Proceedings of the ASME 2024
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC-CIE2024
August 25-28, 2024, Washington, DC

DETC2024-143740

LLM4CAD: MULTI-MODAL LARGE LANGUAGE MODELS FOR 3D COMPUTER-AIDED DESIGN GENERATION

Xingang Li, Yuewan Sun, Zhenghui Sha*
Walker Department of Mechanical Engineering
University of Texas at Austin
Austin, Texas 78712
Email: zsha@austin.utexas.edu

ABSTRACT

The evolution of multimodal large language models (LLMs) capable of processing diverse input modalities (e.g., text and images) holds new prospects for their application in engineering design, such as the generation of 3D computer-aided design (CAD) models. However, little is known about the ability of multimodal LLMs to generate 3D design objects, and there is a lack of quantitative assessment. In this study, we develop an approach to enable two LLMs, GPT-4 and GPT-4V, to generate 3D CAD models (i.e., LLM4CAD) and perform experiments to evaluate their efficacy. To address the challenge of data scarcity for multimodal LLM studies, we created a data synthesis pipeline to generate CAD models, sketches, and image data of typical mechanical components (e.g., gears and springs) and collect their natural-language descriptions with dimensional information using Amazon Mechanical Turk. We positioned the CAD program (programming script for CAD design) as a bridge, facilitating the conversion of LLMs' textual output into tangible CAD design objects. We focus on two critical capabilities: the generation of syntactically correct CAD programs (Cap1) and the accuracy of the parsed 3D shapes (Cap2) quantified by intersection over union. The results show that both GPT-4 and GPT-4V demonstrate potential in 3D CAD generation. Specifically, on average, GPT-4V outperforms when processing only text-based input, exceeding the results obtained using multimodal inputs, such as text with image, for Cap 1 and Cap 2. However, when examining category-specific results of mechanical components, while the same trend still holds for Cap 2, the prominence of multimodal inputs is increasingly evident for more complex geometries (e.g., springs and gears) in Cap 1. The potential of multimodal LLMs in enhancing 3D CAD generation is clear, but their application must be carefully calibrated to the complexity of the target CAD models to be generated.

Keywords: Multimodal Large Language Models, GPT-4, GPT-4V, Computer-Aided Design, Generative Design

1 INTRODUCTION

The emergence of large language models (LLMs), including the generative pre-trained transformer (GPT) series [1], represents a significant advancement in the capabilities of artificial intelligence (AI) to interact with the world. These models, trained on vast datasets, exhibit remarkable proficiency in "understanding" the nuances of human language and generating text that mirrors human-like communication [2]. However, the inherent vagueness of natural language continues to pose a significant challenge, especially when it comes to conveying complex instructions to LLMs. To this end, cutting-edge multimodal LLMs, such as OpenAI's GPT-4 Vision (GPT-4V) [3] and Google's PaLM-E [4], have been proposed. These models are designed to process more input modalities besides text, such as images, thereby broadening how users can interact with LLMs for more sophisticated tasks.

The utility of LLMs in processing natural language data has extended their application in design research for conceptual de-

*Corresponding author.

sign [5–7]. One particular limitation of these studies is that they use textual information only as the input. However, it might be difficult to effectively describe intended design artifacts and associated parameters through text only, which often encompass the structural and layout specifications of a component and the desired shapes of the component. Furthermore, conceptual design is inherently multimodal, frequently incorporating visual elements ranging from sketches for design ideation to engineering drawings for fabrication and assembly [8–10].

Therefore, recent design research emphasizes the significance of multimodal machine learning (MMML) in improving the conceptual design by integrating diverse modalities [8, 10, 11]. Multimodal input, such as images and sketches beyond texts, could potentially improve LLMs' performance in understanding designers' intent, thus generating more precise and quality design output. Therefore, multimodal LLMs that can take multiple input modalities have the potential for their application in AI-assisted conceptual design, promising to revolutionize design tools and human-AI collaborative design. However, to our knowledge, while a few works conducted qualitative evaluations using the ChatGPT interface [12, 13], no quantitative evaluation has been performed to assess the efficacy of multimodal LLMs in the CAD generation of 3D shapes for conceptual design.

To fill this research gap, we develop an approach to enable multimodal LLMs for ***3D CAD Generation*** (hereafter referred to as LLM4CAD) and conduct a quantitative analysis to evaluate LLM4CAD's effectiveness in conceptual design. Specifically, we seek to understand the capabilities of multimodal LLMs to generate high-quality 3D design concepts with precise dimensions and to identify strategies to improve their capabilities. Unlike methods that directly generate 3D models [14, 15], the direct output of our method is CAD programs, which can be compiled into 3D CAD models. The programming codes effectively represent a sequence of CAD operations, which offers insight into the historical construction process and the associated design knowledge of 3D CAD modeling. Additionally, it supports geometry modification through parametric modeling (e.g., modifying the radius of a *Circle* to change the size of the inner hole of a nut). This study is driven by two research questions (RQs): *1) To what extent can multimodal LLMs generate 3D design objects when employing different design modalities or a combination of various modalities? 2) What strategies can be developed to enhance the ability of multimodal LLMs to create 3D design objects?*

To enable LLM4CAD, one technical challenge is that LLMs cannot directly create 3D shapes, such as meshes, voxels, and boundary representations. However, LLMs can generate programs in languages such as Python [12,13,16]. Therefore, we developed an approach to enable an indirect synthesis of 3D design objects by generating CAD programs. To quantitatively evaluate the performance, we propose a data synthesis pipeline along with an evaluation framework. This evaluation specifically focuses on two capabilities. **Cap1**: the success rate of the generated programming codes in program-to-CAD translation, and **Cap2**: the extent to which these resultant 3D design objects align with the ground-truth shapes. We summarize our contributions as follows.

1. This study created a new CAD dataset of five categories of mechanical components (i.e., shafts, nuts, flanges, springs, and gears with diverse geometry complexity) for multimodal LLMs, including textual descriptions, sketches, images, and 3D CAD models. In particular, textual descriptions of the target design objects are in natural languages with detailed dimensional information collected with Amazon Mechanical Turk (AMT) [1], an online crowdsourcing platform.
2. The effectiveness of the GPT-4 and GPT-4V models in 3D design generation was evaluated, and new knowledge of their strengths and limitations was obtained.
3. A new method was developed and implemented to enhance the GPT models' proficiency in generating 3D CAD models. Specifically, we develop a debugger to correct syntax errors in the synthesized CAD programs to improve their success rate of being translated to 3D CAD models.

We found that the GPT-4 and GPT-4V models demonstrated the potential of LLM4CAD, especially when enhanced by the proposed debugger. However, they still struggle with handling complex geometries. Additionally, GPT-4V's performance was examined with four input modes including text-only, text with sketch, text with image, and a combination of text, sketch, and image. The results show that, on average, GPT-4V excels when processing purely text-only input, outperforming multimodal inputs in Cap 1 and Cap 2. This observation is counterintuitive because a prevailing belief in the field of MMML is that incorporating varied input modalities should improve a machine learning (ML) model's predictive accuracy due to an increased amount of information for learning and inference. However, when examining category-specific results of mechanical components, while the same trend is still observed for Cap 2, multimodal inputs start to gain prominence with more complex geometries (e.g., springs and gears) in Cap 1.

Based on these observations, it is clear that the current multimodal LLMs (e.g., GPT-4V) still face limitations in handling multimodal inputs for generating 3D CAD objects. However, the detailed insights from the category-specific results show that multimodal inputs become more effective as the complexity of design objects increases. Therefore, these limitations do not diminish their potential benefits in real-world design scenarios characterized by complex objects. The ability of multimodal LLMs to process diverse input modalities remains a promising avenue for enhancing 3D CAD generation technologies.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of the background related to multimodal machine learning and LLMs for engineering design.

