Proceedings of the ASME 2025
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE2025
August 17-20, 2025, Anaheim, CA

IDETC2025-163946

MEDA: A Multi-Agent System For Parametric CAD Model Creation

Nirmal Prasad Panta[1,6,†,*], Saugat Kafley [1], Rujal Acharya [1], Sashank Parajuli [1], Dikshya Parajuli [1], Prince Panta [1,4], Saroj Belbase [1], Sudikshya Pant [1], Amit Regmi[1,3,4,*], Akio Tanaka[1,2,3], Christopher McComb[5]

[1]Accelerated Komputing Pvt. Ltd., Bhaktapur, Nepal
[2]godelblock Inc., Tokyo, Japan
[3]Accelerated Komputing Inc., Pittsburgh, PA
[4]University of Pittsburgh, Pittsburgh, PA
[5]Carnegie Mellon University, Pittsburgh, PA
[6]Texas A&M University, Texas, TX

ABSTRACT

Parametric modeling is a critical technique in engineering design that enables the rapid generation and evaluation of candidate designs. To create parametric models, engineers need to be familiar with various Computer-Aided Design (CAD) software or have a strong grasp of at least one CAD scripting library. Modern multi-modal large language models (MLLMs) present an opportunity to automate parametric modeling, as they have shown the ability to write and execute code and also to analyze outputs such as images for further refinement. Based on these multi-dimensional capabilities, we propose Mechanical Engineering Design Agents (MEDA), an autonomous multi-agent framework that leverages AI agents to emulate human-like division of labor to create parametric CAD models. Our framework employs a combination of zero-shot and one-shot learning for the constituent agents, striking a balance between efficiency and accuracy.

We evaluate our autonomous multi-agent framework using a dataset of 200 CAD prompts. MEDA achieves a success rate of 99% in script execution. Furthermore, we observe a minimal median point cloud distance of 0.0555 between generated and ground truth CAD models, a 56% reduction compared to prior work. Our findings demonstrate that through division of labor and effective collaboration, Artificial Intelligence (AI)-powered agents can autonomously generate more accurate CAD models relying primarily on their pre-trained knowledge. This paper highlights the significant potential of employing collaborative and dynamic multi-modal AI agents for design automation while also underscoring the current limitations of MLLMs in parametric CAD modeling.

Code and data are available at : https://github.com/AnK-Accelerated-Komputing/MEDA

Keywords: MLLM, CAD, AI Agents, Design Automation

1. INTRODUCTION

Parametric CAD models may be tedious to create, but once instantiated, they enable the rapid creation of vast design space by altering the relevant parameters [1]. However, with the increasing complexity of the design, it becomes more challenging for designers to keep track of the parameters involved in CAD creation [2]. In addition to managing the parameters, the designer must also execute CAD operations or use CAD script libraries. This range of skills is rarely available in a single designer or engineer. These limitations highlight the need for advanced approaches that can efficiently generate and refine CAD models directly from the requirements [3].

Most of the fundamental operations required to produce a parametric model, such as scripting, visual inspection, and debugging, are increasingly feasible using AI [4]. LLMs have shown that they can analyze, retrieve, and generate relevant knowledge and concepts from various domains and help with design ideation, specification generation in design, and manufacturing [5, 6]. Modern MLLMs also show human-level performance on various professional and academic benchmarks without being specifically trained for such tests [7]. They excel at following instructions, general reasoning, multi-modal capabilities, and agentic coding [8]. However, hallucination still remains a significant challenge when using these models [7, 9]. One of the approaches used to mitigate hallucinations and produce reliable

[†]First author
[*]Corresponding authors: nirmal@ank-world.com, amit@ank-world.com

and trustworthy results is to orchestrate multiple specialized AI agents [10].

Furthermore, researchers have already begun investigating the use of MLLMs for the generation and refinement of 3D CAD models [11–13]. Previous works focused primarily on purely LLM-driven approaches or, in a few cases, multi-agent systems that lack autonomous behavior and depend on user involvement for feedback and validation [14].

In order to achieve greater efficiencies and independent decision-making through agentic specialization, we introduce a multi-agent framework for automating parametric CAD model creation— Mechanical Engineering Design Agents (MEDA). In MEDA, a set of specialized agents collaborate to break down the design prompt, plan the design process, write code, execute it, review the output, and iterate until a successful CAD model is created. We implement division of labor with mutual corrections in the agents to assist in design automation through CAD model generation paving the way for automation of problem solving in Computer Aided Engineering (CAE) and Computer Aided Manufacturing (CAM). The major contributions of our work are:

1. We introduce an autonomous multi-agent framework for generating parametric 3D CAD models using natural language descriptions.
2. We demonstrate a successful compilation rate of 99% and minimal median point cloud distance of just 0.055 between ground truth and generated CAD model using MEDA.
3. We explore the effectiveness of specialized agents in the automation of complex problem solving, with and without multi-modal self-refinement capability.

The remainder of this paper is structured as follows. The next section reviews prior research on parametric CAD model generation using LLMs, MLLMs, and agent-based systems for engineering design. Subsequently, the methodology section introduces MEDA framework, detailing its multi-agent architecture and collaborative design process. The results section presents an evaluation of MEDA's effectiveness using a dataset of CAD design prompts. This was done to highlight performance improvements over previous approaches. Finally, the paper concludes with a discussion of key insights, limitations, and opportunities for future research.

2. RELATED WORKS

This section provides an overview of foundational concepts and prior research related to parametric CAD modeling, the use of LLMs and MLLMs for automated design, and multi-agent approaches for engineering design automation.

2.1. Parametric CAD Creation

Parametric CAD modeling is the industry standard approach for creating designs, as it naturally supports re-usability and design space exploration [15, 16]. This modeling approach enables rapid solution search across a variety of engineering fields [17, 18]. However, incorporating design intent naturally in the parametric CAD creation is challenging, often requiring an experienced designer [16]. For that reason, various CAD datasets have been developed to advance 3D shape learning and geometric deep learning for the generation of parametric CAD models [19–22].

For instance, Khan et al. [23] introduced a transformer-based architecture that is trained on sequences of model operations. Other research has explored approaches to parametric CAD creation that make use of multi-modal contrastive learning by training on large CAD datasets [20, 24]. The performance of these approaches is heavily influenced by their training data and their robustness outside of the training data may not be satisfactory. For that reason, this work focuses on CAD generation through scripts, as they make the parameters involved in CAD generation explicit and, thus, more trustworthy [15].

