ASME Journal of Computing and Information Science in Engineering
Online journal at:
https://asmedigitalcollection.asme.org/computingengineering

LLM4CAD: Multimodal Large Language Models for Three-Dimensional Computer-Aided Design Generation

Xingang Li
Walker Department of Mechanical Engineering,
University of Texas at Austin,
Austin, TX 78712
e-mail: xingang.li@utexas.edu

Yuewan Sun
Walker Department of Mechanical Engineering,
University of Texas at Austin,
Austin, TX 78712
e-mail: yuewansun@utexas.edu

Zhenghui Sha[1]
Walker Department of Mechanical Engineering,
University of Texas at Austin,
Austin, TX 78712
e-mail: zsha@austin.utexas.edu

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 LLMs to generate 3D CAD models (i.e., LLM4CAD) and perform experiments to evaluate their efficacy where GPT-4 and GPT-4V were employed as examples. 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 great potential in 3D CAD generation by just leveraging their zero-shot learning ability. 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, the prominence of multimodal inputs is increasingly evident for more complex geometries (e.g., springs and gears) in both Cap 1 and Cap 2. The potential of multimodal LLMs to improve 3D CAD generation is clear, but their application must be carefully calibrated to the complexity of the target CAD models to be generated.
[DOI: 10.1115/1.4067085]

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 developed. These models are designed to process more input modalities besides text, such as images, thereby broadening the way 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 design [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 the 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]. These visual elements are crucial for describing the intricacies of design that textual descriptions alone cannot capture. Therefore, recent design research emphasizes the significance of multimodal machine learning (MMML) in improving the conceptual design by integrating diverse modalities [8–11]. Multimodal input, such as images and

[1]Corresponding author.
Manuscript received April 22, 2024; final manuscript received October 25, 2024; published online December 12, 2024. Assoc. Editor: Yaoyao Fiona Zhao.

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 great potential for their application in AI-assisted conceptual design, promising to revolutionize design tools and human-AI collaborative design.

Specifically, computer-aided design (CAD) has become an indispensable tool in conceptual design, enabling the transformation of design ideas or design requirements into 3D models. Translating design requirements, whether textual or visual, into accurate 3D geometries is challenging, even for experienced human designers, and poses considerable challenges for LLMs. There are two primary challenges: (1) CAD involves the creation of complex geometric and design elements, where components are highly interdependent. Designers must carefully consider the placement, dimensions, and relationships between various parts of a design, ensuring that each element aligns correctly with others. (2) CAD tasks require robust visual–spatial reasoning, demanding a deep understanding of the relationships between different geometric components within a 3D space, as well as the ability to manipulate these relationships effectively. The "chain-of-thought" method [12], commonly used in LLMs to solve mathematical or reasoning problems, may not perform well in this context as it is more suited to tasks involving linear logical reasoning. Given these challenges, the question arises: to what extent can LLMs be applied to CAD generation? To our knowledge, 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. 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? and (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. The capability of LLMs to generate text and code presents an opportunity to apply these models to CAD programming [13]. However, the effectiveness of LLMs in acquiring and applying the specialized domain knowledge required for CAD packages is still uncertain. Additionally, there is limited research on enhancing the executability of the CAD programs generated by these models. Therefore, we developed an approach to enable an indirect synthesis of 3D design objects by generating CAD programs [14]. 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 (GT) 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),[2] 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 developed 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 GPT-4 and GPT-4V models have significant potential for LLM4CAD by just leveraging their zero-shot learning ability. Especially, the performance can be further enhanced by the proposed debugger. However, they still struggle with generating 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 particularly excels when processing purely text-only input, outperforming multimodal inputs in both 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, multimodal inputs start to gain prominence with more complex geometries (e.g., springs and gears) in both Cap 1 and Cap 2.

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 Sec. 2, we provide an overview of the background related to multimodal machine learning and LLMs for engineering design. Section 3 outlines the methodology for data collection and generation, as well as the evaluation of multimodal LLMs. Subsequently, Secs. 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 Sec. 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 and their applications for engineering design.

2.1 Engineering Design Using 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 multimodal 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 and image captioning) is becoming popular [15]. 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 [15].

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

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 [16,17]. For example, Song et al. [16] 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 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.

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 [18]. Building upon the foundation of various NLP technologies, the emergence of LLMs such as the GPT series [1] marks a significant leap in AI proficiency.

A remarkable example of LLMs is ChatGPT [19], 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 [20,21]. 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 [20,22,23].

LLMs can process not only natural language but also programming language [24]. Leveraging LLM's capability to synthesize programs, program-aided language (PaL) models [12] have shown remarkable success in solving complex mathematical problems and reasoning tasks. PaL models effectively utilize LLMs to convert natural language problems into executable programs, which are then processed by a runtime environment, such as a PYTHON interpreter, to obtain solutions. They leverage the strengths of both natural language processing and traditional programming, providing a robust framework for tackling intricate problems that require a combination of linguistic understanding and precise calculations.

Our study draws inspiration from PaL to address a design problem using program code, but it is different from PaL in two significant ways. First, while PaL has primarily focused on solving general mathematical and reasoning tasks using basic PYTHON programs, our work tackles the more complex challenge of generating CAD-related programming code (e.g., CADQuery). This requires not only an understanding of geometric and spatial relationships but also adherence to design constraints and the production of syntactically correct and efficient code for 3D modeling software'a task that extends beyond mathematical and general reasoning capabilities. In this study, we evaluate how LLMs can interpret design requirements expressed through various combinations of design modalities, such as textual descriptions and images, and generate corresponding CAD programming code for 3D designs. Second, while PaL generally falls under the category of few-shot prompt engineering research—providing a natural language problem and generating programs as intermediate reasoning steps in their prompts, followed by a new task—our study is rooted in zero-shot prompt research. In this approach, no examples are provided in the prompts. This decision is motivated by two primary considerations: (1) it is impractical for users to provide a CAD program example, as creating such examples is both challenging and time-consuming, and (2) if a CAD program example from the same category of mechanical components is provided, the benefits of utilizing LLMs would be diminished, as simply substituting parameters could yield the correct CAD program for the new task.

We contribute to the existing literature on PaL models and demonstrate their potential to be applied to a broader range of cross-modal tasks beyond traditional mathematical and logical reasoning problems with a zero-shot prompt strategy.