[1] https://www.mturk.com/

Section 3 outlines the methodology for data collection and generation, as well as the evaluation of multimodal LLMs. Subsequently, Sections 4 and 5 present, analyze, and discuss the experimental results, from which we summarize the primary findings and acknowledge limitations. Conclusions and closing remarks are made in Section 6, where we present key insights and suggest potential directions for future research.

2 LITERATURE REVIEW

In this section, we review the most relevant literature to our work, including multimodal machine learning and large language models for engineering design.

2.1 MULTIMODAL MACHINE LEARNING IN ENGINEERING DESIGN

Multimodal machine learning (MMML) approaches exhibit significant promise in enhancing the field of engineering design, as evidenced by recent review studies [8, 10]. Specifically, when confronted with inputs comprising multiple modalities, such as a combination of text and sketches, MMML techniques can integrate this information through a process known as multi-modal fusion. This fusion enables integrating data from diverse modalities to facilitate prediction tasks such as regression or classification. The application of multimodal fusion in different areas (e.g., audio-visual speech recognition, image captioning) is becoming popular [17]. The data from different modalities can supplement each other, aiding in increasing the accuracy of predictions. Even if one modality is missing, predictions can still be viable. While there might be overlap in information from multiple modalities, this redundancy can strengthen the reliability of the predictions [17].

Despite these advantages, the extent to which multimodal fusion can enhance engineering design remains largely unexplored, with only a few pioneering works looking into this area [18, 19]. For example, Song, Miller, and Ahmed [18] pioneered a multimodal learning model that integrates sketch and textual description modalities using a cross-attention mechanism. This approach facilitated a comprehensive assessment of design concepts, revealing that MMML significantly enhances the model's predictive and explanatory capabilities. The findings underscore the advantages of employing multimodal representations in conceptual design evaluation.

MMML for engineering design is still in its initial stage, presenting ample opportunities for extensive research exploration into the theory and methodology for enhancing design evaluation and generation. Our study investigates multimodal large language models (LLMs)' capability to generate 3D design concepts when taking multiple design modalities (e.g., a combination of text, image, and sketch) as input compared to unimodal input (e.g., text), contributing to the field of MMML for engineering design.

2.2 LARGE LANGUAGE MODELS FOR ENGINEERING DESIGN

Natural language processing (NLP) is a foundational technology in AI advancements, primarily focusing on enabling computers to understand and interact with humans using natural language [20]. Building upon the foundation of various NLP technologies, the emergence of LLMs, such as the generative pre-trained transformer (GPT) series [1], marks a significant leap in AI proficiency.

A remarkable example of LLMs is ChatGPT [21], launched in 2022 by OpenAI. ChatGPT, an advanced Chatbot built upon the GPT-3.5 model, provides detailed and structured responses based on specific user prompts. Its capabilities span a broad spectrum of language understanding and generation tasks, including multilingual translation, creative writing, and programming code creation and debugging. A distinctive feature of ChatGPT is its ability to recall previous conversation segments, enabling more coherent and sustained interactions [22, 23]. ChatGPT is the state-of-the-art LLM and stands apart from earlier NLP and LLM tools due to its exceptional conversational skills and reasoning abilities across various domains [24–26].

Researchers have been examining how ChatGPT can be applied to enhance the engineering design process, from conceptual design to manufacturing [5, 6, 12, 16, 27]. For instance, Kocaballi [5] undertook a hypothetical design project that leveraged ChatGPT to create personas in the roles of designers or users. The approach facilitated various design-related activities, including conducting user interviews, generating design concepts, and evaluating user experiences. However, these studies primarily focus on the text generation capabilities of LLMs by taking textual input. Taking the generation of design concepts as an example, it can be advantageous to employ the generated text for brainstorming design ideas [7]. However, translating these conceptual ideas into concrete 3D designs still presents a significant challenge.

While LLMs' ability to directly generate 3D objects (e.g., meshes, voxels, and boundary representations) seems limited, an alternative approach involves the generation of 3D designs using CAD programming languages such as CADQuery and OpenSCAD. This can be achieved by exploiting LLMs' capacity for program synthesis [28], and some research has been investigating the potential of LLMs in producing 3D designs through CAD programs, which involves interpreting human language instructions and converting them into CAD designs [12, 16]. Nevertheless, these studies are still limited to textual descriptions for the design intent, and it is often challenging to convey complex tasks solely through text.

The evolution of OpenAI's GPT architecture, transitioning from the text-only GPT-3.5 and GPT-4 to its multimodal successors, GPT-4 Vision (GPT-4V) [3, 23, 29] marks significant

advancements. It offers opportunities to incorporate multiple modalities besides text. However, little is known about its practicality in and for engineering design, such as 3D CAD generation. That motivates our study to conduct a quantitative analysis on to what extent multimodal LLMs can generate 3D design objects when employing different design modalities or a combination of various modalities. We employed GPT-4 and GPT-4V as examples of unimodal and multimodal LLMs for our experiments due to their acknowledged outstanding performance.

3 METHODOLOGY

We develop an approach to enabling LLM4CAD by taking various design modalities and assessing the extent of their capabilities, as shown in Figure 1. This approach consists of three major steps: 1) Data Synthesis: multimodal design data collection and generation, 2) Code Generation: CAD program code generation, and 3) Evaluation: the evaluation of 3D CAD model generation in terms of success rate and precision.

For clarity of illustration, we consider a gear as a representative 3D design object. The process begins with the generation of ground-truth (GT) 3D CAD models; dimensional data is recorded alongside the generation process. Direct rendering techniques can be employed to obtain images from 3D shapes. Meanwhile, textual descriptions incorporating dimensional information and sketches can be acquired through automated algorithms or human involvement. With the input data in three design modalities (i.e., text, image, and sketch), we evaluate the capability of the GPT-4 and GPT-4V models to generate CAD programs, which are then parsed into 3D shapes. The quality of the resulting 3D shapes is then benchmarked against the GT shapes to gauge the efficacy of generation automatically (see more details in Section 3.3). In addition, we proposed a debugger to enhance the models' capabilities in CAD program generation.

3.1 DATA SYNTHESIS

We chose mechanical components as the target 3D objects, given their pivotal role in engineering design. Upon examining the literature and online resources, we could not find any CAD model dataset of mechanical components that incorporates multiple design modalities and detailed dimensional information. The existing datasets of mechanical components [30–32] do not provide essential dimensional data. Therefore, they are not appropriate for this study because a quantitative evaluation of the generated CAD models is impossible since no dimensional information can be provided to LLMs as input. This motivates us to develop a new synthesis pipeline for CAD objects with detailed dimensional information. Such a dataset would benefit various machine learning tasks in engineering design, where the details of the design specifications are critical.

