quality graphical representations, which is a relevant benchmark for our work in the domain of geometric diagram generation.

Evaluation Metrics. Our goal is to ensure that the diagrams both adhere to textual instructions and conform to typical visual characteristics of geometric illustrations, distinguishing them from photorealistic images. To evaluate these two aspects, we utilize CLIP (Radford et al., 2021) to calculate cosine similarity. Given that CLIP is designed for general images, we also conduct a user study to validate the effectiveness of our model.

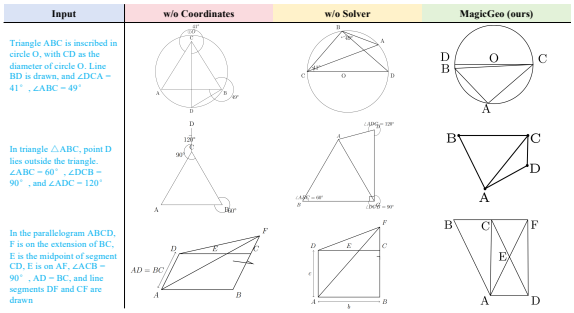


Figure 5: Illustrating that the solver effectively ensures precise alignment with the accompanying text.

4.3 Results

Quantitative Evaluation. Table 1 presents a comparison between our approach and competitive baselines in terms of CLIP score. When compared to the dedicated image model SD3.5 (Sauer

Method	Textual alignment				Image quality			
	Circle	Triangle	Quadrangle	Avg	Circle	Triangle	Quadrangle	Avg
SD3.5-Large (Sauer et al., 2024)	2.50	2.20	2.72	2.47	2.42	2.11	2.11	2.21
AutomaTikZ (Belouadi et al., 2023)	2.28	2.08	2.12	2.16	2.28	2.08	2.15	2.17
MagicGeo (ours)	**1.22**	**1.12**	**1.06**	**1.13**	**1.30**	**1.20**	**1.20**	**1.23**

Table 2: Average user ranking score of textual alignment and image quality. 1 is the best, 3 is the worst. It is evident that users prefer our results more than others given the superior quality of ours.

	Circle	Triangle	Quadrangle	Avg
w/o Verification	92.0	95.7	96.3	94.7
w Verification	**97.3**	**100**	**98.8**	**98.7**

Table 3: Illustrating the pivotal role of the verification mechanism in enhancing the autoformalization process.

et al., 2024), our model exhibits a higher similarity to the reference image, as suggested by the image-to-image score. However, we notice that in terms of image-text alignment, our model achieves a lower CLIP score. This discrepancy is likely due to the CLIP model's broad training on general image-text pairs, which may bias towards general image generation models. More importantly, our model surpasses the text-to-diagram baseline AutomaTikZ (Belouadi et al., 2023) in both image-text alignment and image quality, demonstrating the effectiveness of our approach.

Qualitative Evaluation. In addition to quantitative metrics, qualitative evaluation plays a crucial role in assessing generation task. We provide the qualitative visual comparison of these methods in Figure 4. The baseline method SD3.5 (Sauer et al., 2024) generally succeeds in generating simple geometric shapes such as circles and triangles, but it struggles to accurately produce more complex geometric configurations, such as a triangle inscribed within a circle. On the other hand, AutomaTikZ (Belouadi et al., 2023) is capable of generating visually appealing diagrams owing to its use of the TikZ language. However, both methods fail to consistently adhere to underlying geometric constraints, resulting in diagrams that exhibit noticeable inconsistencies upon inspection. In contrast, our proposed method rigorously adheres to geometric constraints while simultaneously maintaining a high level of perceptual quality.

User Study. In order to obtain the user's subjective evaluation of the generated image, we conduct a user study involving 20 participants. In the study, we use 60 samples with 20 in each category. Each sample is consisted of a text input paired with three corresponding output images. Participants were instructed to independently rank each image (1 is the best, 3 is the worst) on two distinct aspects: (i) image quality and (ii) adherence to the textual description. We report the average ranking score in Table 2. It is evident that users prefer our results more than others given the superior quality of ours.