2.2. LLMs for CAD model generation through coding

CAD model generation has been an objective of LLM systems almost since their inception. The studies performed by Makatura et al. [25] explored the capabilities of LLMs in creating design and design space from text using the OpenSCAD library through iterative dialogue between the user and the generative model. They also pointed out the limitations of these models in spatial reasoning. Expanding on this work, Picard et al. [26] performed experiments with GPT-4V, a MLLM, to assess its ability to understand image data for the description of the CAD model, exploring an approach requiring iterative refinement between the user and AI. Extending the work of Makatura et al., that involved high-level inputs, other researchers introduced a self-refining framework for parametric modeling by providing a detailed prompt and using LLMs to control 3D software through code [27]. However, the refinement strategy used here lacked iterative improvement in the CAD scripts themselves. Similarly, Query2CAD utilized LLM capabilities to generate FreeCAD macro scripts for creating desired CAD models, and then used the BLIP-2 model (a rudimentary image captioning model) for visual feedback [28].

Recently, Alrashedy et al. [13] proposed CADCodeVerify to refine the CAD code using an MLLM for improvement through question development and answering process. Similarly, LLM4CAD used a debugger to generate executable code to create CAD models focusing on five mechanical components, such as gears, flanges, and shafts [11]. However, these frameworks for parametric CAD model generation do not leverage autonomous agents or emulate human-like cognition and collaborative problem-solving strategies. Hence, we have introduced MEDA to explore the potential of an autonomous, multi-agent approach to parametric CAD model generation.

2.3. Agentic systems for 3D CAD modeling

LLMs demonstrate innate capabilities in various applications due to the incomprehensible scale of their training data [29]. Intelligent generative agents defined with specific roles and instructions can work together to handle complex tasks that require the execution of a series of sub-tasks such as planning, coding, reviewing and debugging [30] - all of which are core to the creation of parametric models. Agentic workflows can also decrease the likelihood of hallucinations and increase the reliability of the final output [31].

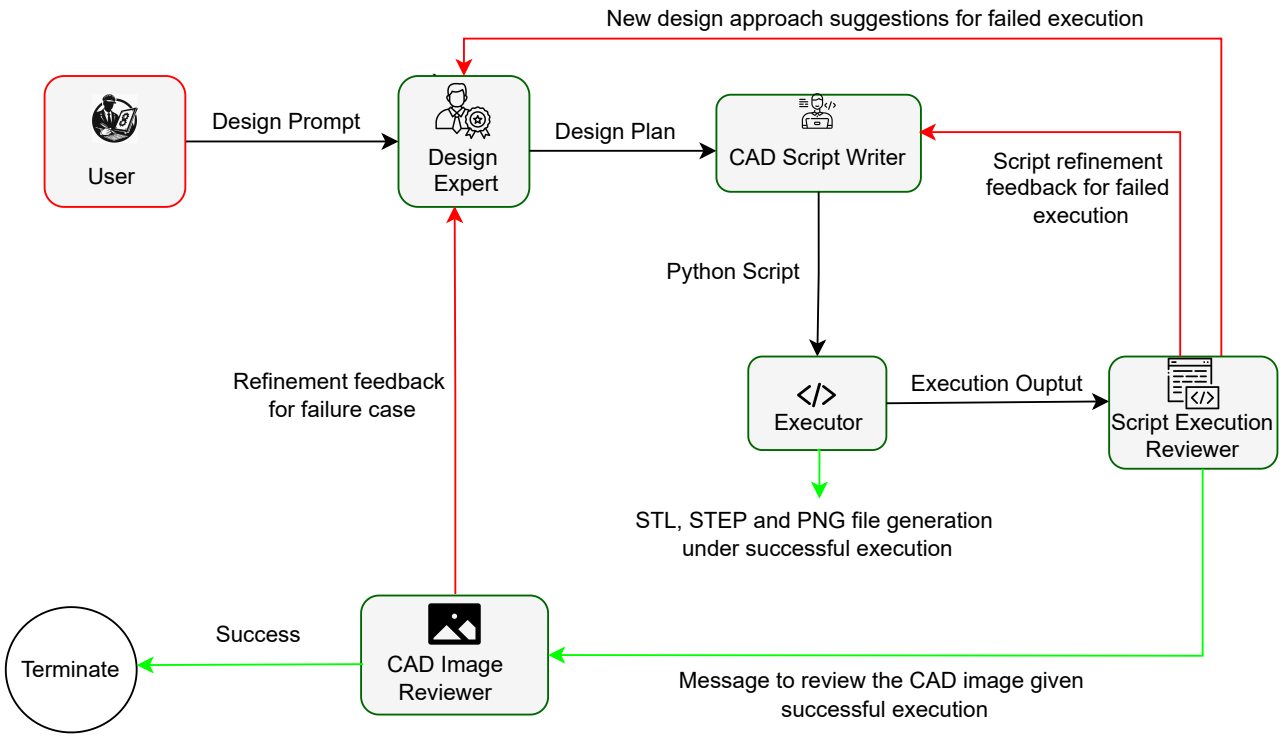


FIGURE 1: MECHANICAL ENGINEERING DESIGN AGENTS (MEDA) OVERVIEW: THIS ARCHITECTURE UTILIZES A MULTI-AGENT SYSTEM FOR PARAMETRIC CAD MODEL GENERATION.

Previous works have explored the generation of 3D CAD content through a multi-agent system, but those works are focused on scene generation rather than parametric modeling [32]. In addition, others have explored single-agent approaches to mechanical design tasks, but that work failed to capitalize on the efficiencies feasible with multiple agents [33]. More recently, Ocker et al. [14] introduced a human-AI multi-agent team for CAD model creation, incorporating human feedback to validate the generated model. Albeit promising, this approach depends on human input and does not explore autonomous CAD model generation. In contrast, this work proposes an autonomous multi-agent architecture for 3D CAD model generation, highlighting its potential to automate related engineering tasks such as computational design, simulation, and manufacturing.

3. METHODOLOGY

This section describes the roles of individual agents that make up MEDA, their collaborative workflow, and the evaluation approach used to assess the performance of MEDA.

3.1. MEDA Architecture

Figure 1 illustrates the overall architecture of MEDA. The workflow is initiated by a user who provides the natural language prompt to create the CAD model with the desired specification. Each agent then plays a dedicated role to achieve the common goal of creating the required CAD model through mutual collaboration.

Design Expert. This agent interprets and analyzes the specifications mentioned in the prompt from the user. In case the prompt lacks relevant dimensions for the CAD model, it makes appropriate assumptions of the missing dimensions.[1]

For a partially specified prompt such as "Design a box.", the agent fully assumes necessary dimensions such as length, breadth, and width. It then generates a modeling plan stating the relevant methods and functions of open-source Python library, CadQuery [34], to generate the required CAD model according to the specifications. To counteract the effect of poor spatial reasoning in language models [35], this agent is carefully instructed to use appropriate coordinates by adopting a fixed datum and precise dimension-driven equations for positioning the features, such as holes in the 3D CAD model.

CAD Script Writer. This is the only agent provided with one-shot learning to generate a Python script in the desired structure. This enables parameterization and generation of files of our choice. The agent is provided with a system prompt that describes how it should write a Python script- by importing the required libraries, stating the parameters of the CAD model, using the relevant methods for CAD creation, and finally exporting the CAD model in formats such as .stl and .stp [2].