2.3 Large Language Models for Engineering Design.

Researchers have been examining how LLMs, such as ChatGPT, can be applied to enhance the engineering design process, from conceptual design to manufacturing [5,6,14,25–30]. 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 PaL models that exploit LLMs' capacity for program synthesis [24] 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 [14,26]. 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) [3,21,31] 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 Fig. 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 GT 3D CAD models, and dimensional data are recorded alongside the generation process. Direct rendering techniques can be employed to obtain images from the 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

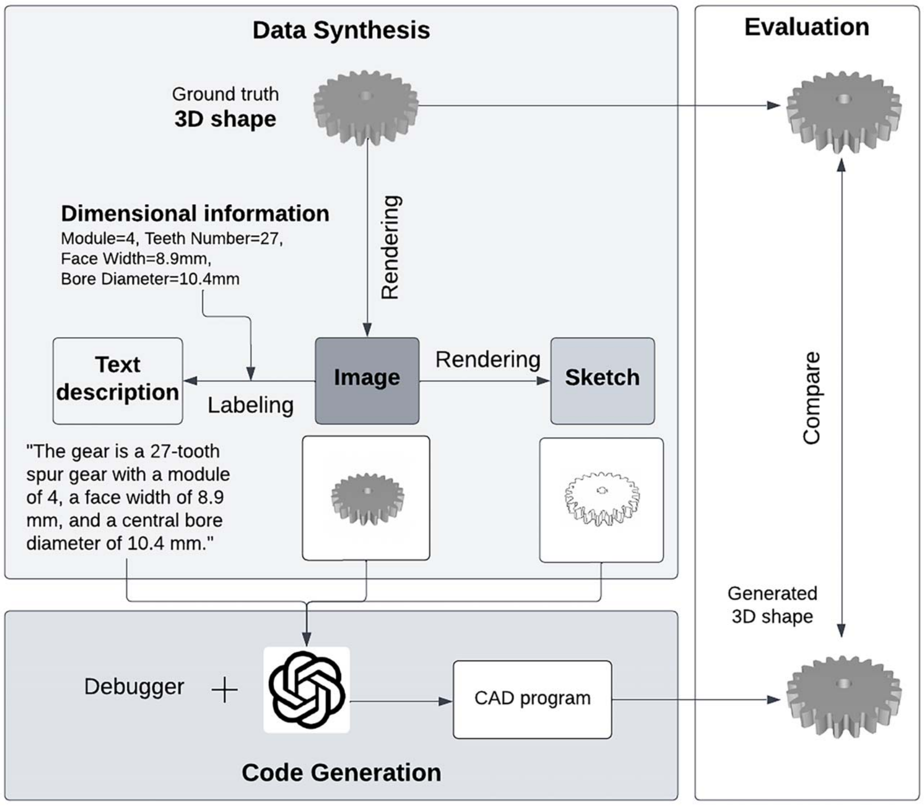
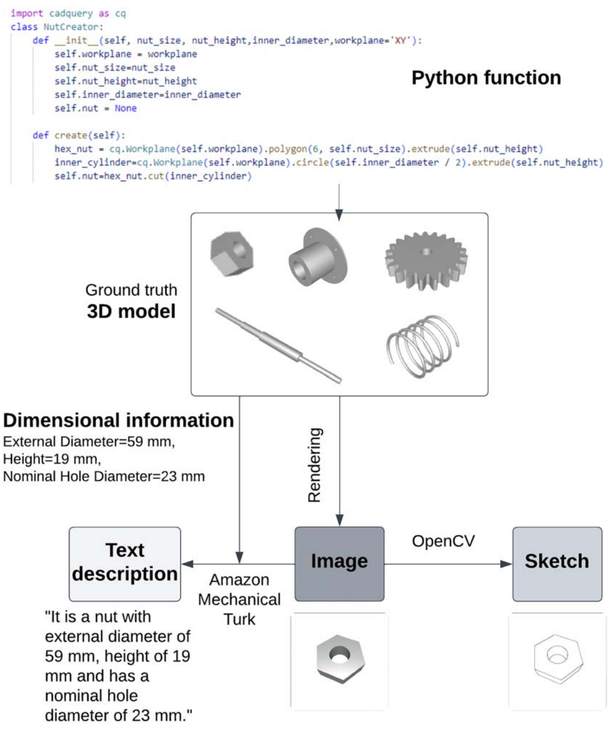


Fig. 1 The overview of our approach

parsed into 3D shapes. The quality of the resulting 3D shapes is then benchmarked against the GT shapes to gauge the efficacy of generation. In addition, we proposed a debugger to enhance the models' capabilities in CAD program generation.

3.1 Data Synthesis. We choose mechanical components as 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 [32–34] 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 Fig. 2, a semi-automated pipeline for data 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. A component with more complex geometry (i.e., requiring more parameters and calculations to determine its shape) is theoretically more challenging to create its CAD model due to the increased number of CAD operations and computations needs. Generating CAD programs for components such as shafts, nuts, and flanges, which primarily utilize a small number of CAD operations, such as *Sketch* and *Extrude*, is relatively simple. However, the creation of CAD programs for springs and gears presents more challenges. For example, gears require complex calculations to determine the profiles of gear teeth, while springs have spiral shapes and need less-frequently used CAD operations such as *Evolve*. To show this, we performed an analysis of the execution time in generating each of the five mechanical components on computers using the CADQuery application programming interface



```
import cadquery as cq
class NutCreator:
    def __init__(self, nut_size, nut_height,inner_diameter,workplane='XY'):
        self.workplane = workplane
        self.nut_size=nut_size
        self.nut_height=nut_height
        self.inner_diameter=inner_diameter
        self.nut = None

    def create(self):
        hex_nut = cq.Workplane(self.workplane).polygon(6, self.nut_size).extrude(self.nut_height)
        inner_cylinder=cq.Workplane(self.workplane).circle(self.inner_diameter / 2).extrude(self.nut_height)
        self.nut=hex_nut.cut(inner_cylinder)
```

Fig. 2 The pipeline for data synthesis

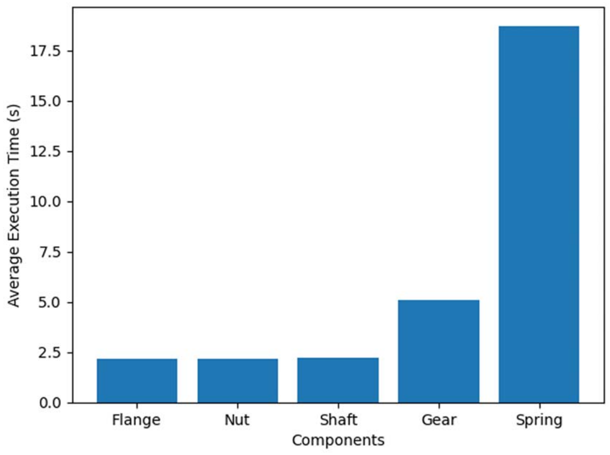


Fig. 3 Execution time analysis of mechanical components

(API), as illustrated in Fig. 3. On average, gears and springs require more execution time than flanges, nuts, and shafts. Therefore, in this paper, we refer to shafts, nuts, and flanges as simple geometries, while springs and gears are complex geometries.

3.1.1 Three-Dimensional 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. CADQuery was selected due to its ease of use, flexibility, and ability to generate parametric CAD models through a script-based approach. CADQuery allows for the creation of complex geometries with relatively simple code, making it well-suited for integration with LLMs such as GPT-4 and GPT-4V. Additionally, CADQuery's open-source nature and active community support ensure that LLMs can more easily understand CADQuery, enabling them to provide appropriate responses based on the design requirements. As introduced in Sec. 3.2, CADQuery is also utilized for generating the LLMs' responses. 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 Fig. 2. To achieve uniform sampling of the design space, we employed Latin hypercube sampling [35] of design parameters (such as the external diameter and height of a nut). 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 1000. For the other four components, 1000 shapes for each are created. The dimensional information of these shapes was recorded alongside the GT 3D 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 with the camera fixed at the position (20, 20, 20) and oriented toward the origin (0, 0, 0). Subsequently, sketches of these images were produced by sketch-style rendering using OpenCV. Only a single view of the image and sketch was generated, and no dimensional information was provided on the images or sketches. 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 [34]. With the consideration of such trade-offs, we decided to use computer renderings for the sketch data.