As shown in Figure 2, a semi-automated pipeline for Data

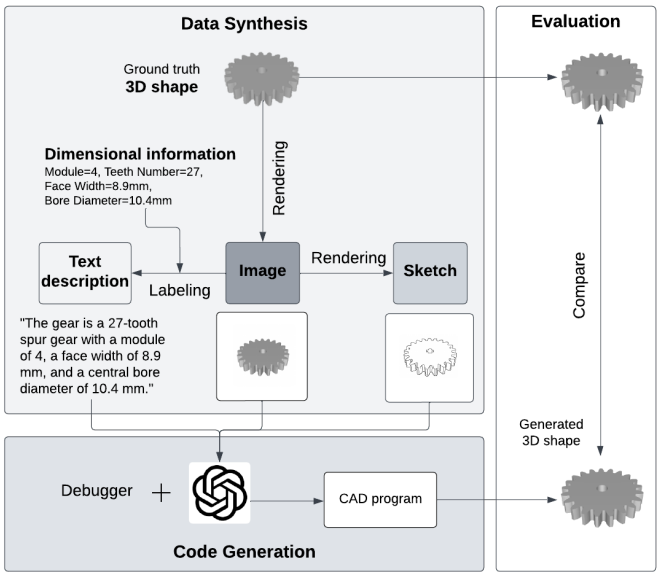

FIGURE 1. The overview of our approach

Synthesis is developed to generate the textual descriptions, images, sketches, and GT 3D shapes of five common types of mechanical components: shafts, nuts, flanges, springs, and gears. They were chosen for their popularity in engineering design and their varying levels of complexity, allowing us to test the robustness of our approach and investigate how geometric complexity would influence the results. The complexity of mechanical components in this study is determined by the solvability of a design problem as introduced by Summers and Shah [33] (i.e., less solvable is more complex). Specifically, solvability here refers to the capability of LLMs to formulate a CAD program to meet the input design requirements. It is relatively simple to generate CAD programs for components such as shafts, nuts, and flanges, which primarily utilizes *Sketch* and *Extrude* CAD operations. However, creating CAD programs for springs and gears presents more challenges due to longer CAD sequences (i.e., more lines of code) or special auxiliary Python packages. For example, the *Sweep* CAD operation is needed to create springs. Auxiliary Python packages (e.g., cq_gears [2]) or complex calculations (e.g., understanding the fundamental geometry of the gear teeth to perform detailed geometric and trigonometric computations) are needed to determine the gear tooth profiles. Moreover, complex components need more curvatures to describe geometries and more triangles to represent a shape digitally using the STL format. In our dataset, the average number of triangles for springs and gears is around 200k and 10k, respectively. In contrast, flanges, nuts, and shafts have an average of 1,500, 500, and 1,500 triangles, respectively. This difference in the num-

[2] https://github.com/meadiode/cq_gears

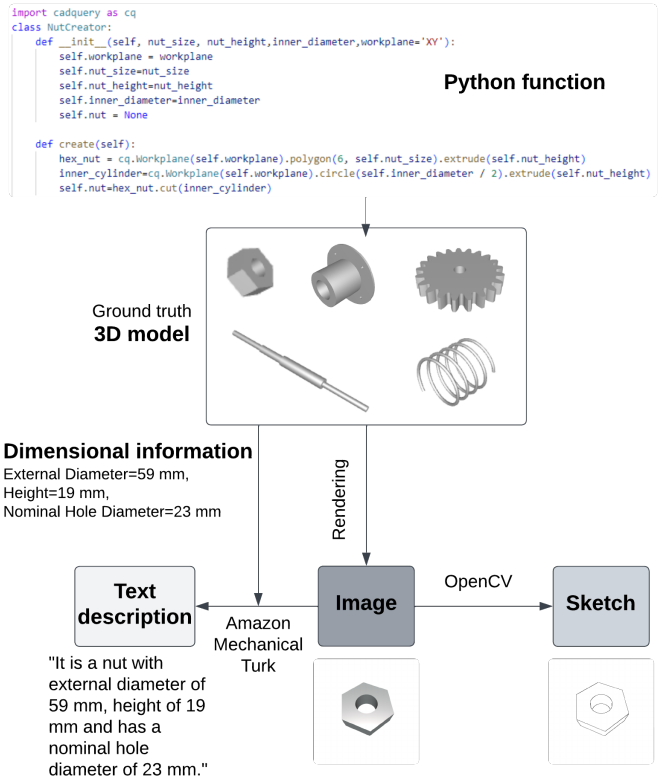

FIGURE 2. The pipeline for Data Synthesis

ber of triangles indicates varying levels of geometric complexity. Therefore, shafts, nuts, and flanges are referred to as relatively simple geometries, while springs and gears are more complex.

3.1.1 3D SHAPES, IMAGES, AND SKETCHES

Using CADQuery (Version 2.3.1) [3], a Python-based CAD programming language, we created five distinct Python classes, each corresponding to one of the mechanical component categories. In each class, the design is parameterized, so a variety of designs can be generated from a defined design space. An example of the classes is given in Figure 2. To achieve uniform sampling of the design space, we employed Latin Hypercube Sampling (LHS) [34] of design parameters (such as the external diameter and height of a nut). LHS is often used in computer experiments and simulations to efficiently sample points from a high-dimensional space. For shafts, we synthesized 250 shapes for each of the four types of shafts (with each type having 2, 3, 4, and 5 sections, respectively), totaling 1,000. For the other four components, 1,000 shapes for each are created. The dimensional information of these shapes was recorded alongside the GT 3D

[3] https://cadquery.readthedocs.io/en/latest/installation.html

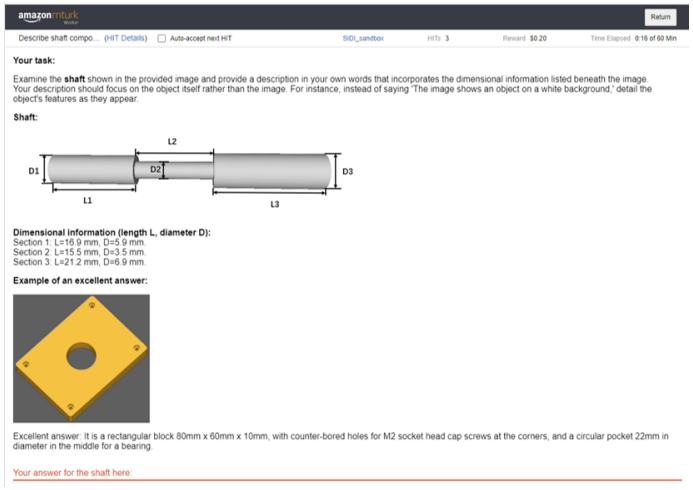

FIGURE 3. An example of the HITs on Amazon Mechanical Turk

models. A piece of the dimensional information of a nut is given in the figure.

The 2D image representation of these 3D shapes is obtained from computer rendering. Subsequently, sketches of these images were produced using sketch-style rendering using OpenCV. Although hand sketches of mechanical components from human participants would be a better data source for research validity, the efficiency and effectiveness of rendered sketches from computer algorithms have been demonstrated [32]. With the consideration of such trade-offs, we decided to use computer renderings for the sketch data.

3.1.2 TEXTUAL DESCRIPTIONS WITH DIMENSIONAL INFORMATION

It is feasible to synthesize textual data integrated with dimensional information from images via GPT models [35]. However, as we need to input this textual data into GPT models for analysis, it might introduce a risk of biasing the results. To that end, we tested the other popular automatic captioning methods, such as the Contrastive Language-Image Pretraining (CLIP) model [36], but found the results unsatisfactory for mechanical components.

To avoid the potential biases introduced by LLMs and ensure the quality of the textual data, we chose to crowdsource textual descriptions through Amazon Mechanical Turk (AMT), a platform renowned for its efficacy in gathering data across a broad demographic spectrum. This diversity, spanning geographical, cultural, and age-related differences, is crucial for the richness of our dataset and aligns with established precedents in engineering design research for collecting data on human subjects [37, 38]. We designed human intelligence tasks (HITs) on AMT to recruit participants (known as Workers) for our study. These individuals were instructed to describe mechanical components in natural language based on provided images. A critical requirement

of these descriptions was incorporating specific dimensional information, which was presented alongside the images. This approach ensures that our data collection method not only captures the varied interpretations of mechanical components but also includes precise dimensional information, enhancing the utility and accuracy of the dataset.