Executor. This agent takes a message from the CAD Script Writer containing Python code block and saves the code to a

[1] This could also be supplemented with a user-querying procedure, but this work is focused on autonomous evaluation.

[2] The system messages used for all agents described here are provided in the codebase linked in the abstract.

.py file. It executes the code in the local environment via the command line. It then reads the console output after the execution and forwards it to the Script Execution Reviewer. This agent uses a designated working directory to run the Python scripts and generate the required CAD files.

Script Execution Reviewer. This agent analyzes the message from Executor to decide if the execution was a success or a failure. If the execution was a failure, as suggested by the presence of error logs, this agent provides feedback to the CAD Script Writer on how to modify the code to mitigate such errors.

The Script Execution Reviewer can also provide suggestions to the Design Expert, as illustrated in Fig. 1, to change the design approach altogether when execution fails. This ensures the exploration of alternative modeling strategies. For example, when creating a cone, it can be directly constructed using `Solid.makeCone` method available in CadQuery. Alternatively, a cone can be created by sketching a right triangle with a base equal to the cone's radius and a perpendicular equal to the cone's height, and then revolving the triangle about its perpendicular. In this case, the Design Expert might mistakenly use a non-existent `cone` method, resulting in execution failure. To address this, the Script Execution Reviewer agent may escalate to Design Expert to revise a plan that uses a new method or an entirely new design approach.

Finally, if script execution is a success, this agent escalates to CAD Image Reviewer for reviewing the generated CAD model.

CAD Image Reviewer. This agent is equipped with a tool that analyzes the CAD model image and provides description based on its visual appearance. It compares this description with the design prompt to evaluate correct model generation. The tool call enables analysis of the most recently generated CAD model image after successful exectuion, producing a description based on the capabilities of the MLLM. If the generated description satisfies the design prompt, then this agent terminates the chat. Otherwise, it provides refinement feedback to Design Expert for re-approaching the design problem.

3.2. MEDA Implementation

We construct our agents using Autogen [36], an open-source platform that enables multi-agent applications. All agents inherit attributes and methods from Autogen's `ConversableAgent` class, which allows them to send and receive messages with one another and to integrate LLMs, tools, or human-in-the-loop feedback.

The user is defined by the `UserProxyAgent` subclass of `ConversableAgent`, which acts as a human proxy to receive the design prompt as the initial step in the conversation. The Design Expert, CAD Script Writer, and Script Execution Reviewer are agents implemented using `AssistantAgent` subclass and are configured to use an LLM. The Executor, also a conversable agent, is powered by local command-line code execution to run Python scripts. Finally, the CAD Image Reviewer is defined as an assistant agent integrated with MLLM powered tool to compare the design prompt against the AI-generated description of the CAD model image.

Collaboration between these agents is enabled by a group chat created using `GroupChat` class. This group chat can be configured to operate in a dynamic mode, allowing transitions to any agent with or without the decision of LLM, or in a restrictive mode with controlled transitions. To ensure robust and reproducible performance of the MEDA architecture, we define a finite state machine (FSM) that constrains agent transitions during the conversation, as shown in Figure 1. In the given figure, the directed edge represents a possible transition path. This FSM is implemented as a key-value pair in a Python dictionary, where each key represents the current agent, and the corresponding value is a list of agents allowed to act next. This Python dictionary is passed as input when instantiating the `GroupChat` enabling the controlled transitions based on FSM. When multiple agents are allowed in the value list, the next agent is selected dynamically, guided by LLM's response. By enforcing this structure, we ensure logical sequencing of the agents and their operations, prevent unintended transitions, and improve overall multi-agent performance.

The group chat is orchestrated by a group chat manager, instantiated using `GroupChatManager` class, which selects the next agent based on the transition policy described earlier and broadcasts its response to the other agents. The feedback link between the user and other agents is severed to enable design automation.

3.3. Evaluation Approach

In order to evaluate the performance of MEDA, we design a multi-stage experimental pipeline that includes dataset selection, model setup, and quantitative assessment using established metrics.

Prompt Dataset. Our evaluation dataset consists of 200 natural language prompts and their corresponding ground truth CAD models from the CADPrompt dataset [13, 20] (see Fig. 2 for example models from the dataset). We restrict our tests to utilize the prompts with specific measurements. Two examples of the representative design prompts are presented in Section 4.3. Although, these prompts fully specify CAD model dimensions, they might lack plane selection and axis-alignment instructions, thus necessitating alignment of CAD models before evaluation.

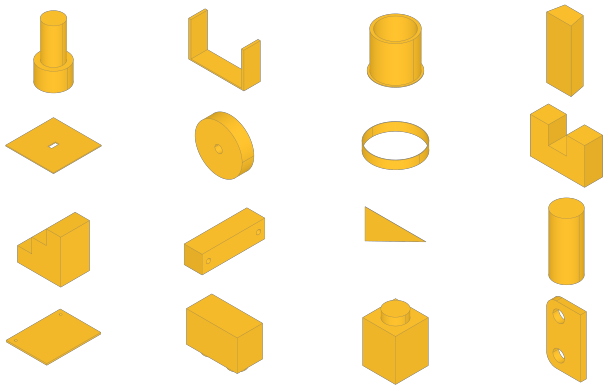


FIGURE 2: RANDOMLY SELECTED REPRESENTATIVE CAD MODELS FROM CADPROMPT DATASET [13, 20]

MLLM Experimental Setup. To create CAD models from the design prompts using our multi-agent framework, we use the GPT-4o-2024-0806 model through Azure OpenAI Service as the driving LLM. GPT-4o is selected for its code generation capabilities as well as its multi-modal capabilities. The temperature of the model is set to 0.3 for a deterministic yet creative CAD generation approach for most agents. The only exception is the temperature of the Design Expert, which is set to 0.6 to result in the balanced creativity that is required for planning and designing novel CAD models. The pricing of this model is set at $2.5 per million input tokens and $10 per million output tokens for the global model [37].

Evaluation Metrics. We compare our multi-agent approach with a recent CAD generation framework— CADCodeVerify [13], and use three evaluation metrics for quantitative comparison of the ground truth CAD models with the generated CAD models. These three metrics are specifically used, despite their inherent noisiness, to allow direct performance comparison of our multi-agent framework with CADCodeVerify. The compile rate for successful execution of Python Scripts for CAD generation is also reported. The three CAD evaluation metrics can be defined as follows for two sets of point clouds A and B:

1. Point Cloud Distance (PCD): It defines the similarity between two sets of point clouds by measuring the average point distances in both direction.

$$D(A,B) = \frac{1}{2|A|}\sum_{a\in A}\min_{b\in B} d_{a,b} + \frac{1}{2|B|}\sum_{b\in B}\min_{a\in A} d_{a,b} \quad (1)$$

   Where $d_{a,b} = ||a - b||_2$ is Euclidean distance between points a and b [38].