We decided not to include or annotate dimensional information in 2D images or sketches, considering real-world applications. The scenario in which an LLM4CAD model is most likely to be utilized is during reverse engineering, where detailed engineering features and dimensions are typically limited in images or drawings. For instance, this might occur when an individual takes a photo of an object they wish to convert into a 3D model. Consequently, in our research design, we intentionally chose to annotate dimensional information solely within text data.

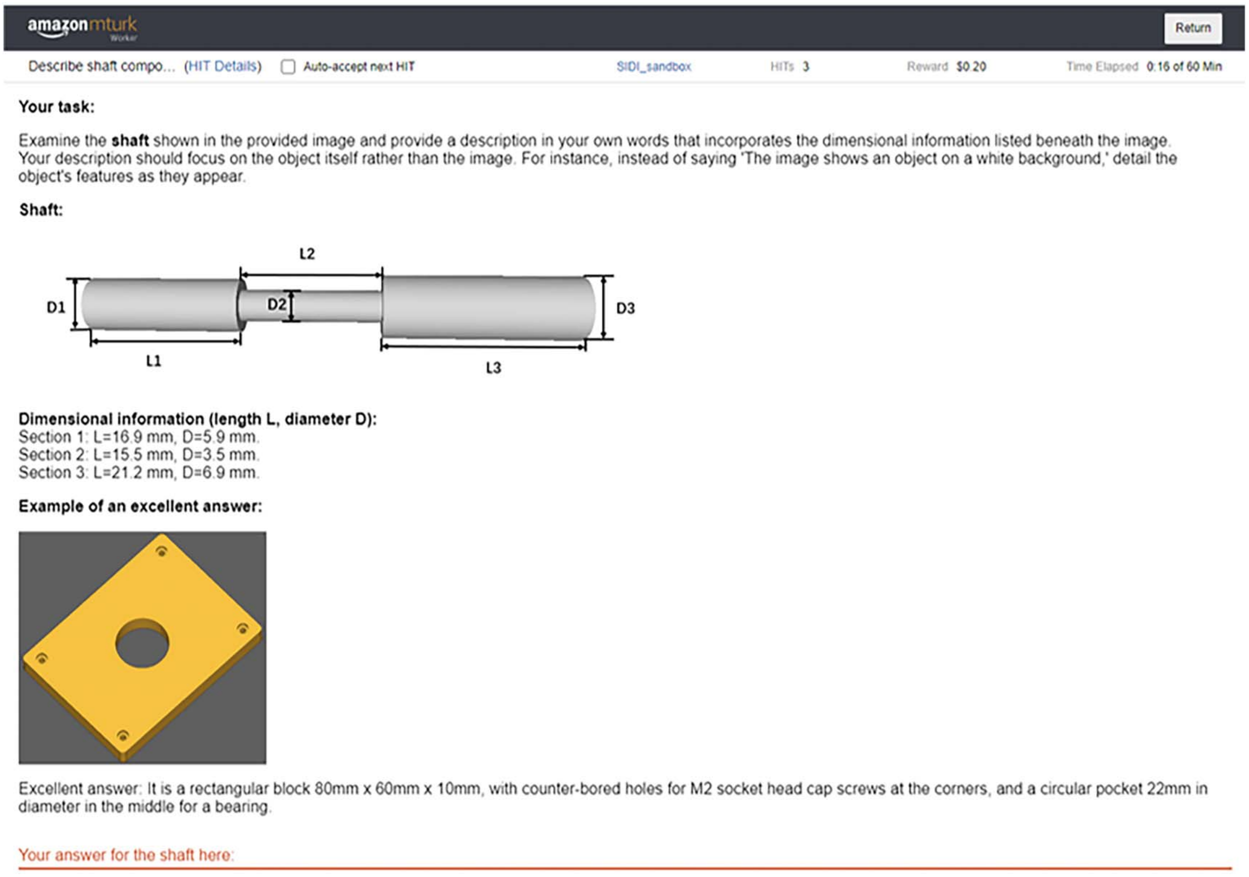
3.1.2 Textual Descriptions With Dimensional Information. It is feasible to synthesize textual data integrated with dimensional information from images via GPT models [36]. However, as we need to input this textual data into GPT models for analysis, it might introduce a risk of biasing the results. To that end, we tested the other popular automatic captioning methods, such as the contrastive language-image pretraining model [37], but found the results unsatisfactory for mechanical components.

To mitigate this and ensure the quality of the textual data, we chose to crowdsource textual descriptions through 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 [38,39]. We designed human intelligence tasks (HITs) on AMT to recruit participants 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 the incorporation of 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, corresponding to the four types of shafts and the other four mechanical components as aforementioned. We published 1000 assignments for each category of the mechanical components (250 for each of the four HITs of shafts (250×4) and 1000 for each of the other four 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 Fig. 4. 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 to 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 67% of the responses were deemed to be of high quality. The final dataset included a collection of textual descriptions: 628 for shafts, 671 entries for nuts, 692 for flanges, 679 for springs, and 661 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. Additional details on the dataset can be found in Table 5 in the Appendix.

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


amazon mturk Worker

Return

Describe shaft compo... (HIT Details) Auto-accept next HIT SIDI_sandbox HITs 3 Reward $0.20 Time Elapsed 0:16 of 60 Min

Your task:

Examine the **shaft** shown in the provided image and provide a description in your own words that incorporates the dimensional information listed beneath the image. Your description should focus on the object itself rather than the image. For instance, instead of saying 'The image shows an object on a white background,' detail the object's features as they appear.

Shaft:

Dimensional information (length L, diameter D):
Section 1: L=16.9 mm, D=5.9 mm.
Section 2: L=15.5 mm, D=3.5 mm.
Section 3: L=21.2 mm, D=6.9 mm.

Example of an excellent answer:

Excellent answer: It is a rectangular block 80mm x 60mm x 10mm, with counter-bored holes for M2 socket head cap screws at the corners, and a circular pocket 22mm in diameter in the middle for a bearing.

Your answer for the shaft here:

Fig. 4 An example of the HITs on Amazon Mechanical Turk

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

	Published tasks	Filtered responses	Examples of textual descriptions collected from AMT
Shafts	1000	628	It is a shaft which has four sections. In the first, it has 14.9 mm length and 15.5 mm diameter. In the second, it has 12.8 mm length and 21.3 mm diameter. In the third, it has 20.3 mm length and 21.1 mm diameter. In the fourth, it has 25.6 mm length and 2.6 mm diameter.
Nuts	1000	671	It is a hexagon nut with an external diameter of 47 mm, a nominal hole diameter of 7 mm, and a height of 14 mm
Flanges	1000	692	It is a flange with a 124 mm diameter and 14 mm thickness, with a raised face of 86 mm diameter, a 16 mm bore diameter, and a 144 mm face height
Springs	1000	679	The spring is a coil with a diameter of 8 mm and a pitch of 14 mm. It is 46 mm long when uncompressed, made of wire with a 1.5 mm radius. It seems strong and flexible, suitable for many uses.
Gears	1000	661	It is a gear with a module of 6 and 44 teeth, a face width of 8.7 mm, and a bore diameter of 19.3 mm