For the five distinct mechanical component categories — shafts, nuts, flanges, springs, and gears — each category is represented by a unique standardized image for visual depiction within a HIT. Specifically, the category of shafts is further distinguished by incorporating four separate HITs and each HIT with a standardized image. These images correspond to the four distinct types of shafts, which are categorized based on the number of sections they contain. Thus, eight HITs were created and published on the AMT marketplace to be visible to Workers, corresponding to the four types of shafts and the other four mechanical components as aforementioned. In the current implementation, 400 assignments were successfully published for each category of the mechanical components (100 for each of the 4 HITs of shafts (100x4) and 400 for each of the other 4 HITs). Within a single HIT, every assignment featured the same image with its dimensional information. According to the rules of AMT, once a participant completes an assignment within a HIT, they cannot work on the other assignments within the same HIT, thereby avoiding repetitive responses. An example of an assignment under the HIT for triple-section shafts is shown in Figure 3. Accompanying each image, a piece of dimensional information describing the component is provided. Annotations are included on each image to highlight key features of the mechanical components and clarify the relationship between the dimensional information and the component's features. Furthermore, an example of a bearing pillow block with a human's description incorporating dimension information is provided as a reference to aid participants in understanding the task's requirements.

After completing the data collection via AMT, we conducted a cleaning process for the textual data to ensure the accuracy, consistency, and relevance of the information provided by the participants. Approximately 50% of the responses were deemed to be of high quality. The final dataset included a collection of textual descriptions: 251 for shafts, 217 entries for nuts, 206 for flanges, 218 for springs, and 231 for gears. After cleaning, we replaced the generic dimension information within the textual descriptions with specific, accurate specifications paired with the corresponding mechanical components. The integration of dimension information is expected to significantly enhance the richness and applicability of our dataset. The statistics of textual data and representative samples are presented in Table 1.

TABLE 1. Statistics of the textual data collection and cleaning process accompanied with representative examples

	Published tasks	Filtered responses	Examples
Shafts	400	251	It is a shaft which have four section. In first it have 14.9mm length and 15.5mm diameter. In second, it have12.8mm length and 21.3mm diameter. In third, it have 20.3mm length and21.1mm diameter. In fourth it have 25.6 mm length and 2.6 mm diameter.
Nuts	400	217	It is a hexagon nut with an external diameter of 47mm, a nominal hole diameter of 7mm and a height of 14mm.
Flanges	400	206	It is a Flange which has 124mm diameter and 14mm thickness with raised face which has 86mm diameter and 16mm bore diameter with 144mm face height.
Springs	400	218	The spring is a coil with a diameter of 8mm and a pitch of 14mm. It's 46mm long when uncompressed, made of wire with a 1.5mm radius. It seems strong and flexible, suitable for many uses.
Gears	400	231	It is a gear which has 6 module and 44 teeth number with face width is 8.7mm and a bore diameter which is equal to 19.3mm.

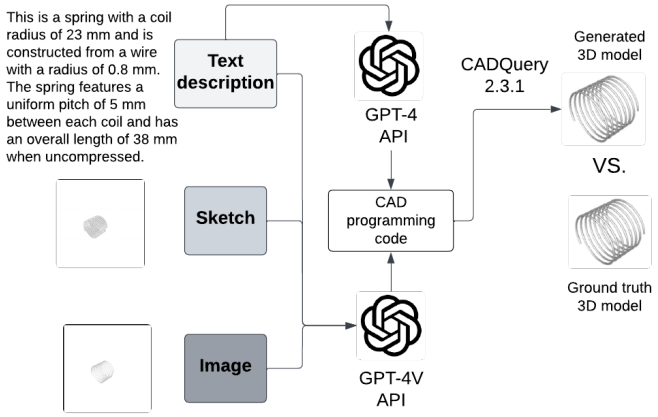

FIGURE 4. The pipeline for Code Generation and Evaluation

3.2 CODE GENERATION AND DEBUGGER

We show the pipeline of the Code Generation and Evaluation processes in Figure 4. The experiment was conducted by utilizing the models' application programming interface (API) and instructing them to generate CAD program code via CADQuery. Other options for CAD programming exist, such as OpenSCAD. It allows the creation of 3D CAD models using Python within the software itself and thus cannot be integrated into our automatic pipeline. On the other hand, CADQuery offers a native Python package that enables the execution of CAD programming code with the Python environment, making it a more suitable choice for our needs. Similar to the generation of GT 3D shapes, we employed Version 2.3.1 for CADQuery here as well. To interact with the OpenAI API model, we assign a persona to it, defining it as an AI assistant specialized in designing 3D objects with CadQuery. We initiate the request with a combination of a description and a specific prompt. The given prompt instructs: **"Generate CadQuery code to construct the specified mechanical component. The code must exclusively utilize CadQuery and**

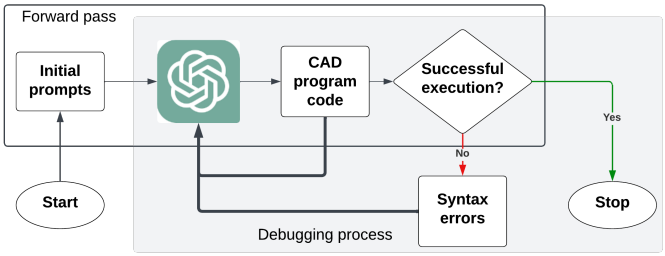

FIGURE 5. The process of generating CAD program code using GPT models with a debugger that can iteratively correct the syntax errors (if any) of the CAD program code. The generated CAD program code will be fed back to the GPT model in the debugging process.

can not incorporate any other CAD design packages or software, ultimately exporting the component as an STL file." The resulting CAD program was subsequently converted into 3D shapes for analysis.

To enhance the quality of the GPT models' output, we proposed a debugger as shown in Figure 5 integrated with the "forward pass" as described previously. The initial prompts (e.g., textual, image, and sketch data) conceivably represent user inputs, commands, or parameters that directly influence the code synthesis mechanism. The "forward pass" ends after executing the generated CAD program code, no matter if the execution is successful or not, which is used to test the inherent capability of GPT models. For the "debugging process," the code is subjected to an execution trial to ascertain its functional integrity. In the event of a successful execution, the process will be terminated. Conversely, an unsuccessful execution indicates the presence of syntax errors within the code, requiring the activation of the debugger. Syntax errors encompass a spectrum of programming language misuse, such as typographical errors to the misapplication of language constructs. The "debugging process" is an iterative procedure dedicated to the identification and correction of errors in the code. Both the previous conversation content (including the user requirements and GPT's responses) and the associated error messages are fed to the same API for the "debugging process". This recursive process is imperative to refine the CAD program code, ensuring its accuracy and reliability before finalization.

3.3 EVALUATION

We employed two key metrics to quantify the capabilities of LLM4CAD: the parsing rate for Cap1 and the intersection over union (IoU) for Cap2. The parsing rate metric evaluates the extent to which the generated CAD program code could be parsed successfully without errors, acknowledging that generating error-free code by GPT models is not guaranteed. Upon successful parsing, the quality of the resulting 3D shapes is measured against the GT shapes by calculating the IoU, thus providing a quantifiable measure of the generation accuracy relative to the input modality. The IoU metric, a critical measure of accuracy, quantifies the overlap between the generated shape and the GT shape as a ratio of their intersection to their union. This metric is widely used and particularly insightful for evaluating the geometric fidelity of the generated designs relative to the GT. Given our focus on the geometry of the generated shapes rather than their positions within a given space, we implemented a prestep to align the principal axes of the generated shapes with those of the GT shapes by rotation and translated the generated shapes to align their centroids with those of the GT shapes. This transformation process ensures that the calculation of IoU is based only on the geometric accuracy of the shapes, excluding any discrepancies that might arise from their positioning or orientation.

TABLE 2. Details of the experiment settings

Mechanical components	Input modality	Model	Metrics
3D shapes • Shaft (4 types) • Nut • Flange • Spring • Gear	Text	gpt-4-1106-preview (GPT-4)	Parsing rate and IoU
		gpt-4-1106-preview (GPT-4) + debugger*	
	Text	gpt-4-1106-vision-preview (GPT-4V)	
	Text + sketch		
	Text + image		
	Text + sketch + image	gpt-4-1106-vision-preview (GPT-4V) + debugger*	

*: Debugging times = 3

4 EXPERIMENTS AND RESULTS

In this section, we introduce the details of the experiment and the results.