2. Hausdorff Distance (HD): It measures the greatest distance between any pair of nearest neighbors between point clouds [39].

$$H(A,B) = \max\left\{\max_{a\in A}\left\{\min_{b\in B} d_{a,b}\right\}, \max_{b\in B}\left\{\min_{a\in A} d_{a,b}\right\}\right\} \quad (2)$$

3. Intersection over Ground Truth (IoGT): It measures the overlap between two point clouds by calculating the ratio of intersection area of A and B to the area of B[13].

$$IoGT = \frac{|A \cap B|}{B} \quad (3)$$

We report the median and inter-quartile range (IQR) of these metrics to facilitate direct comparison with the results reported in [13]. To ensure consistency, we adopt the same evaluation approach as in [13]. We use Iterative Closest Point algorithm (ICP) to optimally align the generated CAD models with the ground truth models before evaluation and also implement the same approach of using the worst-case scenario values of 1.732 for PCD, 1.732 for HD, and 0 for IoGT in cases of execution failure. In an ideal case of perfect similarity between the generated CAD model and ground truth CAD model, the PCD and HD would be 0 and IoGT would be 1.

Ablation Study. In order to build a deeper understanding of MEDA's behavior, we conduct an ablation study. Specifically, we start with a baseline multi-agent system, consisting of the User, Design Expert, CAD Script Writer, and Executor. This system is used to evaluate how well the LLM-driven multi-agent system can autonomously create parametric CAD models without any feedback or iteration. The baseline mimics the generic human design workflow of planning the steps of design through Design Expert, implementing the codes using CAD Script Writer, and executing for CAD model generation using Executor. This baseline represents the point at which the first CAD model is generated before any refinement occurs.

Then, we progressively add additional agents to understand their impact. This starts with adding the Script Execution Reviewer, which enables iteration to increase the execution success rate through code refinement (here, we evaluate both single-refinement and double-refinement versions of the agent). This provides us with insights about the optimal number of refinements required to obtain the most successful execution before adding the agent to review the generated CAD model.

Next, the CAD Image Reviewer is added to the system to enhance the CAD models generated. Given that, CAD model is generated through successful compilation, we ensure that the CAD Image Reviewer has at least three refinement turns, for the case of incorrect CAD model recognition, before terminating the conversation. We also provide the insights on cost incurred for each ablation study. Although a typical ablation study would remove each component in isolation, the dependencies in MEDA require a more precise procedure.

4. RESULTS

The results presented here are based on four primary evaluations. First, we conduct a comparison of MEDA against CADCodeVerify, a state-of-the-art implementation for the same task. Next, we conduct the ablation study to further understand the contribution of unique agents to the MEDA system.

4.1. Evaluation of Framework

Table 1 shows the results from testing MEDA with 200 prompts from CADPrompt dataset. Our multi-agent framework surpasses every performance metric of CADCodeVerify except IoGT when we consider the median values. With MEDA, we achieved successful CAD creation for 198 out of 200 prompts. As a result, our framework achieved a 56.3% reduction in point cloud distance, and the Hausdorff distance was reduced by 37.3%. Although our IoGT median is lower than that of CADCodeVerify, this slight discrepancy can be attributed to the noisiness of IoGT as a metric. The considerable improvements in most evaluation metrics clearly demonstrate the effectiveness of MEDA.

The box plot in Fig. 3 provides a visual representation of the distribution of these metrics. We can observe the presence of some outlier values for PCD and IoGT (the two extreme outlier values resulting from execution failure are not shown to allow for a zoomed-in visualization), suggesting that while MEDA generally performs well, there are instances where the generated CAD models significantly deviate from the ground truth even after review by the MLLM. This observation is further supported by

TABLE 1: COMPARISON OF MEDA AGAINST CADCODEVERIFY

Framework	MLLM used	Point Cloud dist. ↓	Hausdorff dist. ↓	IoGT ↑	Compile Rate ↑
MEDA	GPT-4o	**0.0555** (0.095)	**0.2628** (0.401)	0.9413 (0.0275)	**99%**
CADCodeVerify	GPT-4	0.127 (0.135)	0.419 (0.356)	**0.944** (0.028)	96.5%

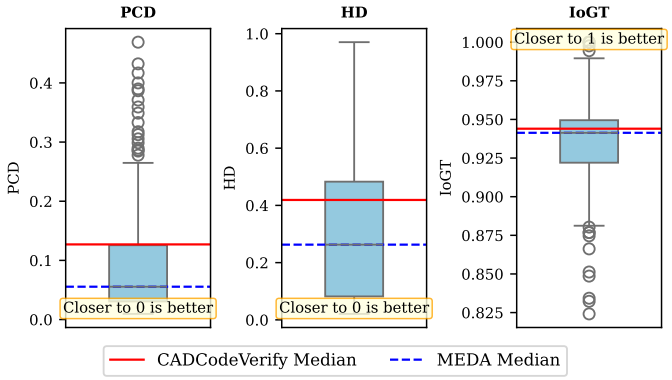


FIGURE 3: BOX PLOT SHOWING THE DISTRIBUTION OF CAD EVALUATION METRICS

Fig. 4, which shows the same subset of randomly selected CAD models from Fig. 2 generated by MEDA. These generated CAD models demonstrate MEDA's ability to produce accurate models while also highlighting the limitations of MLLMs in analyzing spatial relationships and orientation.

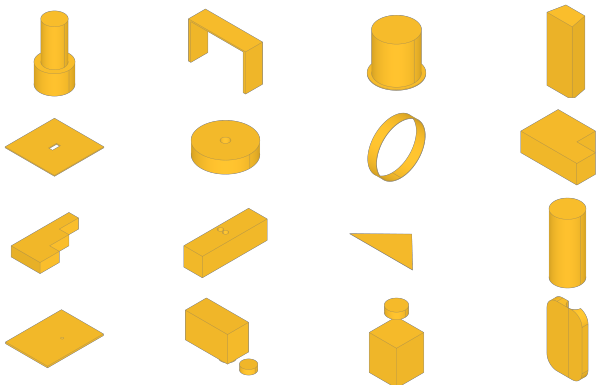
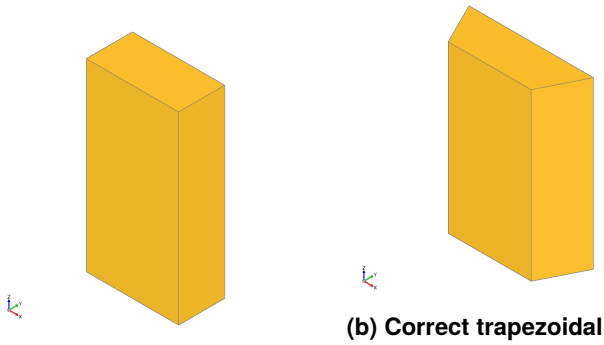


FIGURE 4: CAD MODELS GENERATED BY MEDA AND SUBSEQUENTLY USED FOR EVALUATION

4.2. Ablation Analysis

The quantitative results of the ablation study are shown in Table 2. The inclusion of Script Execution Reviewer in the architecture increased the compilation success rate by 17.5% in a single refinement, leading to an overall execution success rate of 93.5%. A further refinement with the addition of the Script Execution Reviewer further improved this rate, ensuring that 96.5% of the scripts generated were executable. This highlights the crucial role of the Script Execution Reviewer in making scripts executable compared to baseline multi-agent system.