4.4 Ablations

The Effect of Verification. In our framework, when the solver fails to find a solution, feedback is provided to the LLM for re-autoformalization, a process we refer to as verification. This process allows for a maximum of five feedback iterations, aiming to iteratively correct errors identified by the solver. Here we compare its performance against a baseline system that excludes verification. The evaluation criterion focuses on the accuracy of output points and constraints, which are manually verified. As shown in Table 3, the incorporation of verification results in a substantial improvement in autoformalization accuracy, increasing from 94.7% to 98.7%. This highlights the pivotal role of the solver's feedback mechanism in enhancing the autoformalization process.

The Effect of Solver. We propose the use of analytical geometry methods to develop a custom solver designed for precise point location determination, which is subsequently leveraged for diagram generation. To evaluate the effectiveness of our solver, we compare our approach against two alternative methods: (1) w/o Coordinates: this approach utilize LLMs to directly generate TikZ code without incorporating explicit point coordinates, akin to the approach used in AutomaTikZ; (2) w/o Solver: this variant first ask the LLM to infer the point coordinates and then use these coordinates for coordinate-aware TikZ generation. We compare their results in terms of the CLIP score in Table 4. To provide a clearer understanding, we present several visual examples in Figure 5. The incorporation of explicit coordinates significantly enhances the quality of diagram generation. However, some issues persist, such as the failure to satisfy the constraint "angle ABC equals 49 degrees" in the first example and the constraint "angle ADC equals 120 degrees" in

Method	Img-Txt				Img-Img			
	Circle	Triangle	Quadrangle	Avg	Circle	Triangle	Quadrangle	Avg
w/o Coordinates	31.25	31.31	30.44	31.00	86.31	88.46	88.90	87.89
w/o Solver	32.25	31.52	30.77	31.51	87.37	**88.51**	**90.84**	88.91
MagicGeo (ours)	**33.93**	**31.89**	**32.13**	**32.65**	**91.49**	88.16	89.90	**89.85**

Table 4: The effect of solver in terms of CLIP score. Using LLM to directly generate TikZ code is denoted as w/o Coordinates. Asking LLM to infer coordinates followed by coordinate-aware generation is denoted as w/o Solver.

Stage	LLM	Circle	Triangle	Quad	Avg
1	DeepSeek-V3	0.97	1.00	0.99	0.99
	Qwen-plus	0.96	0.99	0.99	0.98
	GPT-4o mini	0.97	0.99	0.99	0.98
3	DeepSeek-V3	1.00	1.00	1.00	1.00
	Qwen-plus	1.00	1.00	0.97	0.99
	GPT-4o mini	0.99	1.00	1.00	1.00

Table 5: Illustrating that our framework is robust to different LLMs, which shows negligible impact.

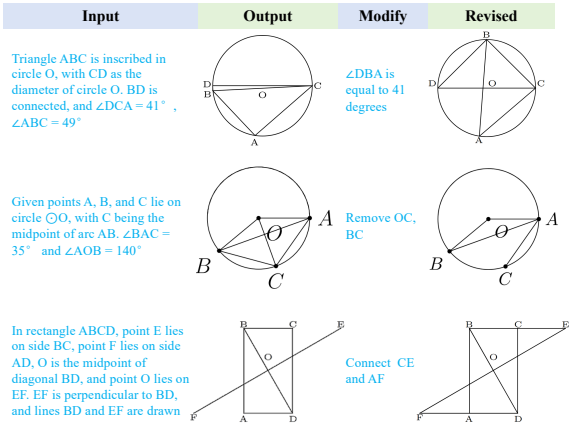


Figure 6: Application to diagram editing.

the second example. In contrast, the application of solver effectively addresses all constraints, ensuring precise alignment with the accompanying text and thus superior quality.