3.2 Code Generation and Debugger. We show the pipeline of the code generation, and evaluation processes in Fig. 5. The experiment was conducted by utilizing the models' API and instructing them to generate CAD program code via CADQuery. 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 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 Fig. 6 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 zero-shot learning ability 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

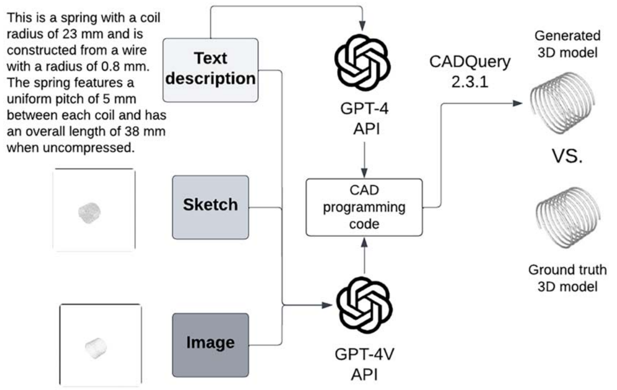

This is a spring with a coil radius of 23 mm and is constructed from a wire with a radius of 0.8 mm. The spring features a uniform pitch of 5 mm between each coil and has an overall length of 38 mm when uncompressed.
Text description
Sketch
Image
GPT-4 API
GPT-4V API
CAD programming code
CADQuery 2.3.1
Generated 3D model
VS.
Ground truth 3D model

Fig. 5 The pipeline for code generation and evaluation

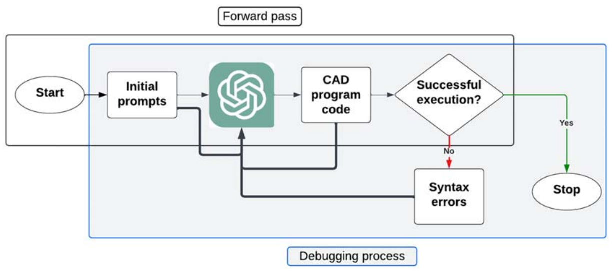


Fig. 6 The process of generating CAD program code using GPT models with a debugger that can iteratively correct the syntax errors (if any) of the CAD program code. The bolded lines indicate three different sources of the input information in the debugging process.

an iterative procedure dedicated to the identification and correction of errors in the code. The previous conversation content (including the user requirements and GPT's responses) and the current error messages as indicated by the three bolded lines in Fig. 6 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 pre-step 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.

4 Experiments and Results

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

4.1 Experimental 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 down to input modes that include textual descriptions, ensuring the necessary dimensional data are available for accurate analysis and comparison. In both scenarios, we first assessed the 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 balancing computational efficiency and model performance based on our observations from the debugging experiments: (1) The debugger, while effective, significantly increases response time and consumes more resources with each iteration. Each iteration involves feeding all historical data into the API, making the process more time-consuming and resource-intensive beyond three iterations. (2) After three iterations, there is no significant improvement in the IoU scores. The debugging process primarily addresses syntax errors rather than iteratively refining the CAD program, and additional iterations do not enhance the IoU meaningfully. (3) The parsing rate sees minimal improvement after the third iteration. For simple components, three iterations are sufficient to achieve near-optimal (i.e., about 98%) parsing rates, while additional iterations do not significantly improve parsing rates for more complex components.

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 GT shapes.

4.2.1 Results of the Parsing Rate. Figure 7 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 7(*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 Fig. 7(*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 gears, and the input using the combination of text, sketch, and image achieves the best in springs.

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

Table 2 Details of the experiment settings

Mechanical components	API	Input modality	Model	Metrics
3D shapes	gpt-4-1106-preview	Text	GPT-4 model	
• Shafts (four types)			GPT-4 model + debugger[a]	
• Nuts				Parsing rate and IoU
• Flanges	gpt-4-1106-vision-preview	Text	GPT-4V model	
• Springs		Text + sketch		
• Gears		Text + image		
			GPT-4V model + debugger[a]	
		Text + sketch + image		

[a]Debugging times = 3.

The result of the comparison between Figs. 7(*b*) and 7(*c*) mirrors the trend observed in the GPT-4 model, demonstrating an improved parsing rate with the introduction of a debugger. Moreover, the parsing rate values of the four input modes for each category of the five mechanical components achieve similar levels. As a result, the difference in the average parsing rate among the four input modes is reduced and these values are approaching a similar level of around 0.7, as shown in the annotation text of Fig. 7(*c*). The pattern in which text-only input is dominant over the other three input modes for simple geometries, as observed in Fig. 7(*b*), no longer holds. Multimodal input modes become more effective for both simple and complex geometries by introducing the debugger.

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 higher average parsing rate (0.525) compared to its GPT-4 counterpart (0.517). However, this advantage diminishes upon the integration of a debugging process (0.711 for GPT-4 versus 0.710 for GPT-4V).

4.2.2 *Results of the Intersection Over Union.* Figure 8 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 Fig. 8(*a*), there is a significant decrease in the average IoU upon inclusion of the debugger with statistical analysis (*P*-value = 0.013 obtained from an independent *T*-test). On the other hand, when examining the performance across specific component categories in Fig. 8(*b*), the *P*-values

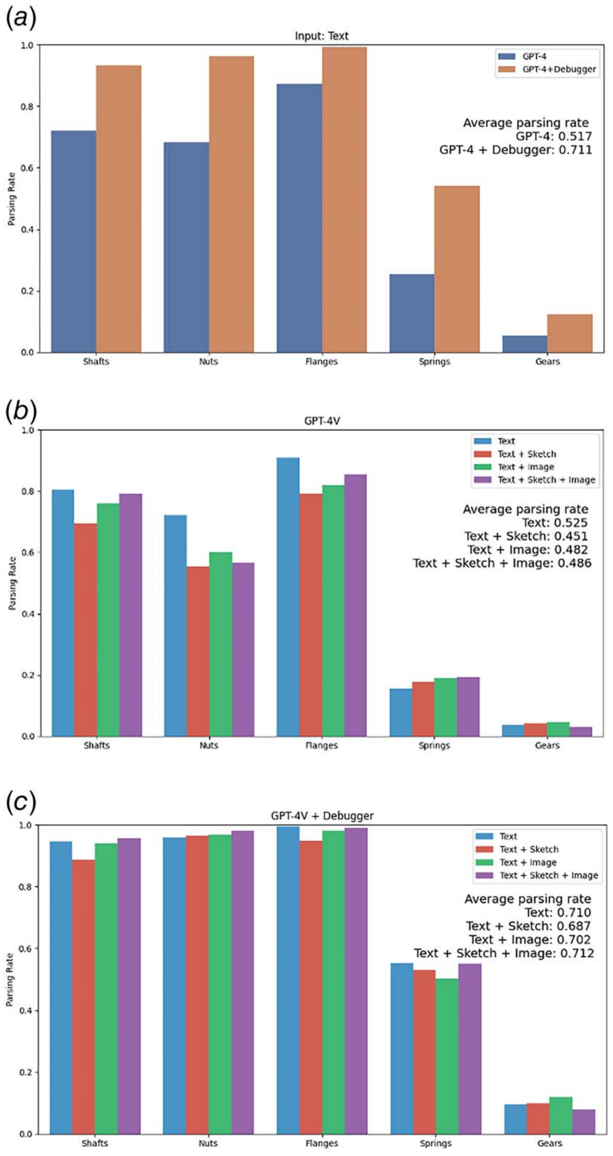


Fig. 7 Results of the parsing rate for the five categories of shafts, nuts, flanges, springs, or gears: (*a*) GPT-4 versus GPT-4 + debugger with text-only input, (*b*) GPT-4V model, and (*c*) GPT-4V + debugger