4.1 EXPERIMENT DETAILS

The details of our experiment settings are outlined in Table 2. We conducted a comparative analysis between GPT-4 and GPT-4V. The API models "gpt-4-1106-preview" and "gpt-4-1106-vision-preview" were employed for GPT-4 and GPT-4V, respectively [4]. These represented the most up-to-date versions of the API available at the time of our study. While GPT-4 accepts only textual input, GPT-4V can process both textual and rasterized data inputs. We explored various modalities and combinations thereof as inputs for the GPT-4V model. In theory, there are other possible input modes (such as sketch only and sketch + image). However, they do not provide dimensional information from the textual descriptions for the GPT models and cannot fulfill our objective of conducting a quantitative comparison between the generated 3D design objects and their GT counterparts. As a result, our selection was strategically narrowed

[4] https://platform.openai.com/docs/models/overview

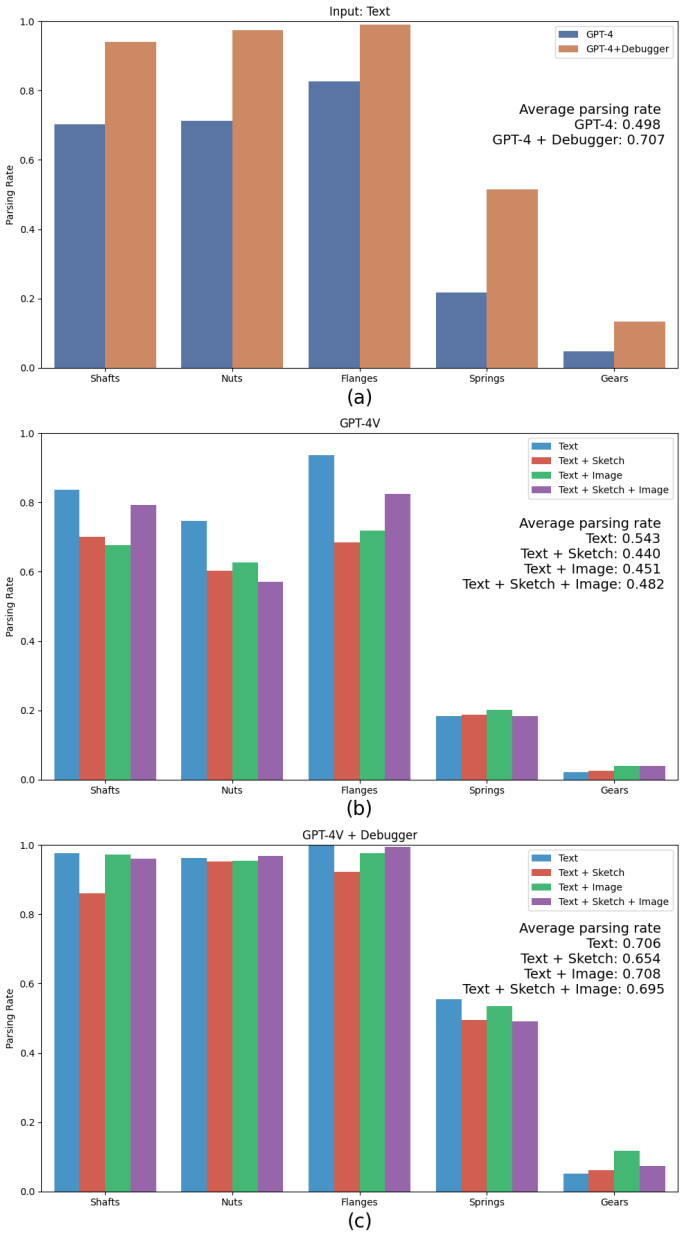

FIGURE 6. Results of the parsing rate for the five categories of shafts, nuts, flanges, springs, or gears. (a) GPT-4 VS. GPT-4 + Debugger with text-only input; (b) GPT-4V model; (c) GPT-4V + Debugger.

down to input modes that include textual descriptions, ensuring the necessary dimensional data is available for accurate analysis and comparison. In both scenarios, we first assessed GPT models' inherent capabilities, followed by the implementation of the debugger to evaluate its effectiveness in improving model performance. Specifically, we limited the debugging process to three times in the current study.

4.2 RESULTS

In this section, we present the results of the parsing rate and IoU. Additionally, we show examples to qualitatively compare the generated 3D design objects with their corresponding ground-truth (GT) shapes.

4.2.1 RESULTS OF THE PARSING RATE

Figure 6 shows the results comparing the parsing rates of the GPT-4 and GPT-4V models in various categories of mechanical components with or without the debugger. The average parsing rate values of both models are also annotated in the figure. Overall, there is a variance in model performance relative to the complexity of the mechanical components being parsed. Both models demonstrate higher efficacy in generating code for simple geometries, such as shafts, nuts, and flanges, than complex geometries (e.g., springs and gears).

Figure 6 (a) details the performance of the GPT-4 model when processing text inputs. It is observed that the inclusion of a debugger significantly enhances the model's parsing rate. In Figure 6 (b), the analysis extends to the GPT-4V model dealing with multiple input modalities, including text-only, text with sketch, text with image, and a combination of text, image, and sketch. In terms of the average parsing rate, the GPT-4V model achieves its highest performance with the text-only input mode, while the results are relatively consistent across the other three input types. For each category of the mechanical components, the text-only input achieves the best in shafts, nuts, and flanges. However, when dealing with more complex geometries (e.g., springs and gears), multimodal input modes are better than or as good as the text-only input. For example, the input of text with image is the best in springs, and the input using a combination of text, sketch, and image achieves the best in gears. Figure 6 (c) mirrors the trend observed in the GPT-4 model, demonstrating an improved parsing rate with the introduction of a debugger, but the difference in the parsing rate across the four input modes is reduced. The input of text with image is the best for gears. For the other four components, the multimodal input modes are as good as the text-only input.

A comparative analysis focusing on text-only inputs between the GPT-4 and GPT-4V models indicates a significant difference in performance. Specifically, the GPT-4V model exhibits a superior average parsing rate (0.543) compared to its GPT-4 counterpart (0.498). However, this advantage diminishes upon the integration of a debugging process (0.707 for GPT-4 VS. 0.706 for GPT-4V).

4.2.2 RESULTS OF THE INTERSECTION OVER UNION

Figure 7 shows the performance evaluation of the GPT-4 model, presenting (a) an overview of the average performance and (b) a detailed breakdown of the performance by component category. In terms of overall performance shown in

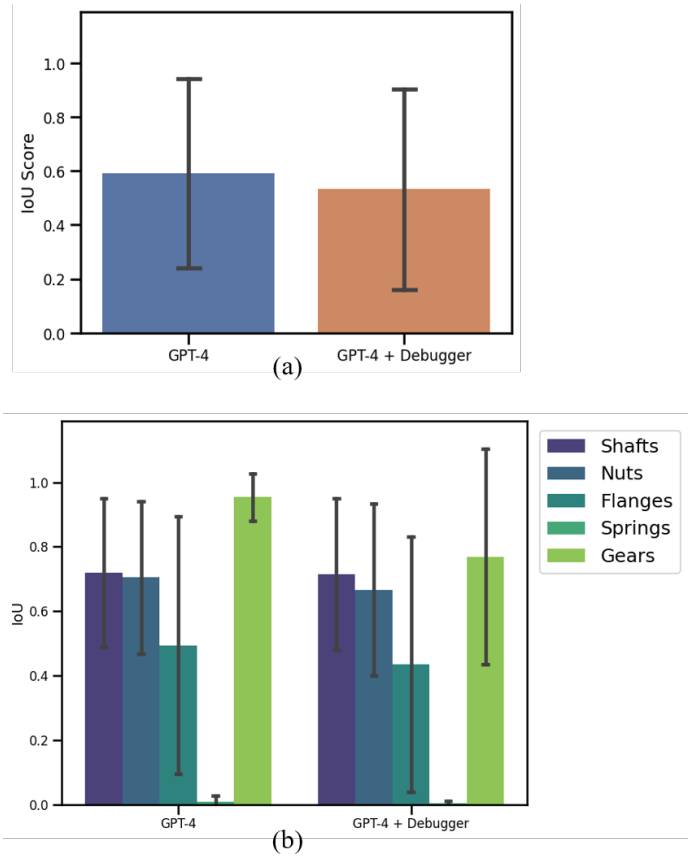

FIGURE 7. Results of IoU of (a) the overall performance and (b) the category-specific performance of the GPT-4 model

