Proceedings of the Design Society, Volume 5: ICED25
https://doi.org/10.1017/pds.2025.10203

From text to design: a framework to leverage LLM agents for automated CAD generation

Aurel Schüpbach [1], **Raul San Miguel** [1]✉, **Julian Ferchow** [1] **and Mirko Meboldt** [2]

[1]*Inspire AG, Switzerland,* [2]*ETH Zürich, Switzerland*

✉ raul.sanmiguel@inspire.ch

ABSTRACT: Design generation using traditional Computer-Aided Design (CAD) tools remains a labor-intensive and manual task. This paper introduces a framework for automating CAD geometry generation using Large Language Models (LLMs) with function calling and agent workflows. The framework enables both expert and novice designers to use textual prompts to automatically generate CAD code. We evaluate it with five LLMs and four agent workflows. The agent workflow incorporating automated visual feedback outperforms the others, especially with multimodal LLMs like ChatGPT-4o. A case study shows its use in topology optimization and additive manufacturing with minimal human input. Remaining challenges include limitations in spatial reasoning, prompt dependency, and workflow adaptability. Future work should focus on improving design-for-manufacturing capabilities, visual tools, and evaluation benchmarking.

KEYWORDS: artificial intelligence, large language models (LLMs), LLM agents, computer aided design (CAD), computational design methods

1. Introduction

The product development process requires the profound expertise of professionals such as designers, simulation specialists, and tool makers. A constant exchange of information between these roles is essential for the successful development of a new product. Within this process, the step of design generation using traditional Computer-Aided Design (CAD) tools remains a labor-intensive and manual task. This is partially due to the steep learning curves and intricate user interfaces of existing CAD tools, which present significant barriers to entry.

Consequently, there is a critical need for new tools to streamline design generation and provide substantial support to design engineers. There are different approaches in the field of automated design tools that offer this kind of support. Design automation methodologies like algorithm-based design, knowledge-based engineering and parametric modeling are widely used for generating adaptable mechanical parts within narrowly defined constraints (Biedermann et al., 2021), (Cagan et al., 2005). Topology optimization and numerical simulations have also emerged as standard practices for iterative design refinement (Zhu et al., 2021), (Alshare et al., 2019). Generative design represents an advanced methodology that integrates simulation and topology optimization methods with machine learning algorithms (Oh et al., 2019). However, existing automated design approaches exhibit critical limitations. They tend to focus on specific design tasks and are hard to transfer into other design domains. They also require specific knowledge about the requirements of each design task, representing a hurdle for novice designers. These constraints underscore the imperative need for a design tool that offers intuitive usability for designers with varying levels of expertise, broad transferability across multiple design domains, and reduced dependency on specialized domain-specific knowledge.

The rapid advancements in Large Language Models (LLMs) present a transformative opportunity to allow users to expand their design capabilities. Rather than relying on graphical interfaces or complex programming, users can convey their design intent through simple textual inputs, which LLMs interpret and translate into CAD code (Makatura et al., 2023). This approach simplifies the design process for

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

both novice and experienced engineers, offering both broad domain transferability and streamlined incorporation of domain-specific knowledge. Current research strategies predominantly utilize a single LLM and depend on lengthy conversations with the human user (Nelson et al., 2023). Alternative methodologies build iterative loops around LLMs to streamline and automatize the design process (Yuan et al., 2024). Some research provides end-to-end case studies using primarily manual workflows (Makatura et al., 2023). These studies don't provide a highly automated framework in which LLMs can tackle problems via various approaches. They also tend to employ a single LLM, and the case studies require considerable human intervention. Therefore, the research gap lies in the need for a highly automated design framework that integrates diverse LLM approaches and models. This should be accompanied by a comprehensive end-to-end case study with minimal human input.

In this work, we present a novel framework leveraging function calling and agent workflows for automating Computer-Aided Design (CAD) geometry generation. This framework allows LLMs to automatically interpret intricate design requirements from textual prompts and generate the necessary CAD code for design realization. We evaluate its performance with five LLMs and four iterative, multi-step approaches known a as agent workflows. Additionally, we demonstrate the framework's potential through a case study involving topology optimization and additive manufacturing with minimal human input. Consequently, this paper provides three core contributions:

- We propose a new framework that gives LLMs varying levels of agency and access to tools and information through function calling and agent workflows.
- We quantitively compare a set of geometries, incorporating various LLMs and agent workflows.
- We evaluate the framework for different LLMs via an end-to-end case study.

2. Background

This section describes related work in the field of Diffusion Models, LLMs in CAD, and LLM agents.