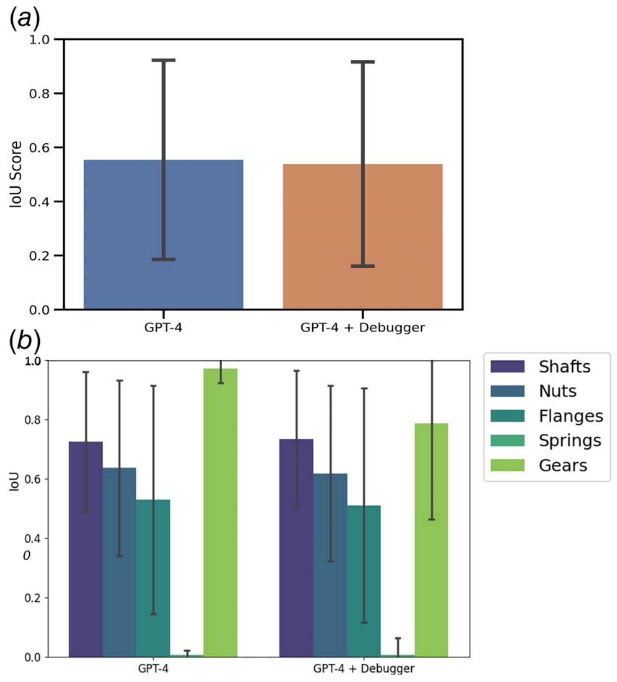


Fig. 8 Results of IoU for (*a*) overall performance and (*b*) category-specific performance of the GPT-4 model

Table 3 Summary of the *P*-values of statistical analysis for the IoU values: (a) *T*-test for the effects of including the debugger for either GPT-4 or GPT-4V, (b) one-way ANOVA for the effects of different input modes when using GPT-4V with or without the debugger

		Category					Overall
	Input mode	Shaft	Nut	Flange	Spring	Gear	
(a) With or without debugger							
GPT-4	Text-only	0.511	0.321	0.378	0.951	**<0.01**	**0.013**
GPT-4V	Text-only	0.714	0.464	0.181	0.597	0.285	**<0.01**
	Text + sketch	0.193	0.343	0.148	0.362	0.285	**<0.01**
	Text + image	0.414	0.330	**<0.01**	0.956	0.202	**<0.01**
	Text + sketch + image	0.369	0.251	0.295	0.662	0.066	**<0.01**
		Category					Overall
		Shaft	Nut	Flange	Spring	Gear	
(b) Comparison of different input modes							
GPT-4V		**0.019**	**<0.01**	0.157	0.767	0.662	**<0.01**
GPT-4V + debugger		0.464	**<0.01**	**<0.01**	0.382	0.099	**<0.01**

Note: *P*-values that meet the significance threshold of 0.05 are highlighted in bold.

are 0.511, 0.321, 0.378, 0.951, and <0.01 for shafts, nuts, flanges, springs, and gears, respectively, as summarized in Table 3. So, a significant difference is only detected for gears but there is no significant difference for the other four categories.

Figure 9(*a*) provides a comprehensive summary of the GPT-4V model's overall performance across distinct input modalities. As summarized in Table 3(*a*), 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). This result suggests that the introduction of the debugger may inadvertently affect the precision of generated 3D design objects. However, in general, there is no significant difference when examining the category-specific results, except for the following two cases: (1) gears for GPT-4 and (2) flanges for GPT-4V when using the text + image input mode.

Additionally, as shown in Table 3(*b*), a one-way analysis of variance (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 differences. 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. Nonetheless, when we look at each category of the mechanical components as shown in Fig. 9(*b*), the text-only input achieves significantly higher IoU values only for certain simple geometries (e.g., nuts for both GPT-4V and GPT-4V + debugger, shafts for GPT-4V, and flanges for GPT-4V + debugger). For more complex geometries (i.e., springs and gears), the power of multimodal input becomes more prominent compared to text-only input.

Furthermore, a comparative analysis between the GPT-4 and GPT-4V models focusing on the text-only input mode indicates that 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 versus GPT-4V + debugger).

4.2.3 Qualitative Results. Figure 10 illustrates an example of a response generated by LLMs given the design requirement for a flange, comprising three main sections: introduction, CAD program code, and summary. In particular, the CAD program code involves importing CadQuery as the CAD package, defining parameters, calling functions to create a 3D model, and exporting the final model as an STL file. Generating such a response that meets the design requirements needs not only an understanding of geometric and spatial relationships but also adherence to design constraints and the ability to produce syntactically correct codes for 3D modeling software—skills that go beyond mathematical and general reasoning capabilities as investigated in program-aided LLMs [12].

Figure 11 presents 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 the difficulty in 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 versus 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 Intersection Over Union. For the IoU outcomes of GPT-4V shown in Fig. 9, the statistical analyses on the overall performance in Table 3 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,