The Effect of Using Different LLMs. We employ the DeepSeek-V3 (Liu et al., 2024) model for both Stage 1 and Stage 3 in our framework. To study the impact of different LLMs, we investigate two models: Qwen-plus (Yang et al., 2024) and GPT-4o mini (Shahriar et al., 2024) . We isolate the LLM variation to a single stage—either Stage 1 or Stage 3—while maintaining the other stage constant. For evaluation, we manually examine autoformalization accuracy in Stage 1 and visually inspect the generated diagrams in Stage 3. A sample of 60 instances was experimented, with the accuracy presented in Table 5. It is important to note that we utilize distinct prompts for different LLMs to fully harness their respective capabilities. Our findings indicate that the choice of LLM has a negligible impact on the final outcomes, demonstrating the suitability of LLMs for these tasks. This suggests that our framework maintains consistent performance regardless of the specific LLM.

4.5 Application to Diagram Editing

Our method leverages precise coordinate information and thus is able to make effective diagram modifications based on user intent. We present examples of diagram editing results in Figure 6. Simple tasks, such as adding or deleting lines, are accomplished by re-executing the third stage. More complex adjustments(e.g., specify a new angle degree in the first example), are handled effectively by our framework, which quickly determines the necessary coordinates for adjustment. This demonstrates the potential of our framework in real-world diagram editing applications.

5 Conclusion

In conclusion, this paper presents MagicGeo, a novel framework for the automatic generation of geometric diagrams from textual descriptions, which stands out for its training-free approach and high precision. By reframing the diagram generation task as an optimization problem, MagicGeo ensures the accuracy of key geometric properties—such as parallelism and orthogonality—by leveraging analytical geometry rules. The comprehensive evaluation of MagicGeo, including empirical comparisons with state-of-the-art baselines and ablation studies, demonstrates its effectiveness in producing accurate and reliable diagrams. Ultimately, MagicGeo offers significant potential for streamlining the creation of educational and academic diagrams, with broader implications for enhancing content generation in scientific and educational settings.

6 Limitations

While MagicGeo demonstrates notable advancements in the automatic generation of geometric diagrams from textual descriptions, several limitations must be acknowledged.

One limitation of our framework is its reliance on LLMs to translate complex geometric descriptions into formal representations that adhere to geometric conventions and generate accurate TikZ code. Although current translation performance, as shown in the ablation, is highly effective, it is not yet flawless, with visual examples presented in the Appendix section. We anticipate that ongoing advancements in LLM research, particularly in mathematical reasoning and code generation, will mitigate this limitation.

The current solver exhibits extended processing times for complex diagrams, with efficiency influenced by factors such as input complexity, the number of geometric entities, and the precision required for diagram generation. Preliminary experiments indicate that generation times typically range in the order of seconds; while for very intricate complex geometric diagram, processing times can exceed one hour. Future work will focus on enhancing solver efficiency through parallelization, optimized constraint-solving methods, and the development of heuristic techniques that balance computational cost and diagram accuracy.

References

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. 2022. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*.

Jonas Belouadi, Anne Lauscher, and Steffen Eger. 2023. Automatikz: Text-guided synthesis of scientific vector graphics with tikz. *arXiv preprint arXiv:2310.00367*.

Fengxiang Bie, Yibo Yang, Zhongzhu Zhou, Adam Ghanem, Minjia Zhang, Zhewei Yao, Xiaoxia Wu, Connor Holmes, Pareesa Golnari, David A Clifton, et al. 2024. Renaissance: A survey into ai text-to-image generation in the era of large model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Eman S Btoush and Mustafa M Hammad. 2015. Generating er diagrams from requirement specifications based on natural language processing. *International Journal of Database Theory and Application*, 8(2):61–70.

Hanqun Cao, Cheng Tan, Zhangyang Gao, Yilun Xu, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. 2024. A survey on generative diffusion models. *IEEE Transactions on Knowledge and Data Engineering*.

Agneet Chatterjee, Yiran Luo, Tejas Gokhale, Yezhou Yang, and Chitta Baral. 2024a. Revision: Rendering tools enable spatial fidelity in vision-language models. In *European Conference on Computer Vision*, pages 339–357. Springer.