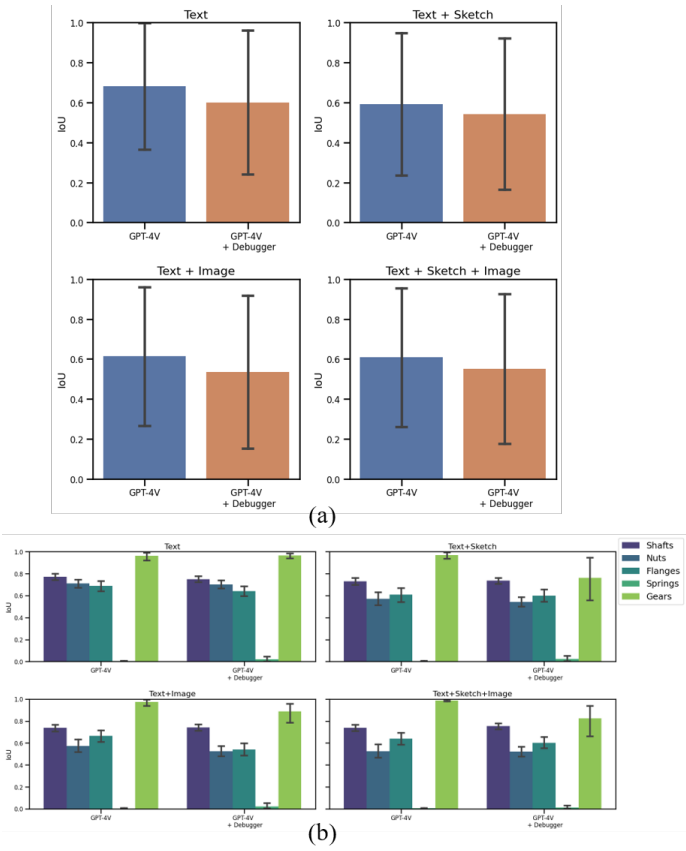

FIGURE 8. Results of IoU of (a) the overall performance and (b) the category-specific performance of the GPT-4V for four input modes

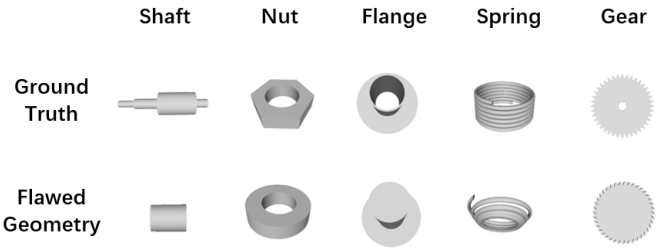

FIGURE 9. Examples of flawed geometry generated by the GPT-4V model

Figure 7 (a), there is a significant decrease in the average IoU upon inclusion of the debugger with a statistical analysis (P-value<0.01 obtained from an independent T-test). On the other hand, when examining the performance across specific component categories in Figure 7 (b), the P-values are 0.85, 0.17, 0.16, 0.06, and 0.09 for shafts, nuts, flanges, springs, and gears, respectively. While no significant difference is detected, the result indicates marginal significant differences (P-values<0.1) for springs and gears.

Figure 8 (a) provides a comprehensive summary of the GPT-4V model's overall performance across distinct input modalities. A one-way ANOVA is applied to the GPT-4V results, revealing a statistically significant difference (P-value < 0.01) between the different input modes. Subsequent pairwise comparisons were conducted using Tukey's Honestly Significant Difference test to pinpoint the specific modalities that exhibit significant discrepancies. The analyses indicate that the text-only input mode achieved higher IoU values compared to the other three input modes. This trend persists after integrating a debugger into the GPT-4V model, further solidifying the text-only mode's superior performance. Furthermore, when evaluating the impact of the debugger for each input mode, significant differences are observed between GPT-4V and GPT-4V + Debugger (all P-values < 0.05 using an independent T-test), suggesting that the introduction of the debugger may inadvertently affect the precision of generated 3D design objects. Figure 8 (b) shows that the most significant adverse effect caused by the debugger seems to be in the category of gears. Despite the clear impact observed, the statistical

analysis did not identify significant differences, primarily due to the limited number of resultant shapes. This issue arises from the low parsing rate associated with gears, limiting the power of statistical analysis.

A comparative analysis between the GPT-4 and GPT-4V models focusing on text-only inputs indicates the GPT-4V model exhibits a significantly higher IoU score compared to GPT-4 (P-value$<$0.01). This trend persists even when a debugger is incorporated (i.e., GPT-4 + Debugger VS. GPT-4V + Debugger).

4.2.3 QUALITATIVE RESULTS Figure 9 presents a sample of the flawed geometries generated by the GPT-4V model within five distinct component categories compared to the GT shapes. For shaft components, the issue is the exclusion of multiple shaft sections. In the context of nuts, the prevalent error consists of producing a circular nut instead of the specified hexagonal configuration. This issue could stem from the GPT models' limitations in generating CAD programs that require a sequence of precise operations. For instance, forming a hexagon needs six distinct steps involving the "Line" operation, with a specific angle between each segment. This process demands a high degree of accuracy and an understanding of geometric principles that may be difficult for GPT models to replicate. In contrast, GPT models may find it much easier to utilize a "Circle" operation to create the base shape so they "slept on the job." For flanges, the geometric fault is the omission of the flange's borehole. In the case of springs, the error commonly observed is the improper formation of the helix. Similarly, gears exhibit an issue similar to that of the flange components, characterized by the loss of the borehole.

The results highlight the models' current limitations in handling tasks that require detailed procedural generation and a deep understanding of spatial relationships. They might intentionally return incorrect CAD programs due to difficulty returning the correct ones. Improving their capability to accurately execute complex sequences of operations such as those needed for detailed CAD modeling remains an area for further development.

5 DISCUSSION

Based on the observations of the results, we extend our discussion to three aspects: 1) the effects of multimodal input for GPT-4V; 2) GPT-4 VS. GPT-4V; and 3) the effects of the debugger in 3D CAD generation. Furthermore, we acknowledge the limitations of our study and propose potential avenues for future research.

5.1 EFFECTS OF MULTIMODAL INPUT FOR GPT-4V

5.1.1 EFFECTS ON IOU For the IoU outcomes of GPT-4V shown in Figure 8, the statistical analyses reveal that the text-only input mode outperforms the other three modes with input modalities of text+sketch, text+image, and text+image+sketch. This trend was also observed for the integration of the debugging process, further underscoring the superior efficacy of the text-only input mode. This observation challenges our assumption on multimodal machine learning (MMML) that integrating various input modalities enhances the predictive capabilities of the ML models. The possible explanations for this result may be based on the following three aspects.