The impact of adding the CAD Image Reviewer for further CAD refinement is shown in Figure 5. Given the prompt to create a trapezoidal prism, the baseline agentic system creates an incorrect CAD model (i.e., a rectangular prism). The CAD Image Reviewer identifies this discrepancy between the prompt and the generated CAD model and provides feedback for correct CAD model generation. By providing this feedback to the multi-agent system, the CAD Image Reviewer plays an important role in further improving the CAD model and thus the metric upon successful refinement of the generated CAD model as shown in Table 3.



(a) Incorrect CAD generated by baseline system

(b) Correct trapezoidal prism after refinement with CAD Image Reviewer

(c) Cylinder created without a semi-circular cut at the middle

(d) Cylinder with required cut after refinement by CAD Image Reviewer

FIGURE 5: ABLATION STUDY RESULTS

Additionally, Fig. 6 shows the average cost per design prompt when using GPT-4o during the ablation study. The cost associated with using CAD Image Reviewer for CAD refinement is more than seven times higher than that of the baseline system. While integrating CAD Image Reviewer significantly improved the CAD evaluation metrics, it also led to substantial increase in cost. This trade-off between performance gains and increased cost highlights the need of a careful consideration when implementing MLLM-based refinement in real-world applications.

TABLE 2: ABLATION STUDY EVALUATING THE BASELINE MULTI-AGENTIC SYSTEM, SCRIPT EXECUTION REVIEWER (SINGLE AND DOUBLE REFINEMENT), AND CAD IMAGE REVIEWER

Condition	Point Cloud dist. ↓	Hausdorff dist. ↓	IoGT ↑	Compile Rate ↑
Baseline Multi-Agent System (No Feedback)	0.0650 (0.0998)	0.321 (0.4147)	0.9407 (0.0325)	76%
+ Script Execution Reviewer (Single Refinement)	0.0616 (0.098)	0.3107 (0.4078)	0.9411 (0.0323)	93.5%
+ Script Execution Reviewer (Double Refinement)	0.0613 (0.1005)	0.3126 (0.3907)	0.9405 (0.0273)	96.5%
+ CAD Image Reviewer (Final Evaluation)	0.0555 (0.095)	0.2628 (0.401)	0.9413 (0.0275)	99%

TABLE 3: EVALUATION METRICS FOR MODELS IN FIGURE 5 BEFORE AND AFTER REVIEW PROCESS

CAD Models	Figure	PCD ↓	HD ↓	IoGT ↑
Trapezoidal prism (before review)	Figure 5a	0.051	0.16	0.948
Trapezoidal prism (after review)	Figure 5b	0.029	0.064	0.951
Cylinder with cut (before review)	Figure 5c	0.0199	0.103	0.94
Cylinder with cut (after review)	Figure 5d	0.0195	0.0997	0.941

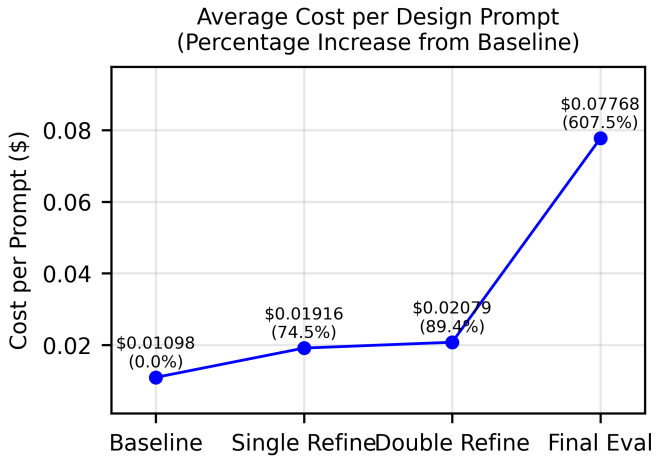


FIGURE 6: AVERAGE COST PER DESIGN PROMPT FOR 200 CAD DESIGN PROMPTS

4.3. Detailed MEDA Examples

Despite the performance observed in the previous section, successful CAD refinement is not always achieved due to the spatial reasoning challenges that are inherent to LLMs. In this section, we present two detailed examples of parametric CAD model creation using the full MEDA framework. In Example 1, we show how MEDA successfully refines the CAD model to achieve successful results. In Example 2, we examine a failure case.

Example 1: Successful CAD Creation of Rectangular pipe. Figure 7 shows the step-by-step process that MEDA follows when creating a parametric model successfully. The process begins when a prompt is passed to the Design Expert for planning. The specific prompt used in this case is:

> *Write Python code using CADQuery to create a rectangular pipe. Begin by creating a sketch of a square with dimensions 0.23077 by 0.23077 units in the X-Z plane and extrude it 0.75 units along the Y-axis. Next, create a sketch of a slightly smaller square, specifically 0.18462 by 0.18462 units (calculated by subtracting twice the inside padding of 0.023077 units from each dimension), centered at the same point as the original square. Extrude this smaller square negatively along the Y-axis by 0.75 units to create a hole in the original extrusion.*[13]

This design plan is first passed to the CAD Script Writer for code writing. Once the script is written by the CAD script writer, it is executed by the Executor resulting in an output log. In this case, as the execution is successful, the Script Execution Reviewer escalates to the CAD Image Reviewer for validation of the generated CAD model. Upon reviewing the generated image against the user prompt, the CAD Image Reviewer finds discrepancies.

In this example, the CAD Image Reviewer correctly identifies the missing blind cut in the block compared to the requirements in the prompt. It, then, passes this failure message to the Design Expert for creating a new design plan. The process of CAD script writing and review is repeated until the output passes checks by