Agneet Chatterjee, Gabriela Ben Melech Stan, Estelle Aflalo, Sayak Paul, Dhruba Ghosh, Tejas Gokhale, Ludwig Schmidt, Hannaneh Hajishirzi, Vasudev Lal, Chitta Baral, et al. 2024b. Getting it right: Improving spatial consistency in text-to-image models. In *European Conference on Computer Vision*, pages 204–222. Springer.

Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. 2023. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):1–10.

Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. 1996. An introduction to geometry expert. In *Automated Deduction—Cade-13: 13th International Conference on Automated Deduction New Brunswick, NJ, USA, July 30–August 3, 1996 Proceedings 13*, pages 235–239. Springer.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Guillaume Couairon, Marlene Careil, Matthieu Cord, Stéphane Lathuiliere, and Jakob Verbeek. 2023. Zero-shot spatial layout conditioning for text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2174–2183.

Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. 2015. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pages 378–388. Springer.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. 2021. Cogview: Mastering text-to-image generation via transformers. *Advances in neural information processing systems*, 34:19822–19835.

Amy P Felty and Aart Middeldorp. 2015. Automated deduction–cade-25.

Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2022. Training-free structured diffusion guidance for compositional text-to-image synthesis. *arXiv preprint arXiv:2212.05032*.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. 2022. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*.

Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. 2022. Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision*, pages 89–106. Springer.

Sutirtha Ghosh, Prasenjit Mukherjee, Baisakhi Chakraborty, and Rezaul Bashar. 2018. Automated generation of er diagram from a given text in natural language. In *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, pages 91–96. IEEE.

Tejas Gokhale, Hamid Palangi, Besmira Nushi, Vibhav Vineet, Eric Horvitz, Ece Kamar, Chitta Baral, and Yezhou Yang. 2022. Benchmarking spatial relationships in text-to-image generation. *arXiv preprint arXiv:2212.10015*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. 2022. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10696–10706.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

John Harrison. 1996. Hol light: A tutorial introduction. In *International Conference on Formal Methods in Computer-Aided Design*, pages 265–269. Springer.

Gérard Huet, Gilles Kahn, and Christine Paulin-Mohring. 1997. The coq proof assistant a tutorial. *Rapport Technique*, 178.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Predrag Janičić and Ivan Trajković. 2003. Wingclc: a workbench for formally describing figures. In *Proceedings of the 19th Spring Conference on Computer Graphics*, pages 251–256.

Zhen Jia, Zhang Zhang, Liang Wang, and Tieniu Tan. 2024. Human image generation: A comprehensive survey. *ACM Computing Surveys*, 56(11):1–39.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*.

Jill H Larkin and Herbert A Simon. 1987. Why a diagram is (sometimes) worth ten thousand words. *Cognitive science*, 11(1):65–100.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. 2023. Gligen: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22511–22521.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Xiaoxiao Ma, Mohan Zhou, Tao Liang, Yalong Bai, Tiejun Zhao, Huaian Chen, and Yi Jin. 2024. Star: Scale-wise text-to-image generation via auto-regressive representations. *arXiv preprint arXiv:2406.10797*.

Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*.

Josh Achiam OpenAI, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2024. Gpt-4 technical report, 2024. *URL https://arxiv. org/abs/2303.08774*, 2:6.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from

natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *International conference on machine learning*, pages 1060–1069. PMLR.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494.

Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. 2024. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11.

Muhammad Shahbaz, Syed Ahsan, Muhammad Shaheen, Rao Muhammad Adeel Nawab, and Syed Athar Masood. 2011. Automatic generation of extended er diagram using natural language processing. *Journal of American Science*, 7(8):1–10.

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

Jaskirat Singh, Stephen Gould, and Liang Zheng. 2023. High-fidelity guided image synthesis with latent diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5997–6006. IEEE.

Mandayam Srivas and Albert Camilleri. 1996. *Formal Methods in Computer-Aided Design: First International Conference, FMCAD'96, Palo Alto, CA, USA, November 6-8, 1996, Proceedings*, volume 1166. Springer Science & Business Media.

Keith Stenning and Jon Oberlander. 1995. A cognitive theory of graphical and linguistic reasoning: Logic and implementation. *Cognitive science*, 19(1):97–140.