TABLE 3. Summary of the parsing rate and IoU for GPT-4 and GPT-4V models using the text-only input

	GPT-4 VS. GPT-4V (Average across five categories)	
	Parsing rate	IoU
Inherent capability	0.50 VS. **0.54**	(0.59$\pm$0.35) VS. (**0.68$\pm$0.32**)
Enhanced capability with the debugger	0.71 VS. 0.71	(0.53$\pm$0.37) VS. (**0.60$\pm$0.36**)

First, the simplicity of text-only data might help reduce computational burden and noise, leading to more efficient processing and accurate results. On the one hand, this implies that, under certain conditions, the advantage of multimodal might be negated by the associated data complexity. On the other hand, it implies that textual descriptions, especially those that include dimensional information, can provide substantial and adequate information for the GPT-4V model to "comprehend" the design requirements of mechanical components. Second, the hypothesis that integrating various input modalities could improve the predictive performance of machine learning (ML) models may be contingent upon how relevant the information from these modalities is to the design target, how precise it is, and how well the model can interpret the data. In our study, it is possible that the GPT-4V model mistakenly processed information from images or sketches. In fact, we undertook qualitative analyses, examining 10 images of each mechanical component through the GPT-4V API. These experiments revealed occasional misinterpretations, such as recognizing a three-section shaft as a two-section shaft. Third, it appears that the GPT-4V model is naturally more proficient at processing textual data than image data. This is particularly true in tasks that require precise spatial localization and perspective relationships. For example, GPT-4V often generates a tapered spring due to the effect of perspective in the rendered image. This suggests a possible limitation in the model's ability to accurately interpret renderings of mechanical components compared to its success with more commonly represented objects such as humans or vehicles [35].

5.1.2 EFFECTS ON PARSING RATE

Similar to the IoU outcomes, the text-only input mode surpasses the three multimodal input modes in terms of average parsing rate. However, this trend diverges when examining results specific to different categories of mechanical components. For more complex mechanical geometries (e.g., springs and gears), multimodal inputs demonstrate an advantage, either matching or exceeding the performance of text-only input. For instance, the text+image input is the best for springs, but text+sketch+image turns out to be the most effective input for gears. This emphasizes the value of incorporating visual information alongside textual data in improving the model's efficiency in parsing complex geometries. With the addition of the debugger to the GPT-4V model, the input of text with image achieves the best result in parsing gears. In contrast, for the remaining four components, the efficacy of multimodal inputs aligns closely with that of text-only input. This observation indicates the debugger's potential to amplify the model's proficiency in utilizing visual data, particularly in handling complex geometries.

5.2 GPT-4 VS. GPT-4V

The results of our experiments revealed the advantage of the GPT-4V model over the GPT-4 model in processing text input, both for the inherent and enhanced (i.e., with the debugger) versions, in the generation of 3D CAD models. This superiority is evidenced in terms of a higher parsing rate and IoU scores, as summarized in Table 3. According to the GPT-4V system card, OpenAI's official evaluation report of GPT-4V [3], GPT-4V is built upon the GPT-4 architecture as quoted here: "As GPT-4 is the technology behind the visual capabilities of GPT-4V, its training process was the same." Furthermore, the API models for both GPT-4 and GPT-4V utilized in our study share an identical knowledge cutoff date of April 2023. Given that GPT-4V is designed to accommodate visual inputs alongside textual data, its performance in processing text is anticipated to be comparable to that of GPT-4. Nonetheless, the reason for the observed performance differences is unclear at this stage. In addition, the absence of comparative studies in the literature specifically addressing the text-processing capabilities of GPT-4 and GPT-4V underscores further research's need to clarify these differences.

5.3 EFFECTS OF THE DEBUGGER

Figure 6 demonstrates that the integration of the debugger enhances the parsing rates for both GPT-4 and GPT-4V models, underscoring the debugger's efficacy in iteratively correcting syntax errors within the generated CAD program codes.

However, this enhancement in parsing rate comes at the cost of the reduction in IoU values for both models, as detailed in Figures 7 and 8. This decrease suggests that the GPT models used with a debugger may prioritize the correction of syntax errors and compromise on accurately fulfilling the design requirements. This hypothesis is supported by our qualitative experiments with the ChatGPT Pro version which is built on the GPT-4 model. We observed instances where, in the process of debugging syntax errors, ChatGPT prioritized correcting the CAD program code over sticking to the design requirements despite having access to the entire conversation history. This resulted in an oversimplification of the code, which may ultimately lead to incorrect geometry (e.g., generating a round nut for the required hexagonal nut). Addressing this limitation requires future research to improve the debugger's functionality to balance between syntax correction and simplification.

While the debugger presents a viable strategy for GPT model enhancement, alternative approaches, including model fine-tuning and the incorporation of function calls, could be potential ways to advance the application of GPT models in 3D CAD generation. Model fine-tuning is to adjust a GPT model by further training it on a specialized dataset, such as the multimodal CAD dataset proposed in this study, to enhance its ability to perform 3D CAD generation. In addition, function calls involve generating output by calling existing functions (e.g., the Python class for creating a shaft) to create 3D shapes.

5.4 LIMITATIONS AND FUTURE WORK

In this study, we evaluated five representative categories of mechanical components with different geometric complexities. Although the insights gained from the current synthesized dataset are valuable, we acknowledge that the sample size is relatively small compared to the wide array of mechanical components. To obtain an in-depth understanding of the role that multimodal LLMs play in the generation of 3D CAD models, an expansion of the CAD dataset is essential. It is also critical to note that designs often consist of interconnected components in the form of assemblies rather than individual components [39, 40]. This requires improvements in the current data synthesis pipeline, specifically the inclusion of additional categories of CAD models and the capability to synthesize system design objects. Lastly, the examination of additional CAD programming languages, such as OpenSCAD and Fusion 360 Python API, as well as other LLMs, such as Google's Gemini, will help gain more comprehensive views on the capability of multimodal LLMs in CAD generation of 3D shapes.

6 CONCLUSION

This study is motivated by answering two research questions: *1) To what extent can multimodal LLMs generate 3D design objects when employing different design modalities or a combination of various modalities? 2) What strategies can be developed to enhance the ability of multimodal LLMs to create 3D design objects?* Therefore, we first developed an approach to enable multimodal LLMs in 3D CAD generation. Then, we

studied the performance of the GPT-4 and GPT-4V models with different input modalities, including the text-only, text+sketch, text+image, and text+sketch+image data.

Both GPT-4 and GPT-4V showed significant potential in the generation of 3D CAD models, especially with the enhancements enabled by a debugging process. Additionally, in our GPT-4V experiment, we tested four input modes: text-only, text with sketch, text with image, and a combination of text, sketch, and image. Surprisingly, GPT-4V's performance with text-only input surpassed that of the other three multimodal inputs on average. This observation challenges the common belief in MMML that incorporating varied input modalities always improves a machine learning model's predictive accuracy due to increased information for learning and inference. However, when examining category-specific results of mechanical components, while the same trend is still observed for geometric accuracy, multimodal inputs start to gain prominence with more complex geometries (e.g., spring and gears) in the successful parsing rate of the generated CAD programs.

From these observations, we see that the current multimodal LLMs are still limited in handling multimodal inputs when applied to LLM4CAD. However, the insights from the category-specific results indicate that multimodal LLMs have potential benefits in real-world design scenarios characterized by complex objects, although it remains challenging for them to generate complex design objects. Improving the capability of these models to process diverse input modalities and proposing strategies to improve their capability to handle complex design objects are promising research avenues.

To further address the two RQs posed and achieve a comprehensive understanding, future studies should broaden the research scope to include a more diverse dataset featuring more complex 3D design objects. Strategies, including model fine-tuning and the integration of function calls, to enhance the utility of multimodal LLMs for CAD are worthy to explore. Moreover, while this study focused on the CAD generation of 3D shapes during the conceptual design phase, future research can explore other stages of the engineering design process, such as customer needs analysis, design evaluation, and manufacturing. This will contribute to a deeper understanding of how LLMs, particularly multimodal LLMs, can be employed to facilitate the overall engineering design process, thus making contributions to advanced design methodologies for human-centered generative design [41].