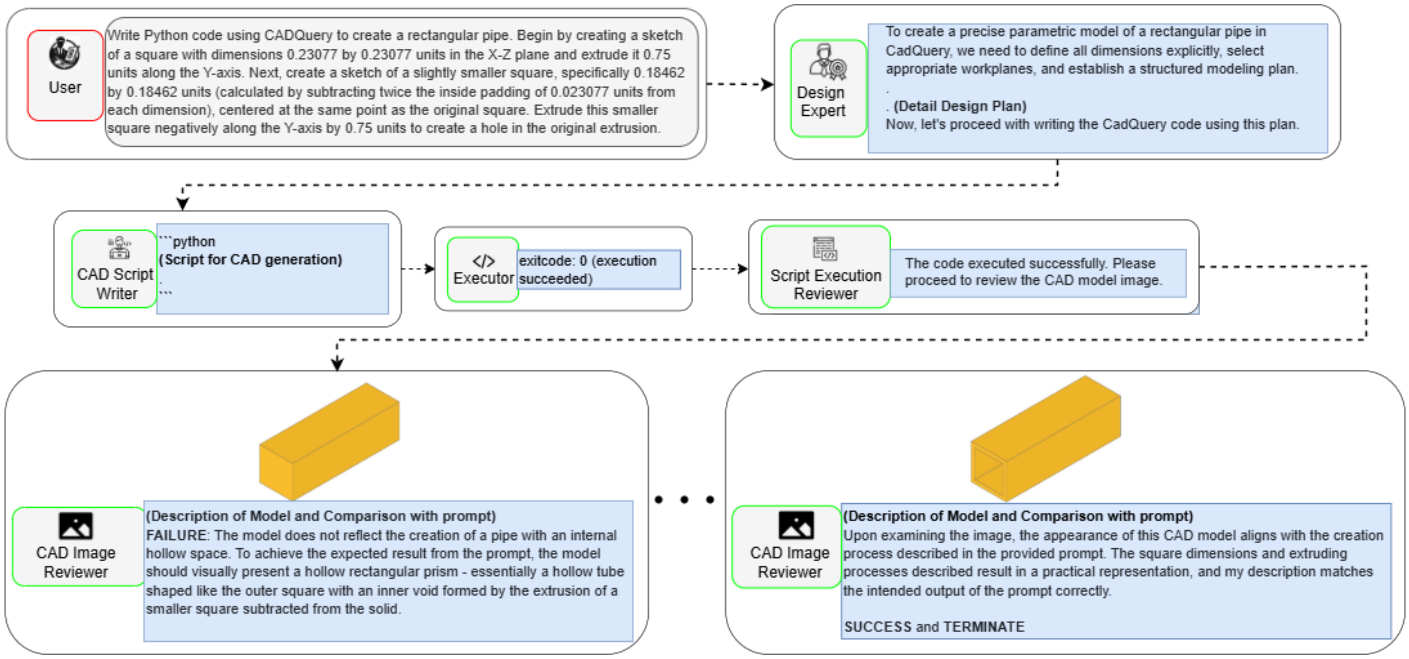


FIGURE 7: SUCCESSFUL CAD CREATION WITH MEDA

both the Script Execution Reviewer and the CAD Image Reviewer. Ultimately, after appropriate refinement, MEDA correctly creates the rectangular pipe, as shown in the final step.

Example 2: Failure case with MEDA. For the given prompt to create two perpendicularly connected rectangles, MEDA failed to create the correct model even after multiple refinements. The specific prompt used is:

> *Write Python code using CADQuery to create a 3D model consisting of two extruded rectangles, each with a length of 0.0625 units, a width of 0.75 units, and a height of 0.0034 units. These rectangles should be connected at their edges, similar to the intersection between a wall and the floor, and positioned perpendicular to each other. The connection should occur along their elongated edge, forming a right angle between the two rectangles.* [13]

The detailed workflow followed by MEDA is shown in Figure 8. In the first review, the CAD Image Reviewer identifies that the model is incorrect with respect to the prompt, and thus further refinement is required. However, its specific reasoning for the error is imperfect, as it does not appropriately interpret the isometric angle of the drawing. Given the failure case, the task is escalated to the Design Expert to resolve the discrepancies and correct the CAD model. However, successive refinements were unable to correct the output. This example illustrates some of the failure modes that are possible in MEDA and other systems that use MLLMs, suggesting iterative refinement and improvement in future work.

5. LIMITATIONS AND FUTURE WORK

In this work, we have evaluated our multi-agent framework using only one MLLM- GPT-4o model. While the multi-agent framework shows improved performance with this MLLM, further experiments using various commercial and open-source models, along with random seeds, can provide more insights into the effectiveness of the multi-agent approach. Furthermore, agents powered by different LLMs (smaller or fine-tuned) may be more effective than a single MLLM-based framework. For example, Design Expert and CAD Script Writer could benefit from LLMs trained specifically on parametric operation sequences for improved CAD model generation. Additionally, since the output of the agents can be sensitive to even minor changes in their system messages, further investigations could be carried out to identify the most effective system prompt for each agent, even when using different language models.

The framework evaluation is limited to textual design prompts with specified dimensions for consistent comparison with the existing CAD generation framework. We plan to extend the evaluation to a dataset comprising fully defined 2D drawings in order to assess the multi-modal capabilities of the existing MLLMs for CAD model generation. In the current work, our primary focus was on integrating the overall design process into a multi-agent framework, with less emphasis on CAD script knowledge. Future research could explore the impact of incorporating external sources of knowledge on detailed scripting functions and methods through retrieval augmented generation (RAG). Furthermore, expanding MEDA with new specialized agents may further improve the performance of the system.

6. CONCLUSION

We introduce MEDA, a multi-agent architecture that autonomously generates parametric CAD models using a multi-modal large language model as both a generator and as a refiner. This framework enables iterative refinement of CAD outputs based on user input and visual appearance. By evaluating MEDA

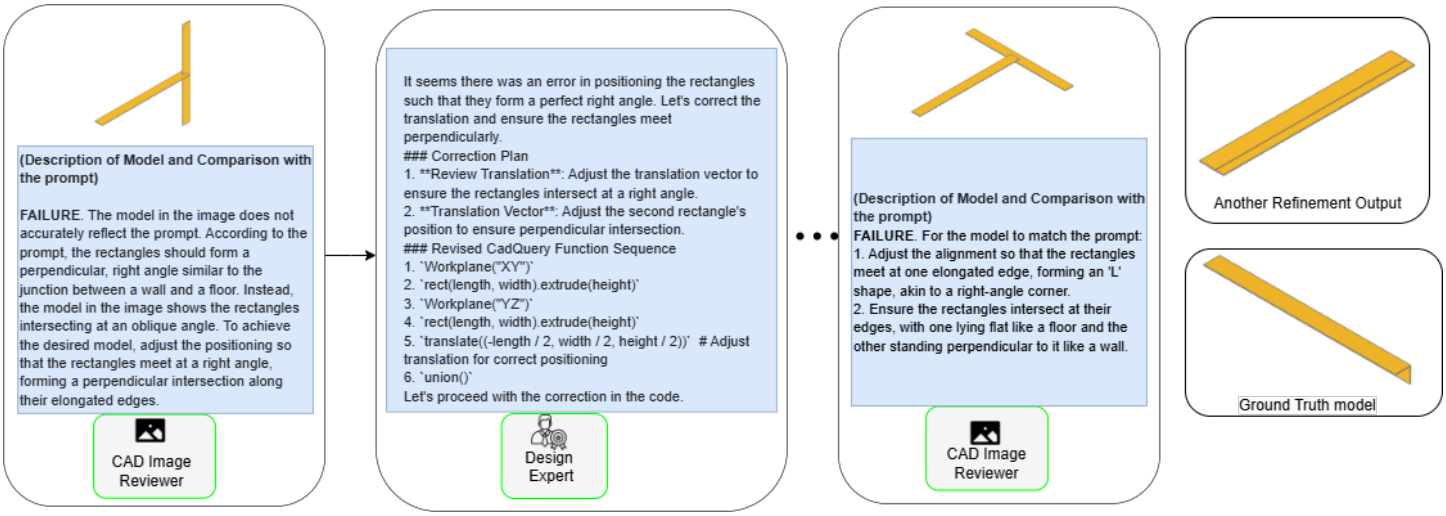


FIGURE 8: EXAMPLE OF FAILURE CASE WITH MEDA

on 200 design prompts, we demonstrate a 99% success rate in CAD model generation, achieving improved quantitative metrics compared to prior approaches. Our findings highlight the effectiveness of multi-agent systems in automating complex problem-solving tasks using zero-shot and one-shot learning, paving the way for broader applications in CAD, CAE, and CAM with integrated error detection and self-refinement processes.