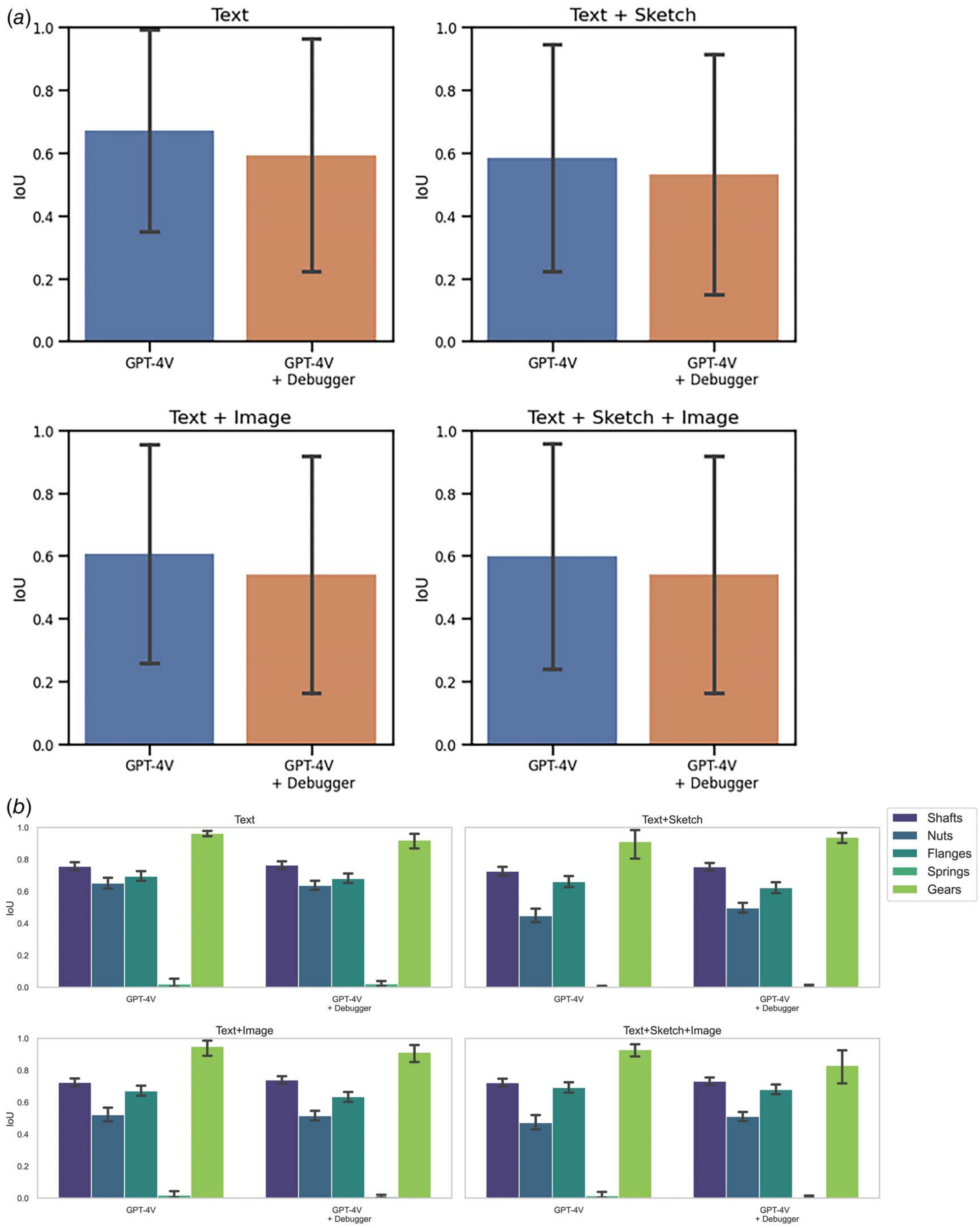
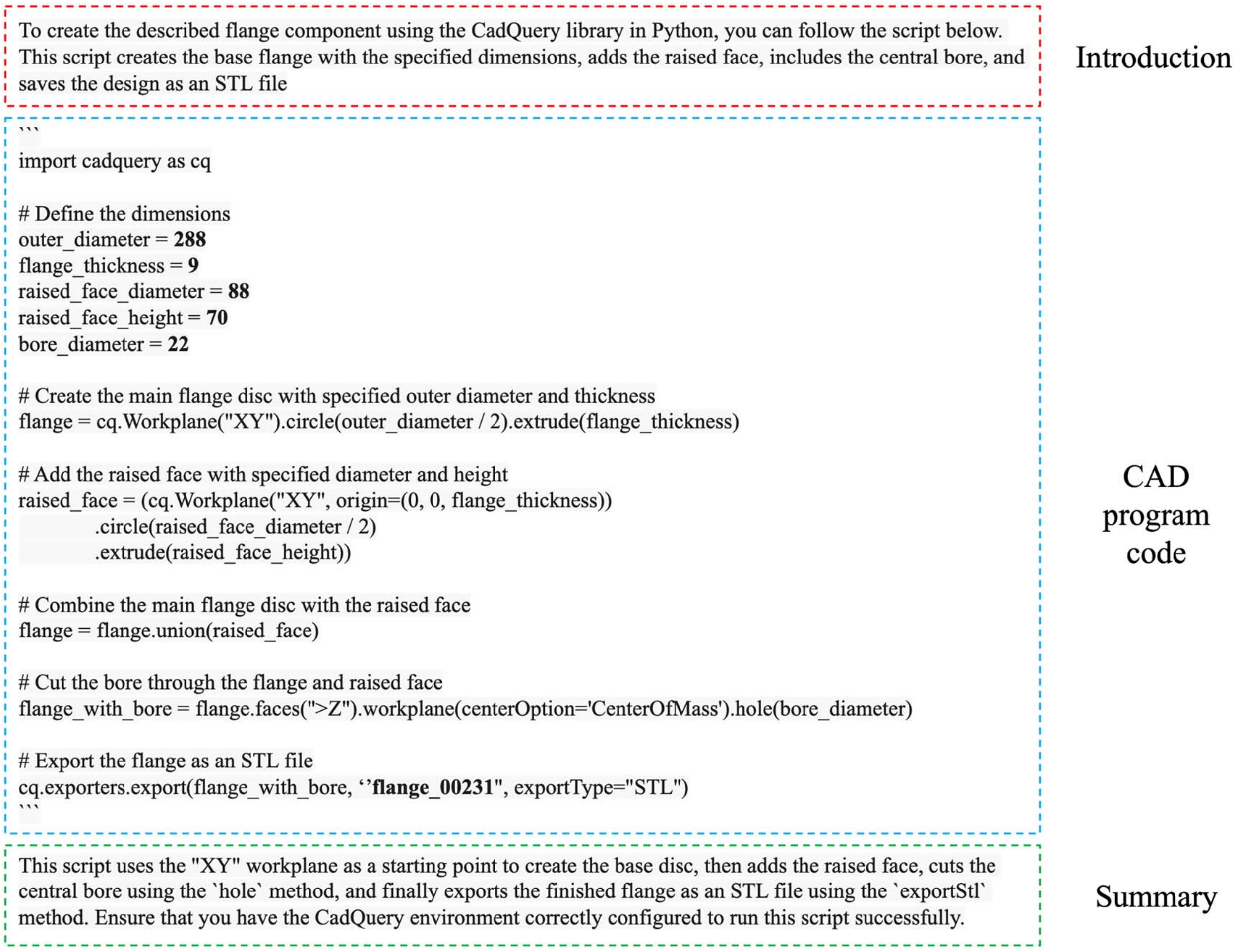


Fig. 9 Results of IoU of (*a*) the overall performance and (*b*) the category-specific performance of the GPT-4V model for four input modes

further underscoring the superior efficacy of the text-only input mode. This observation challenges our assumption on 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.

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 ML models may be contingent upon how relevant the



Introduction

To create the described flange component using the CadQuery library in Python, you can follow the script below. This script creates the base flange with the specified dimensions, adds the raised face, includes the central bore, and saves the design as an STL file

CAD program code

```
import cadquery as cq

# Define the dimensions
outer_diameter = 288
flange_thickness = 9
raised_face_diameter = 88
raised_face_height = 70
bore_diameter = 22

# Create the main flange disc with specified outer diameter and thickness
flange = cq.Workplane("XY").circle(outer_diameter / 2).extrude(flange_thickness)

# Add the raised face with specified diameter and height
raised_face = (cq.Workplane("XY", origin=(0, 0, flange_thickness))
        .circle(raised_face_diameter / 2)
        .extrude(raised_face_height))

# Combine the main flange disc with the raised face
flange = flange.union(raised_face)

# Cut the bore through the flange and raised face
flange_with_bore = flange.faces(">Z").workplane(centerOption='CenterOfMass').hole(bore_diameter)

# Export the flange as an STL file
cq.exporters.export(flange_with_bore, ''flange_00231", exportType="STL")
```

Summary

This script uses the "XY" workplane as a starting point to create the base disc, then adds the raised face, cuts the central bore using the `hole` method, and finally exports the finished flange as an STL file using the `exportStl` method. Ensure that you have the CadQuery environment correctly configured to run this script successfully.