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support from the National Science Foundation through Award 2207408.

REFERENCES

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. "Language models are few-shot learners". *Advances in neural information processing systems,* ***33***, pp. 1877–1901.

[2] Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al., 2023. "Chatgpt for good? on opportunities and challenges of large language models for education". *Learning and individual differences,* ***103***, p. 102274.

[3] OpenAI, 2023. "Gpt-4v(ision) system card".

[4] Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al., 2023. "Palm-e: An embodied multimodal language model". *arXiv preprint arXiv:2303.03378*.

[5] Kocaballi, A. B., 2023. "Conversational ai-powered design: Chatgpt as designer, user, and product". *arXiv preprint arXiv:2302.07406*.

[6] Filippi, S., 2023. "Measuring the impact of chatgpt on fostering concept generation in innovative product design". *Electronics,* ***12***(16), p. 3535.

[7] Ma, K., Grandi, D., McComb, C., and Goucher-Lambert, K., 2023. "Conceptual design generation using large language models". *arXiv preprint arXiv:2306.01779*.

[8] Li, X., Wang, Y., and Sha, Z., 2023. "Deep learning methods of cross-modal tasks for conceptual design of product shapes: A review". *Journal of Mechanical Design,* ***145***(4), p. 041401.

[9] Li, X., Wang, Y., and Sha, Z., 2022. "Deep learning of cross-modal tasks for conceptual design of engineered products: A review". In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 86267, American Society of Mechanical Engineers, p. V006T06A016.

[10] Song, B., Zhou, R., and Ahmed, F., 2023. "Multi-modal machine learning in engineering design: A review and future directions". *arXiv preprint arXiv:2302.10909*.

[11] Li, X., Xie, C., and Sha, Z., 2022. "A predictive and generative design approach for three-dimensional mesh shapes using target-embedding variational autoencoder". *Journal of Mechanical Design,* ***144***(11), p. 114501.

[12] Makatura, L., Foshey, M., Wang, B., HähnLein, F., Ma, P., Deng, B., Tjandrasuwita, M., Spielberg, A., Owens, C. E., Chen, P. Y., et al., 2023. "How can large language models help humans in design and manufacturing?". *arXiv preprint arXiv:2307.14377*.

[13] Picard, C., Edwards, K. M., Doris, A. C., Man, B., Giannone, G., Alam, M. F., and Ahmed, F., 2023. "From concept to manufacturing: Evaluating vision-language models for engineering design". *arXiv preprint arXiv:2311.12668*.

[14] Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., and Chen, M., 2022. "Point-e: A system for generating 3d point clouds from complex prompts". *arXiv preprint arXiv:2212.08751*.

[15] Jun, H., and Nichol, A., 2023. "Shap-e: Generating conditional 3d implicit functions". *arXiv preprint arXiv:2305.02463*.

[16] Nelson, M. D., Goenner, B. L., and Gale, B. K., 2023. "Utilizing chatgpt to assist cad design for microfluidic devices". *Lab on a Chip,* ***23***(17), pp. 3778–3784.

[17] Baltrušaitis, T., Ahuja, C., and Morency, L.-P., 2018. "Multimodal machine learning: A survey and taxonomy". *IEEE transactions on pattern analysis and machine intelligence,* ***41***(2), pp. 423–443.

[18] Song, B., Miller, S., and Ahmed, F., 2023. "Attention-enhanced multimodal learning for conceptual design evaluations". *Journal of Mechanical Design,* ***145***(4), p. 041410.

[19] Su, H., Song, B., and Ahmed, F., 2023. "Multimodal machine learning for vehicle rating predictions using image, text, and parametric data". *arXiv preprint arXiv:2305.15218*.

[20] Chowdhary, K., and Chowdhary, K., 2020. "Natural language processing". *Fundamentals of artificial intelligence*, pp. 603–649.

[21] OpenAI Team. Introducing ChatGPT: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/, November 2022. Accessed: 2024-01-16.

[22] Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., and Tang, Y., 2023. "A brief overview of chatgpt: The history, status quo and potential future development". *IEEE/CAA Journal of Automatica Sinica,* ***10***(5), pp. 1122–1136.

[23] Ray, P. P., 2023. "Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope". *Internet of Things and Cyber-Physical Systems*.

[24] Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., and Tang, Y., 2023. "A brief overview of chatgpt: The history, status quo and potential future development". *IEEE/CAA Journal of Automatica Sinica,* ***10***, pp. 1122–1136.

[25] Haleem, A., Javaid, M., and Singh, R. P., 2022. "An era of chatgpt as a significant futuristic support tool: A study on features, abilities, and challenges". *BenchCouncil Transactions on Benchmarks, Standards and Evaluations,* ***2***, 10, p. 100089.

[26] Abdullah, M., Madain, A., and Jararweh, Y., 2022. "Chatgpt: Fundamentals, applications and social impacts". pp. 1–8.

[27] Wang, X., Anwer, N., Dai, Y., and Liu, A., 2023. "Chatgpt for design, manufacturing, and education". *Procedia CIRP,* ***119***, pp. 7–14.

[28] Gulwani, S., Polozov, O., Singh, R., et al., 2017. "Program synthesis". *Foundations and Trends® in Programming Languages,* ***4***(1-2), pp. 1–119.

[29] OpenAI, 2023. "Gpt-4 technical report". *arXiv preprint arXiv:2303.08774*.

[30] Kim, S., Chi, H.-g., Hu, X., Huang, Q., and Ramani, K., 2020. "A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks". In ECCV 16th European Conference, Glasgow, UK, August 23–28, 2020, pp. 175–191.

[31] Lee, H., Lee, J., Kim, H., and Mun, D., 2022. "Dataset and method for deep learning-based reconstruction of 3d cad models containing machining features for mechanical parts". *Journal of Computational Design and Engineering,* ***9***(1), pp. 114–127.

[32] Manda, B., Dhayarkar, S., Mitheran, S., Viekash, V., and Muthuganapathy, R., 2021. "'cadsketchnet'-an annotated sketch dataset for 3d cad model retrieval with deep neural networks". *Computers & Graphics,* ***99***, pp. 100–113.

[33] Summers, J. D., and Shah, J. J., 2010. "Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability". *Journal of Mechanical Design,* ***132***(2), 01, p. 021004.

[34] McKay, M. D., Beckman, R. J., and Conover, W. J., 2000. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code". *Technometrics,* ***42***(1), pp. 55–61.

[35] Luo, T., Rockwell, C., Lee, H., and Johnson, J., 2024. "Scalable 3d captioning with pretrained models". *Advances in Neural Information Processing Systems,* ***36***.

[36] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al., 2021. "Learning transferable visual models from natural language supervision". In International conference on machine learning, PMLR, pp. 8748–8763.

[37] Mason, W., and Suri, S., 2012. "Conducting behavioral research on amazon's mechanical turk". *Behavior research methods,* ***44***(1), pp. 1–23.

[38] Lopez, C. E., Miller, S. R., and Tucker, C. S., 2018. "Exploring biases between human and machine generated designs". *Journal of Mechanical Design,* ***141***. Amazon MTurk example.

[39] Li, X., Xie, C., and Sha, Z., 2023. "Design representation for performance evaluation of 3d shapes in structure-aware generative design". *Design Science,* ***9***, p. e27.

[40] Li, X., Xie, C., and Sha, Z., 2021. "Part-aware product design agent using deep generative network and local linear embedding". *Proceedings of the 54th Hawaii International Conference on System Sciences*.

[41] Demirel, H. O., Goldstein, M. H., Li, X., and Sha, Z., 2024. "Human-centered generative design framework: an early design framework to support concept creation and evaluation". *International Journal of Human–Computer Interaction,* ***40***(4), pp. 933–944.