László Szirmay-Kalos. 2003. *Proceedings of the 19th Spring Conference on Computer Graphics*. ACM.

Ming Tao, Hao Tang, Fei Wu, Xiao-Yuan Jing, Bing-Kun Bao, and Changsheng Xu. 2022. Df-gan: A simple and effective baseline for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16515–16525.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jingxuan Wei, Cheng Tan, Qi Chen, Gaowei Wu, Siyuan Li, Zhangyang Gao, Linzhuang Sun, Bihui Yu, and Ruifeng Guo. 2024. From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. *arXiv preprint arXiv:2411.11916*.

Makarius Wenzel, Lawrence C Paulson, and Tobias Nipkow. 2008. The isabelle framework. In *Theorem Proving in Higher Order Logics: 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings 21*, pages 33–38. Springer.

Wen-tsün Wu. 2008. On the decision problem and the mechanization of theorem-proving in elementary geometry. In *Selected Works Of Wen-Tsun Wu*, pages 117–138. World Scientific.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368.

Jinheng Xie, Yuexiang Li, Yawen Huang, Haozhe Liu, Wentian Zhang, Yefeng Zheng, and Mike Zheng Shou. 2023. Boxdiff: Text-to-image synthesis with training-free box-constrained diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7452–7461.

Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. 2018. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Zhengyuan Yang, Jianfeng Wang, Zhe Gan, Linjie Li, Kevin Lin, Chenfei Wu, Nan Duan, Zicheng Liu, Ce Liu, Michael Zeng, et al. 2023. Reco: Region-controlled text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14246–14255.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. 2022. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5.

Xinguo Yu, Wenbin Gan, Danfeng Yang, and Sichao Lai. 2015. Automatic reconstruction of plane geometry figures in documents. In *2015 International Conference of Educational Innovation through Technology (EITT)*, pages 46–50. IEEE.

Abhay Zala, Han Lin, Jaemin Cho, and Mohit Bansal. 2023. Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. *arXiv preprint arXiv:2310.12128*.

Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. 2023. Text-to-image diffusion models in generative ai: A survey. *arXiv preprint arXiv:2303.07909*.

Han Zhang, Jing Yu Koh, Jason Baldridge, Honglak Lee, and Yinfei Yang. 2021. Cross-modal contrastive learning for text-to-image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 833–842.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. 2017. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. 2018. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962.

Jing-Zhong ZHANG, Hui-Min Xiong, and Xi-Cheng Peng. 2007. Free software ssp for teaching mathematics. In *Symbolic Computation and Education*, pages 115–135. World Scientific.

Hu Zhengyu and Zhong Xiuqin. 2023. A precise text-to-diagram generation method for elementary geometry. In *2023 20th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 1–7. IEEE.

Yutong Zhou and Nobutaka Shimada. 2023. Vision+ language applications: A survey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 826–842.

A Implementation Details

We set the maximum number of verfication to be five. Both the generated diagram and the reference diagram are resized to 224 × 224 pixels and features are extracted using the CLIP image encoder, while the textual description is encoded with the CLIP text encoder. The time complexity of our custom solver is primarily dependent on the number of variables that require resolution. Consequently, we aim to leverage condition prior to reduce the number of variables. For instance, we may assume the origin of the circle to be at (0,0) and set the radius to 1 if no specific length is provided. This approach significantly accelerates the optimization solver. Furthermore, a time constraint of 60 minutes is imposed; if valid results are not obtained within this period, the problem is deemed unresolvable. We employ DeepSeek-V3 (Liu et al., 2024) for LLM, which is strong in reasoning and coding, in our experiments for both stage one and stage three. We also experiment different LLMs and report their effect in ablation study.