Fig. 10 An example of LLM's response given the design requirements for a flange which consists of three sections: introduction, CAD program code, and summary

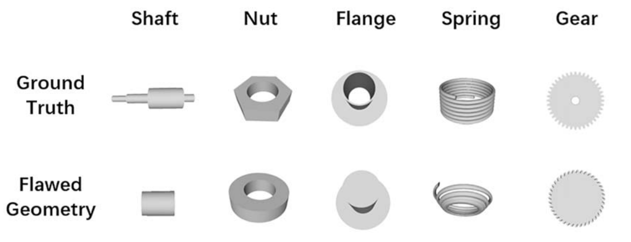


Fig. 11 Examples of flawed geometry generated by the GPT-4V model

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. 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 [36].

Nonetheless, the category-specific results indicate that as the geometries of the CAD models become more complex, multimodal input becomes more effective. Considering real-world design scenarios where products are more complicated in terms of both geometries and structures, multimodal input modes are expected to exhibit better performance than text-only input modes. Exploring this further would present an intriguing avenue for future research.

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 the overall 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, as shown in Fig. 7(*b*), the text + sketch + image input is the best for springs, but text + 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 efficacy of multimodal inputs aligns more closely with that of text-only input for each category and in terms of the overall result compared to the result obtained without using the debugger. 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 Versus 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 comparable

Table 4 Summary of the (a) parsing rate and (b) IoU for GPT-4 and GPT-4V models using the text-only input

	GPT-4	GPT-4V
(a) Parsing rate		
Zero-shot learning ability	0.52	**0.53**
Debugger-enhanced performance	0.71	0.71
(b) IoU		
Zero-shot learning ability	0.57 ± 0.36	$\mathbf{0.67 \pm 0.32}$
Debugger-enhanced performance	0.54 ± 0.38	$\mathbf{0.59 \pm 0.37}$

Note: P-values that meet the significance threshold of 0.05 are highlighted in bold.

parsing rate but much higher IoU scores, as summarized in Table 4. 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 Apr. 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 differences in their performance is not clear 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 7 demonstrates that the integration of the debugger enhances the parsing rates for each category of the mechanical components and in terms of the overall average performance 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 improvement in the parsing rate comes at the cost of the reduction in the IoU values for both models, as detailed in Figs. 8 and 9 in terms of the overall average result.

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 (i.e., text, images, or sketches). 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 can 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. However, this does not mean that the GPT models ignore the design requirements. We conducted experiments using a new debugger where we only fed the synthesized CAD programs back to the GPT models during the debugging process without giving the input modalities. Compared to the debugger we used, we obtained *lower parsing rate values* for the new debugger (i.e., GPT-4: 0.690. GPT-4V: (1) text, 0.678; (2) text + sketch, 0.667; (3) text + image, 0.670; and (4) text + sketch + image, 0.681. See the results of the debugger we used in Fig. 7(*c*) for a comparison) and *similar IoU values* (i.e., all P-values>0.05 for the independent T-test conducted for average values and category-specific values, except for text-only input for average value (P-value =0.02) and text + sketch + image input for gears (P-value =0.04)). This also provides practical evidence that the input modalities should be fed to the GPT models when designing a debugger.

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. This work focused on assessing the zero-shot learning ability of multimodal LLMs and proposed a debugging method for LLMs to improve their output quality in an iterative manner. Future work can explore strategies, such as in-context learning, few-shot learning, and fine-tuning, to further enhance LLM4CAD.

In terms of the dataset, we only evaluated five representative categories of mechanical components with different geometric complexities. Although the insights gained from the current synthesized dataset are valuable, we are cautious to generalize our conclusions due to the small number of mechanical component categories compared to the wide array of mechanical components. Future research efforts are needed to achieve a more comprehensive and generalizable understanding of the role multimodal LLMs play in the generation of 3D CAD models. In terms of the diversity of the CAD dataset, incorporating human-generated sketches could further enhance the multimodal value of the dataset, particularly in real-world design applications where users might want to sketch something quickly to express their design preferences and requirements for LLMs. On the other hand, it is also critical to note that designs often consist of interconnected components in the form of assemblies rather than individual components [40,41]. 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.

For the implementation of the experiments, the need for dimensional information to conduct quantitative comparisons between generated 3D designs and their ground-truth counterparts required the inclusion of textual descriptions as one of the input modalities. This requirement constrained our ability to conduct a broader range of ablation studies, such as those involving sketch-only or sketch + image inputs, as these modes do not inherently provide the dimensional data needed for accurate quantitative evaluation. Future work could focus on expanding the dataset to include richer sources of dimensional information or developing methodologies to extract dimensional data from non-textual inputs to alleviate this limitation and provide a more holistic assessment of LLM4CAD.

Lastly, we only tested OpenAI's GPT-4 and GPT-4V models for the LLMs and generated CAD programs through CADQuery. Future research could explore additional CAD programming languages, such as OpenSCAD and Fusion 360 PYTHON API, as well as other LLMs, such as Google's Gemini, to provide a more comprehensive understanding of the capability of multimodal LLMs in the 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? and (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 by just leveraging their zero-shot learning ability. Especially, the performance can be further enhanced by a debugging process to iteratively refine the output CAD program code. Additionally, in our experiment of GPT-4V, 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, multimodal inputs start to gain prominence with more complex geometries (e.g., shafts and gears) in terms of the successful parsing rate of the generated CAD programs and the geometric accuracy.

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 great 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.

Acknowledgment

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

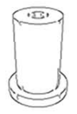
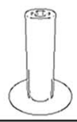
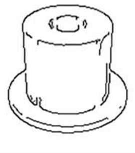
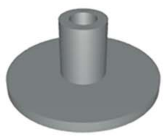
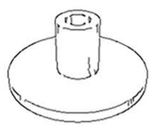
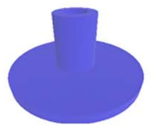
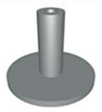
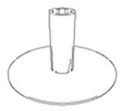
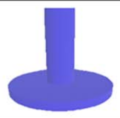
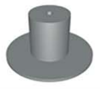
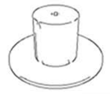
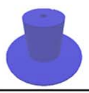
Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data and information that support the findings of this article are freely available.[5]

Table 5 Details of the dataset using flanges as examples

Index	Component Name	2D Image	2D Sketch	3D Mesh	Text Description
1	Flange_00001				The object is a flange with a diameter of 124mm and a thickness of 19mm, featuring a raised face with a diameter of 90mm and a height of 141mm. It has a bore diameter of 36mm.
2	Flange_00002				It is flange which has diameter in 125mm with the thickness 3mm, and it has raised face diameter which is equal to 50mm with bore diameter is 30mm and their raised face height is 162mm.
3	Flange_00003				A flange with a wide base and long face. It has a total diameter of 135mm for the flange, with a thickness of 8mm. The raised face is 93mm in diameter, and the height of it is 84mm. Finally, the bore diameter in the center is 38mm.
...	...	...	...	...	...
692	Flange_00692				The flange is a circular metal component measuring 139mm in diameter and 17mm in thickness. It has a raised face with a diameter of 63mm and a height of 63mm. The bore diameter is 45mm.
693	Flange_00693				None
...	...	...	...	...	...
1000	Flange_01000				None

[5]https://doi.org/10.18738/T8/KV7HON

Appendix

Table 5 presents more details of our dataset used in our experiments demonstrated using examples of flanges. Along with the GT 3D shapes, the table includes visual data (images and sketches rendered from the GT 3D shapes) and textual data (text descriptions obtained via crowdsourcing). It is important to note that "None" indicates the absence of valid textual descriptions obtained through the crowdsourcing process.