2.1. Diffusion models in CAD

Diffusion models, successful in image generation, have been adapted for 3D geometry creation. These models, like DreamFusion (Poole et al., 2022), MVDream (Shi et al., 2023), MicroDreamer (Chen et al., 2024) and DreamFlow (Lee et al., 2024) can generate visually compelling 3D scenes and textured models. However, their ability to produce exact CAD models from engineering constraints is limited, and they lack typical features like design history or CAD interoperability. More recent advancements have introduced diffusion models specifically for CAD geometries, such as VQ-CAD (H. Wang et al., 2024) and BrepGen (Xu et al., 2024), which directly output Boundary representations (B-rep) geometries. Trained on large CAD datasets like DeepCAD (R. Wu et al., 2021) or the ABC Dataset (Koch et al., 2019), these models can generate convincing CAD geometries, but they still struggle to adhere to precise requirements such as dimensional constraints or accurate positions of features.

2.2. LLMs in CAD

Employing LLMs for geometry generation represents an emerging and promising approach currently in its early stages. Challenges are posed by model hallucination, a phenomenon where the model fills knowledge gaps with fabricated information, and by the limited availability of specialized engineering training data for LLMs (Rohrbach et al., 2019). Additionally, since current LLMs primarily rely on text input and output, leveraging their capabilities for broader applications requires new approaches. One such approach is to enable LLMs to execute function calls, such as database querying or code execution (Zhuang et al., 2023), (Hao et al., 2023). This capability allows models to take actions or gather information to further inform responses. In the context of CAD, functions have been used in literature to generate code that is executed by CAD software to create geometries, allowing the LLM to perform geometry generation while dealing exclusively with text (Nelson et al., 2023). Existing work has predominantly utilized ChatGPT-4 and typically requires substantial human intervention. To solve more complex, multi-step tasks, agent-based methods like AutoGen (Q. Wu et al., 2023) have been proposed, where an LLM follows different workflows that delineate how it can autonomously use certain tools to solve problems. This approach has been used in the field of engineering design to automatically

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

generate, execute and refine CAD code (Alrashedy et al., 2024) and to improve CAD code generation by feeding a multimodal LLM an image of the rendered geometry (Yuan et al., 2024).

2.3. LLM agents

LLM agents use LLMs as a tool within a larger system; the LLM is not treated as a black box that will perform any kind of task, but rather as a creative interface mediating between structured inputs and targeted outputs within a precisely defined process. This system can be rigorously defined by giving the LLM a description of what its role is, such as travel agent or programmer. This description also comprehensively outlines the resources and tools at the LLM's disposal (Xi et al., 2023). These include information repositories like documentation and databases alongside a set of functions. These functions enable the LLM to access and employ resources external to its training or given data and to perform tasks beyond text generation. For example, an LLM by itself cannot perform mathematics, but strategically implemented functions can enable calculation capabilities through integration with frameworks like Python. To keep track of the flow of information to and from the LLM, conversation log file allows the LLM to reconstruct the contextual trajectory of previous interactions, typically facilitated through an application programming interface (API) (L. Wang et al. 2024).

3. Framework

This section provides a comprehensive overview of the proposed framework. for CAD geometry generation. This novel framework is based on a set of agent workflows and allows a user to give a prompt describing a geometry to an LLM. The LLM interprets the prompt and generates the CAD code necessary to automatically build that geometry in a CAD program. The agent workflows define the resources and tools available to the LLM, outline the approach the LLM will follow, and specify the level of interaction between the LLM and the user.

3.1. Geometry generation framework

The flowchart in Fig. 1 visualizes the framework. The user launches the geometry generation process by submitting a prompt which is either custom-made or selected from a predefined set (Step 1). The prompt is subsequently fed to the LLM along with predefined context, including the system description and function definitions (detailed in Section 3.1.2). Even though it is understood by the system that the prompt will deal with the creation of geometries, it remains unconstrained, allowing for potentially tangential or unrelated inputs. The agent workflow is then selected from among four variants (Step 2) to determine the LLM's input processing and CAD code generation strategy (discussed in Section 3.1.3). The agent workflow orchestrates input generation and transmission to the LLM, with the model generating responses accordingly (Step 3). Contingent on the choice of agent workflow, the LLM API may necessitate multiple iterative calls. All inputs to and outputs from the LLM are recorded in a conversation log, enabling contextual continuity across API calls. After the LLM has finalized generating the CAD code, the function calls are extracted from the log file into a dedicated function call file (Step 4). This file then serves as the basis for geometry assembly in Grasshopper, the visual programming interface of Rhino3D.