B Algorithm Overview

MagicGeo functions through a three-stage process designed to efficiently transform raw input into well-structured geometric diagram problems. In the initial stage, a LLM is employed to translate the raw input data into standardized formal language propositions, ensuring that the problem is represented in a consistent and clear format. The second stage introduces an optimization solver integrated with verification mechanisms. This solver generates the necessary point coordinates that are pivotal for constructing accurate geometric configurations. Finally, in the third stage, these point coordinates are utilized to construct geometric diagram problems, ensuring that the generated diagrams are both mathematically valid and visually representative of the initial problem. The entire process of MagicGeo is comprehensively outlined in Algorithm 1, which provides a step-by-step breakdown of the system's operation.

Algorithm 1: MagicGeo

Input: Textual description (T)
Output: Geometric diagram (D)

while *not reach the maximum number* **do**
  Stage 1:
  $Points, Constraints$ = $LLM(T, Prompt)$;
  Stage 2:
  if *Type_Break(*$Constraints$*)* **then**
    continue;
  end
  Extract $Variables$;
  Generate Value Combinations $Combos$;
  foreach *combo in Combos* **do**
    $Variables = combo$;
    $find = True$;
    foreach *cons in Constraints* **do**
      if *Constraint_Break(*$cons$*)* **then**
        $find = False$;
        break;
      end
    end
    if *find* **then**
      break;
    end
  end
  if *not find* **then**
    continue;
  end
  Stage 3:
  Assign *combo* to $Points$;
  $Code$ = $LLM(Points, T, Prompt)$;
  Render $Code$ to PDF;
  return;
end

C Prompt Details

Our framework leverages the zero-shot abilities of LLMs during the autoformalization phase, alongside coordinate-aware TikZ code generation. To better elucidate the details of our experimental setup, we provide a comprehensive description of the prompts used to guide the LLM's behavior. Specifically, we prompt the LLM to generate two key pieces of information: coordinates $Points$ represented by variables $Vars$ and the required geometric constraints $Cons$ based on these coordinates $Points$. We show the prompt instruction in Figure 7, which provides the LLM with clear instructions to formalize the geometric problem. In addition to that, our framework incorporates specific geometric constraint instructions to enhance the LLM's performance in this task. These instructions are designed to probe the LLM's capability to understand and encode geometric constraints in a formalized manner. The precise prompt used to

Instruction：
Based on the given mathematical geometry diagram's natural language description, use the provided geometric constraint rules to output the JSON information. The output format should match the following pattern, ensuring correct JSON syntax without any additional language description.

Output Json format:

```
{
  "points": [
    "Point object declarations, listing the names of the points used, represented by uppercase letters like "A1", "A2", where the coordinates of the points are expressed as variables."
  ],
  "constraints": [
    "Using the provided geometric constraint rules, convert the given mathematical geometry diagram's natural language into code that complies with Python syntax. For example: If I want to represent that angle CBD is 23°, the following string should be added to the constraints list: 'angle(C,B,D,23)'"
  ]
}
```

An example:

```
{
  "points": ["A": (x, y), "B": (u, v), "C": (p, q)],
  "constraints": ["angle(A,B,C,23)"]
}
```

Figure 7: The task instruction prompt for LLM in autoformalization.

guide the LLM through this process is presented in Figure 8. The final phase of our framework involves the generation of TikZ code that accurately represents the geometric diagram. The prompt provided to the LLM for this task is succinct yet explicit: "Please provide the LaTeX code for generating the corresponding image with the given correct coordinate positions."

D Failure Case Analysis

While MagicGeo shows significant promise, we conduct a thorough failure case analysis to identify areas for improvement. Our framework consists of three phases, with an example provided for each, as illustrated in Figure 9.

In stage 1, a key failure arises from the reliance on the LLM for autoformalization. Although the LLM is designed to formalize geometric constraints, there are instances where the formalized constraints may not fully adhere to necessary geometric rules. This discrepancy occurs when the formalization introduces constraints that are logically inconsistent with geometric principles, rendering them incompatible with the solver in subsequent steps.

In stage 2, as discussed in the limitations section, the solver may experience delays when handling complex diagrams. To improve efficiency, we provide a value range for the solver, which can lead to scenarios where correct autoformalizations still fail to produce a valid solution within the specified range. Future work will focus on investigating more efficient algorithms for handling larger ranges. Additionally, the solver may generate a technically correct solution that is visually suboptimal. Points may be placed too close together, creating a cluttered diagram, as shown in the figure. This issue can be addressed by the diagram editing technique presented in this paper or by adding a constraint to prevent excessive proximity between points.