REFERENCES

[1] Schulz, Adriana, Xu, Jie, Zhu, Bo, Zheng, Changxi, Grinspun, Eitan and Matusik, Wojciech. "Interactive design space exploration and optimization for CAD models." *ACM Transactions on Graphics* Vol. 36 No. 4 (2017): pp. 1–14. DOI 10.1145/3072959.3073688.

[2] Contero, Manuel, Pérez, David, Company, Pedro and Camba, Jorge D. "A quantitative analysis of parametric CAD model complexity and its relationship to perceived modeling complexity." *Advanced Engineering Informatics* Vol. 56 (2023): pp. 101970–101970. DOI 10.1016/j.aei.2023.101970.

[3] "New Perspectives on Design Automation: Celebrating the 40th Anniversary of the ASME Design Automation Conference." *Journal of Mechanical Design* Vol. 137 No. 5 (2015). DOI 10.1115/1.4030256.

[4] Taico-Valverde, Bryan and Castillo-Sosa, Misael. "Integration of parametric design tools with artificial intelligence in the construction industry - a review." *International Journal of Educational Practices and Engineering*. Vol. 1 No. 1 (2024): pp. 12–24. DOI 10.70504/ijepe.v1i1.11009.

[5] Zhao, Yaoyao Fiona, Niforatos, Evangelos, Custis, Tonya, Lu, Yan and Luo, Jianxi. "Special Issue: Large Language Models in Design and Manufacturing." *Journal of Computing and Information Science in Engineering* Vol. 25 No. 2 (2025): p. 020301. DOI 10.1115/1.4067319. URL https://asmedigitalcollection.asme.org/computingengineering/article-pdf/25/2/020301/7424180/jcise_25_2_020301.pdf, URL https://doi.org/10.1115/1.4067319.

[6] *Conceptual Design Generation Using Large Language Models*, Vol. Volume 6: 35th International Conference on Design Theory and Methodology (DTM) of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (2023). DOI 10.1115/DETC2023-116838. URL https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2023/87349/V006T06A021/7061719/v006t06a021-detc2023-116838.pdf, URL https://doi.org/10.1115/DETC2023-116838.

[7] Achiam, Josh, Adler, Steven, Agarwal, Sandhini, Ahmad, Lama, Akkaya, Ilge, Aleman, Florencia Leoni, Almeida, Diogo, Altenschmidt, Janko, Altman, Sam, Anadkat, Shyamal et al. "GPT-4 Technical Report." (2024). URL 2303.08774, URL https://arxiv.org/abs/2303.08774.

[8] Anthropic. "Claude 3.7 Sonnet and Claude Code." Anthropic. Accessed March 9, 2025, URL https://www.anthropic.com/news/claude-3-7-sonnet.

[9] Ezemba, Jessica, McComb, Christopher and Tucker, Conrad. "Simulation vs. Hallucination: Assessing Vision-Language Model Question Answering Capabilities in Engineering Simulations." *Proceedings of the 7th Workshop on Design Automation for CPS and IoT*. 2025. Association for Computing Machinery, New York, NY, USA. DOI 10.1145/3722573.3727826. URL https://doi.org/10.1145/3722573.3727826.

[10] Gosmar, Diego and Dahl, Deborah A. "Hallucination Mitigation using Agentic AI Natural Language-Based Frameworks." (2025). URL 2501.13946, URL https://arxiv.org/abs/2501.13946.

[11] Li, Xingang, Sun, Yuewan and Sha, Zhenghui. "LLM4CAD: Multimodal Large Language Models for Three-Dimensional Computer-Aided Design Generation." *Journal of Computing and Information Science in Engineering* Vol. 25 No. 2 (2024): p. 021005. DOI 10.1115/1.4067085. URL https://asmedigitalcollection.asme.org/computingengineering/

article-pdf/25/2/021005/7414486/jcise_25_2_021005.pdf, URL https://doi.org/10.1115/1.4067085.

[12] Sun, Yuewan, Li, Xinggang and Sha, Zhenghui. "Large Language Models for Computer-Aided Design (LLM4CAD) Fine-Tuned: Dataset and Experiments." *Journal of Mechanical Design* (2025): pp. 1–19DOI 10.1115/1.4067713.

[13] Alrashedy, Kamel, Tambwekar, Pradyumna, Zaidi, Zulfiqar, Langwasser, Megan, Xu, Wei and Gombolay, Matthew. "Generating CAD Code with Vision-Language Models for 3D Designs." *arXiv preprint arXiv:2410.05340* (2025).

[14] Ocker, Felix, Menzel, Stefan, Sadik, Ahmed and Rios, Thiago. "From Idea to CAD: A Language Model-Driven Multi-Agent System for Collaborative Design." *arXiv preprint arXiv:2503.04417* (2025).

[15] Mathur, Aman, Pirron, Marcus and Zufferey, Damien. "Interactive Programming for Parametric CAD." *Computer Graphics Forum* Vol. 39 No. 6 (2020): pp. 408–425. DOI 10.1111/cgf.14046.

[16] Camba, Jorge D, Contero, Manuel and Company, Pedro. "Parametric CAD modeling: An analysis of strategies for design reusability." *Computer-aided design* Vol. 74 (2016): pp. 18–31.

[17] Yussuf, Naglaa, Maarouf, Ibrahim and Abdelhamid, Mona M. "Parametric-based Approach in Architectural Design Procedures." *Fayoum University Journal of Engineering* Vol. 7 No. 2 (2024): pp. 127–133. DOI 10.21608/fuje.2024.343806.

[18] Niphadkar, Pooja A. and Niphadkar, Ajinkya. "Strategies for Parametric Design in Architecture." *International Journal of Advanced Research in Science Communication and Technology* (2022): pp. 115–119DOI 10.48175/ijarsct-3774.