References

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., et al., 2020, "Language Models Are Few-Shot Learners," Adv. Neural Inform. Process. Syst., **33**, pp. 1877–1901.

[2] Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., et al., 2023, "Chatgpt for Good? On Opportunities and Challenges of Large Language Models for Education," Learn. Individual Differ., **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., and Wahid, A., 2023, "Palm-e: An Embodied Multimodal Language Model," Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, July 23–29, PMLR, pp. 8469–8488.

[5] Kocaballi, A. B., 2023, "Conversational AI-Powered Design: Chatgpt as Designer, User, and Product," 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," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Boston, MA, Aug. 20–23, p. V006T06A021.

[8] Li, X., Wang, Y., and Sha, Z., 2023, "Deep Learning Methods of Cross-Modal Tasks for Conceptual Design of Product Shapes: A Review," ASME J. Mech. Des., **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," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, St. Louis, MO, Aug. 14–17, p. V006T06A016.

[10] Song, B., Zhou, R., and Ahmed, F., 2024, "Multi-modal Machine Learning in Engineering Design: A Review and Future Directions," ASME J. Comput. Inf. Sci. Eng., **24**(1), p. 010801.

[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," ASME J. Mech. Des., **144**(11), p. 114501.

[12] Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G., 2023, "Pal: Program-Aided Language Models," Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, July 23–29, PMLR, pp. 10764–10799.

[13] Li, X., Sun, Y., and Sha, Z., 2024, "Llm4cad: Multi-modal Large Language Models for 3d Computer-Aided Design Generation," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Washington, DC, Aug. 25–28, p. V006T06A015.

[14] Nelson, M. D., Goenner, B. L., and Gale, B. K., 2023, "Utilizing Chatgpt to Assist CAD Design for Microfluidic Devices," Lab Chip, **23**(17), pp. 3778–3784.

[15] Baltrušaitis, T., Ahuja, C., and Morency, L.-P., 2018, "Multimodal Machine Learning: A Survey and Taxonomy," IEEE Trans. Pattern Anal. Mach. Intell., **41**(2), pp. 423–443.

[16] Song, B., Miller, S., and Ahmed, F., 2023, "Attention-Enhanced Multimodal Learning for Conceptual Design Evaluations," ASME J. Mech. Des., **145**(4), p. 041410.

[17] Su, H., Song, B., and Ahmed, F., 2023, "Multi-modal Machine Learning for Vehicle Rating Predictions Using Image, Text, and Parametric Data," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Boston, MA, Aug. 20–23, p. V002T02A089.

[18] Chowdhary, K. R., 2020, "Natural Language Processing," *Fundamentals of Artificial Intelligence*, Springer, New Delhi, pp. 603–649.

[19] OpenAI, 2022, "Introducing ChatGPT: Optimizing Language Models for Dialogue," November, https://openai.com/blog/chatgpt/, Accessed January 16, 2024.

[20] 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 J. Autom. Sin., **10**(5), pp. 1122–1136.

[21] Ray, P. P., 2023, "Chatgpt: A Comprehensive Review on Background, Applications, Key Challenges, Bias, Ethics, Limitations and Future Scope," Int. Things Cyber-Phys. Syst., **3**, pp. 121–154.

[22] 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 Trans. Bench. Standards Eval., **2**(4), p. 100089.

[23] Abdullah, M., Madain, A., and Jararweh, Y., 2022, "Chatgpt: Fundamentals, Applications and Social Impacts," Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS), Milan, Italy, Nov. 29–Dec. 1, pp. 1–8.

[24] Gulwani, S., Polozov, O., Singh, R., 2017, "Program Synthesis," Found. Trends Programm. Lang., **4**(1–2), pp. 1–119.

[25] Wang, X., Anwer, N., Dai, Y., and Liu, A., 2023, "Chatgpt for Design, Manufacturing, and Education," Procedia CIRP, **119**, pp. 7–14.

[26] Makatura, L., Foshey, M., Wang, B., HähnLein, F., Ma, P., Deng, B., Tjandrasuwita, M., et al., 2023, "How Can Large Language Models Help Humans in Design and Manufacturing?" preprint arXiv:2307.14377.

[27] Wu, F., Hsiao, S.-W., and Lu, P., 2024, "An Aigc-Empowered Methodology to Product Color Matching Design," Displays, **81**, p. 102623.

[28] Grandi, D., Patawari Jain, Y., Groom, A., Cramer, B., and McComb, C., 2025, "Evaluating Large Language Models for Material Selection," ASME J. Comput. Inf. Sci. Eng., **25**(2), p. 021004.

[29] Meltzer, P., Lambourne, J. G., and Grandi, D., 2024, "What's in a Name? Evaluating Assembly-Part Semantic Knowledge in Language Models Through User-Provided Names in Computer Aided Design Files," ASME J. Comput. Inf. Sci. Eng., **24**(1), p. 011002.

[30] Naghavi Khanghah, K., Wang, Z., and Xu, H., 2025, "Reconstruction and Generation of Porous Metamaterial Units Via Variational Graph Autoencoder and Large Language Model," ASME J. Comput. Inf. Sci. Eng., **25**(2), p. 021003.

[31] OpenAI, 2023, "Gpt-4 Technical Report," preprint arXiv:2303.08774.

[32] 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," Proceedings of the 16th European Conference on Computer Vision (ECCV), Glasgow, Aug. 23–28, Springer, pp. 175–191.

[33] 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," J. Comput. Des. Eng., **9**(1), pp. 114–127.

[34] 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," Comput. Graph., **99**, pp. 100–113.

[35] 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.

[36] Luo, T., Rockwell, C., Lee, H., and Johnson, J., 2023, "Scalable 3D Captioning with Pretrained Models," Advances in Neural Information Processing Systems 36, New Orleans, LA, Dec. 10–16, Vol. 36, pp. 75307–75337.

[37] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., and Sastry, G., 2021, "Learning Transferable Visual Models From Natural Language Supervision," Proceedings of the 38th International Conference on Machine Learning, Virtual Event, July 18–24, PMLR, pp. 8748–8763.

[38] Mason, W., and Suri, S., 2012, "Conducting Behavioral Research on Amazon's Mechanical Turk," Behav. Res. Methods, **44**(1), pp. 1–23.

[39] Lopez, C. E., Miller, S. R., and Tucker, C. S., 2019, "Exploring Biases Between Human and Machine Generated Designs," ASME J. Mech. Des., **141**(2), p. 021104.

[40] Li, X., Xie, C., and Sha, Z., 2023, "Design Representation for Performance Evaluation of 3d Shapes in Structure-Aware Generative Design," Des. Sci., **9**, p. e27.

[41] 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, Virtual Event, Jan. 5–8, pp. 5250–5259.