In stage 3, failure again stems from the LLM, where points may receive correct coordinates, but the generated TikZ code is incorrect, such as improper labeling. For instance, the diagram in the figure shows two 'O' labels near the same point.

E Additional Visual Comparison

We provide additional qualitative visual comparison with baselines in Figure 10. Upon inspection, it is evident that both baselines fail to consistently adhere to the underlying geometric constraints, resulting in diagrams that exhibit significant inconsistencies. Our proposed method rigorously adheres to geometric constraints while simultaneously maintaining a high level of perceptual quality.

Geometric constraint instructions:

dist(O, A, r): If point A is on a circle with radius r and center O, use this function to represent the distance from point A to the center O as r.

angle(A, B, C, θ): If ∠ABC = **θ**, use this function to represent the angle as **θ**.

angle_relation(B, A, C, A, D, B, m): If ∠BAC = **m** * ∠ADB, use this function to represent the angle relationship, where **m** is the angle multiple.

online_inside(B, E, F): If point **B** is on the segment **EF**, use this function to check if **B** lies inside the segment **EF**.

online_extension(B, E, F): If point **B** is on the extension of segment **EF**, use this function to check if **B** lies on the extended line of segment **EF**.

ortho(A, B, E, F): If **AB** is perpendicular to **EF**, use this function to check the perpendicularity of **AB** and **EF**.

midpoint(A, B, C): If point **A** is the midpoint of **BC**, use this function to indicate that **A** is the midpoint of **B** and **C**.

arc_midpoint(A, B, C): If point **A** is the midpoint of arc **BC**, use this function to represent **A** as the midpoint of arc **BC**.

equal_line(A, B, C, D): If line segments **AB** and **CD** are equal, use this function to check if the lengths of **AB** and **CD** are equal.

angle_bisector(A, D, C, A, B): If **AD** bisects ∠CAB, use this function to represent **AD** as the angle bisector of ∠CAB.

parallel(A, B, C, D): If **AB** is parallel to **CD**, use this function to check if **AB** and **CD** are parallel.

line_ratio(D, C, B, D, m): If line segment **BD** is **m** times the length of line segment **DC**, use this function to represent the ratio.

is_point_in_triangle(A, B, C, P): If point **P** is inside triangle **ABC**, use this function to check if **P** lies inside triangle **ABC**.

is_point_out_triangle(A, B, C, P): If point **P** is outside triangle **ABC**, use this function to check if **P** lies outside triangle **ABC**.

is_acute_triangle(A, B, C): If triangle **ABC** is an acute triangle, use this function to check if triangle **ABC** is an acute triangle.

...

Figure 8: The geometric constraint instruction prompt for LLM to guide the autoformalization.