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

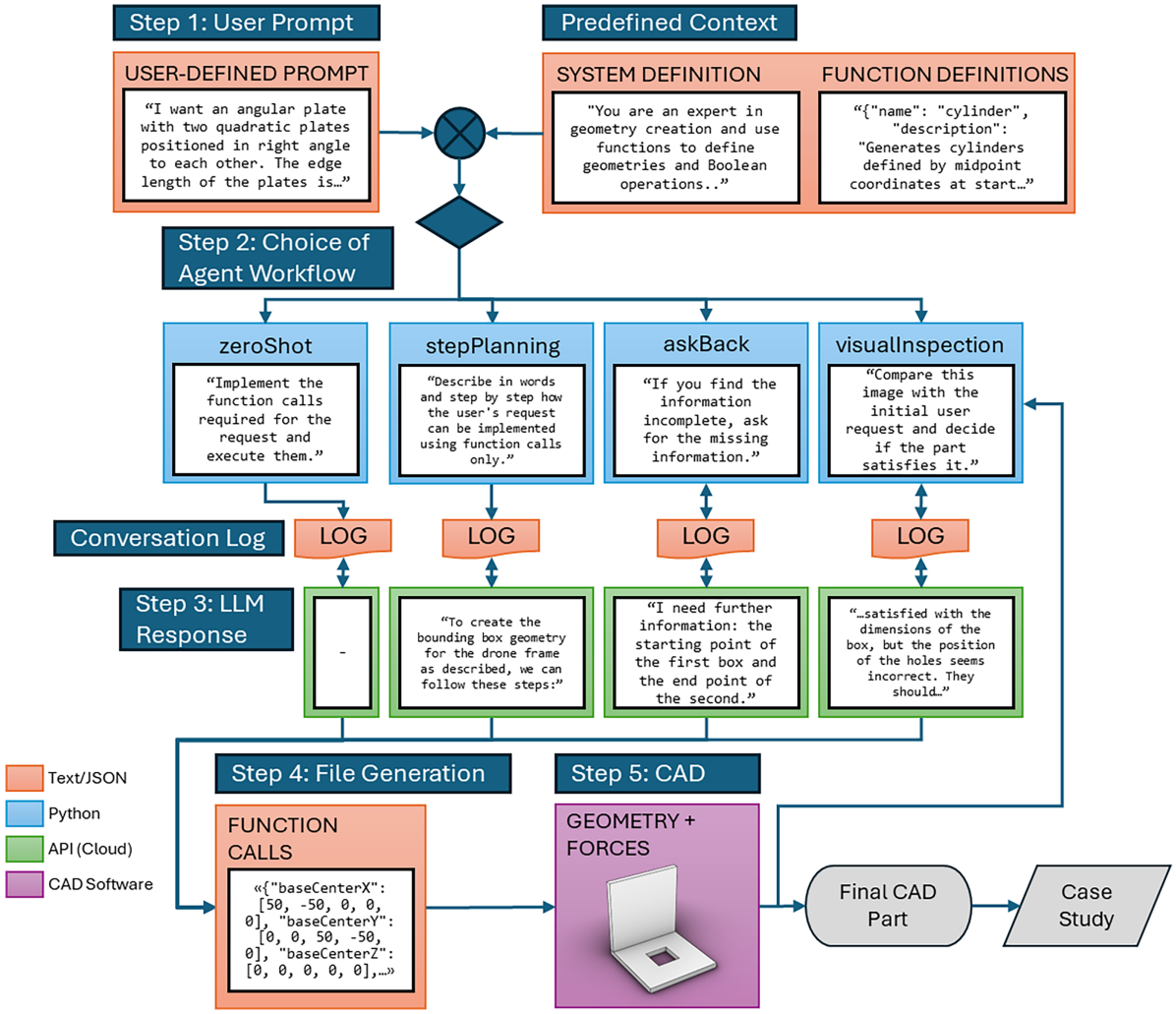


Figure 1. Framework flowchart. The prompts, predefined context and function call file are preliminarily stored as text or JSON files locally, the agent workflows are implemented in Python and run locally, and the LLM is accessed vi an API. In this manner, the LLM is a tool within the larger framework and is employed only when required by each agent workflow's strategy

3.2. Predefined context: system and function definitions

The predefined context including the system description and function definitions serves to delineate the operational boundaries of the LLM and provide an inventory of the functions available for geometry creation. The system is described as having expertise in geometry generation through the combination of primitive objects through Boolean operations in a process known as constructive solid geometry. The function definitions are structured in JSON format, with textual annotations for each function definition and each function variable. The following function definitions are implemented:

1. Block: This function instantiates a block primitive from a basepoint and the dimensions of each of the sides. It allows for Boolean addition and subtraction.
2. Cylinder: This function instantiates a cylinder primitive from a midpoint, a radius, and an axial vector. It allows for Boolean addition and subtraction.
3. Code Generation: This function allows the LLM to execute generated Python code locally. The output or error message is returned to the LLM, allowing it to debug code itself. Intended for geometric operations such as patterning, mirroring, or the calculation of distances.
4. Force: This function defines a force by its magnitude and direction vector.
5. Fixtures: This function accepts an array of bounding geometry primitives, either as fixture or force application regions, for topology optimization or other simulation-based design processes.