[19] Fan, Rubin, He, Fazhi, Liu, Yuxin, Song, Yupeng, Fan, Linkun and Yan, Xiaohu. "A parametric and feature-based CAD dataset to support human-computer interaction for advanced 3D shape learning." *Integrated Computer-Aided Engineering* (2024): pp. 1–22DOI 10.3233/ica-240744.

[20] Wu, Rundi, Xiao, Chang and Zheng, Changxi. "Deepcad: A deep generative network for computer-aided design models." *Proceedings of the IEEE/CVF International Conference on Computer Vision*: pp. 6772–6782. 2021.

[21] Wu, Sifan, Khasahmadi, Amir, Katz, Mor, Jayaraman, Pradeep Kumar, Pu, Yewen, Willis, Karl and Liu, Bang. "Cad-llm: Large language model for cad generation." *Proceedings of the neural information processing systems conference. neurIPS*. 2023.

[22] Koch, Sebastian, Matveev, Albert, Jiang, Zhongshi, Williams, Francis, Artemov, Alexey, Burnaev, Evgeny, Alexa, Marc, Zorin, Denis and Panozzo, Daniele. "ABC: A Big CAD Model Dataset For Geometric Deep Learning." *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[23] Khan, Mohammad Sadil, Sinha, Sankalp, Sheikh, Talha Uddin, Stricker, Didier, Ali, Sk Aziz and Afzal, Muhammad Zeshan. "Text2CAD: Generating Sequential CAD Models from Beginner-to-Expert Level Text Prompts." (2024). URL 2409.17106, URL https://arxiv.org/abs/2409.17106.

[24] Ma, Weijian, Xu, Minyang, Li, Xueyang and Zhou, Xiangdong. "Multicad: Contrastive representation learning for multi-modal 3d computer-aided design models." *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*: pp. 1766–1776. 2023.

[25] Makatura, Liane, Foshey, Michael, Wang, Bohan, Hähn-Lein, Felix, Ma, Pingchuan, Deng, Bolei, Tjandrasuwita, Megan, Spielberg, Andrew, Owens, Crystal Elaine, Chen, Peter Yichen, Zhao, Allan, Zhu, Amy, Norton, Wil J, Gu, Edward, Jacob, Joshua, Li, Yifei, Schulz, Adriana and Matusik, Wojciech. "How Can Large Language Models Help Humans in Design and Manufacturing?" (2023). URL 2307.14377, URL https://arxiv.org/abs/2307.14377.

[26] Picard, Cyril, Edwards, Kristen M., Doris, Anna C., Man, Brandon, Giannone, Giorgio, Alam, Md Ferdous and Ahmed, Faez. "From Concept to Manufacturing: Evaluating Vision-Language Models for Engineering Design." (2024). URL 2311.12668, URL https://arxiv.org/abs/2311.12668.

[27] Yuan, Zeqing, Lan, Haoxuan, Zou, Qiang and Zhao, Junbo. "3D-PreMise: Can Large Language Models Generate 3D Shapes with Sharp Features and Parametric Control?" (2024). URL 2401.06437, URL https://arxiv.org/abs/2401.06437.

[28] Badagabettu, Akshay, Yarlagadda, Sai Sravan and Farimani, Amir Barati. "Query2cad: Generating cad models using natural language queries." *arXiv preprint arXiv:2406.00144* (2024).

[29] Naveed, Humza, Khan, Asad Ullah, Qiu, Shi, Saqib, Muhammad, Anwar, Saeed, Usman, Muhammad, Akhtar, Naveed, Barnes, Nick and Mian, Ajmal. "A comprehensive overview of large language models." *arXiv preprint arXiv:2307.06435* (2023).

[30] Talebirad, Yashar and Nadiri, Amirhossein. "Multi-agent collaboration: Harnessing the power of intelligent llm agents." *arXiv preprint arXiv:2306.03314* (2023).

[31] Bang, Yejin, Cahyawijaya, Samuel, Lee, Nayeon, Dai, Wenliang, Su, Dan, Wilie, Bryan, Lovenia, Holy, Ji, Ziwei, Yu, Tiezheng, Chung, Willy et al. "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity." *arXiv preprint arXiv:2302.04023* (2023).

[32] Sun, Chunyi, Han, Junlin, Deng, Weijian, Wang, Xinlong, Qin, Zishan and Gould, Stephen. "3d-gpt: Procedural 3d modeling with large language models." *arXiv preprint arXiv:2310.12945* (2023).

[33] Lu, Jiaxing, Li, Heran, Ning, Fangwei, Wang, Yixuan, Li, Xinze and Shi, Yan. "Constructing Mechanical Design Agent Based on Large Language Models." *arXiv preprint arXiv:2408.02087* (2024).

[34] AU, Wright, Jeremy, thebluedirt, Boyd, Marcus, Lorenz, Solutions, Innovations Technology, ÖZDERYA, Hasan Yavuz, Agostini, Bruno, Jojain, Greminger, Michael, Fischer, Seth, Buchanan, Justin, cactrot, huskier,

Ruben, iulianOnofrei (U-lee aan), de León Peque, Miguel Sánchez, Budden, Martin, Hecatron, Boin, Peter, Saville, Wink, Penev, Pavel M., Weissinger, Bryan, Christoforo, M. Greyson, Case, Jack, AGD, Jurczak, Paul, nopria, moeb and jdegenstein. "CadQuery/cadquery: CadQuery 2.4.0." (2024). DOI 10.5281/zenodo.10513848. URL https://doi.org/10.5281/zenodo.10513848.

[35] Fulman, Nir, Memduhoğlu, Abdulkadir and Zipf, Alexander. "Distortions in Judged Spatial Relations in Large Language Models." *The Professional Geographer* (2024): pp. 1–9DOI 10.1080/00330124.2024.2372792.

[36] Wu, Qingyun, Bansal, Gagan, Zhang, Jieyu, Wu, Yiran, Li, Beibin, Zhu, Erkang, Jiang, Li, Zhang, Xiaoyun, Zhang, Shaokun, Liu, Jiale, Awadallah, Ahmed Hassan, White, Ryen W, Burger, Doug and Wang, Chi. "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation." (2023). URL 2308.08155, URL https://arxiv.org/abs/2308.08155.

[37] Microsoft Corporation. "Azure OpenAI Service Pricing." (2025). Accessed May 10, 2025, URL https://azure.microsoft.com/en-us/pricing/details/cognitive-services/openai-service/.

[38] Alibekov, Ulugbek, Staderini, Vanessa, Ramachandran, Geetha, Schneider, Philipp and Antensteiner, Doris. "Evaluation of 3D Point Cloud Distances: A Comparative Study in Multi-Point Cloud Fusion Environments." *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*: pp. 59–71. 2024. INSTICC, SciTePress. DOI 10.5220/0012421300003660.

[39] Huttenlocher, Daniel P, Klanderman, Gregory A. and Rucklidge, William J. "Comparing images using the Hausdorff distance." *IEEE Transactions on pattern analysis and machine intelligence* Vol. 15 No. 9 (1993): pp. 850–863.