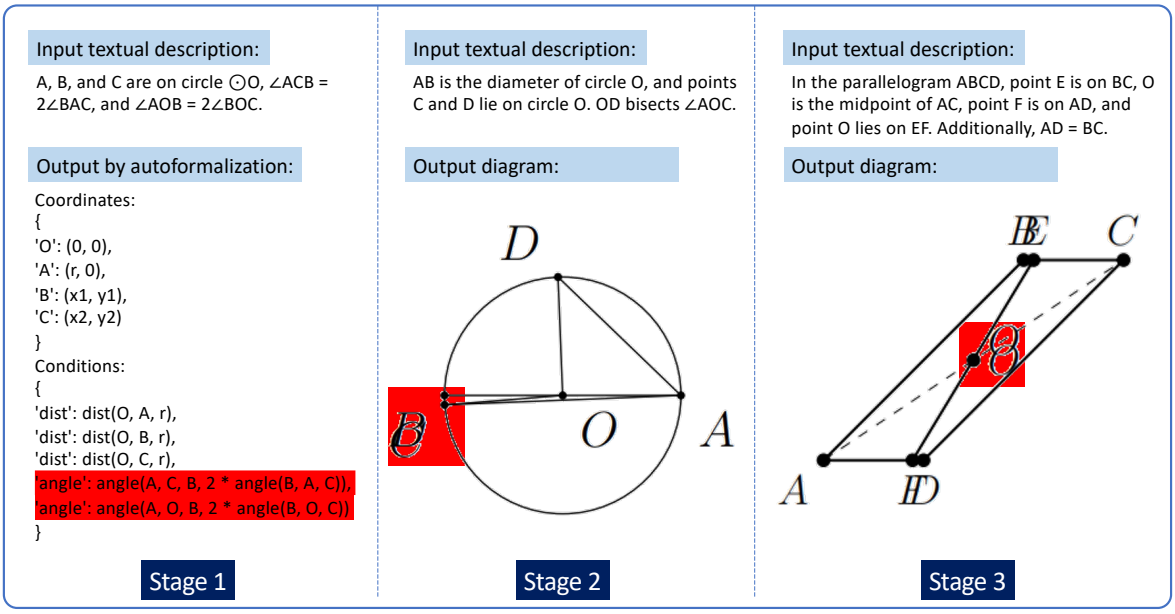


Figure 9: Failure case analysis in our framework.

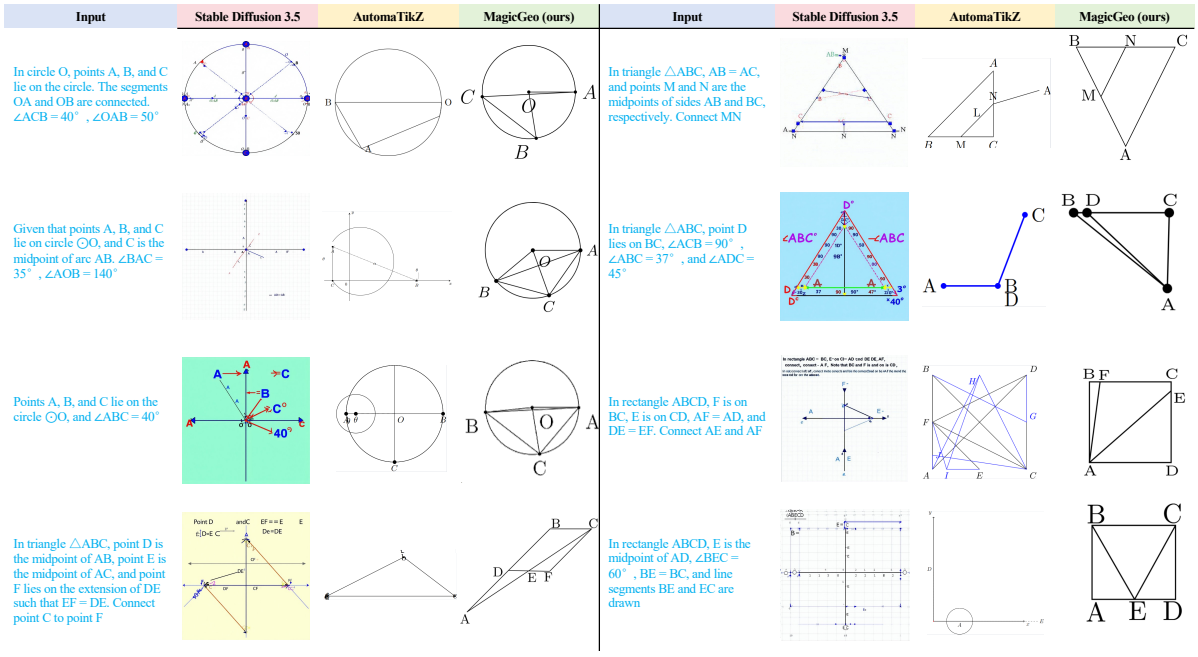


Figure 10: Additional qualitative comparison with other approaches. Our method generates results that rigorously adhere to geometric constraints while maintaining high perceptual quality.