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

6. Finished: This function serves as a signal from the LLM to terminate the CAD code generation sequence, triggering the generation of the function call file and the subsequent assembly of the geometry.

3.3. LLM agent workflows

The LLM agent workflows govern the LLM's input processing and CAD code generation strategy, managing input generation and transmission to the LLM as well as output processing. The proposed framework encompasses four distinct agent workflows:

7. zeroShot: This baseline agent workflow submits the user's textual prompt to the LLM with a message instructing the model to return function calls for geometry generation. After each function call, a message is injected prompting the next function call until the LLM returns the terminal *Finished* function, signaling the end of the CAD code generation process.
8. stepPlanning: This agent workflow instructs the LLM to first generate step-by-step instructions for geometry creation, including the function calls required. These instructions are then converted into executable function calls as in the zeroShot agent workflow.
9. askBack: This agent workflow enables the LLM to engage in dialogue with the user to solicit additional information if it recognizes that the user prompts might lack clarity or completeness. After the LLM deems it has achieved a sufficiently unambiguous understanding, the agent workflow proceeds with the planning and geometry generation stages as in the stepPlanning agent workflow.
10. visualInspection: This agent workflow adds a visual inspection stage by leveraging the multimodal capabilities of LLMs such as ChatGPT-4o. After the geometry generation is performed following the same procedure as in the askBack agent workflow, an image of the rendered geometry is shared with the LLM for assessment. If discrepancies are detected, the LLM generates an improved step-by-step plan, which is then recursively fed into the geometry generation process.

4. Results

This section critically analyses the results of the proposed framework and conducts an end-to-end case study to highlight its potential.

4.1. Framework evaluation

The efficacy of our workflow was assessed by a set of test geometries with various levels of complexity and ambiguity. Each geometry creation was called with three distinct textual prompts (see Appendix A), deliberately constructed to assess the LLM's capability to comprehend and interpret design intent from diverse descriptions. The geometries in Fig. 2a represent the desired output of the prompts as a human designer would interpret and create them. The quality of the generated parts was assessed using a binary success/failure visual inspection method. A geometry was deemed successful only if the part was generated, complete and accurate. Fig. 2b illustrates an example of the criteria applied to the corner plate. The test geometries were chosen to evaluate specific capabilities of the LLMs:

a) box: This geometry serves as a baseline, testing the model's ability to interpret basic shapes and perform a straightforward Boolean subtraction operation.
b) corner plate: This geometry assessed the LLM's spatial understanding of a “right angle” and its ability to correctly orient and position two plates to form a corner. A square cutout was specified to be centrally located on one of the plates, intentionally without additional specifications.
c) u-profile: By providing only wall thickness and outer dimensions, this geometry evaluated the model's comprehension of simple shape descriptors such as “u-shaped.”
d) toy car: This complex geometry assessed the LLM's capacity to create intricate shapes with deliberately imprecise specifications. The prompts for the wheel position and cutout for the windshield focused on relative positioning rather than precise dimensions.

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

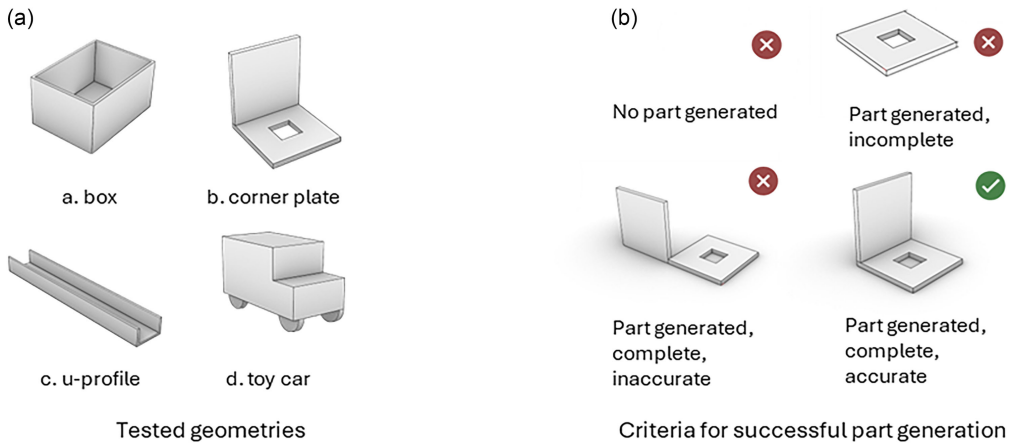


Figure 2. Test prompt grounds truth CAD models and example of corner plate geometry generation

The performance of the different agent workflows was tested with ChatGPT-4 Turbo along with the predefined test geometries. ChatGPT-4 Turbo was selected as it was the most advanced and stable multimodal LLM available at the time of the experiments. This allowed all agent workflows, including visualInspection, to be compared. As depicted in Table 1, the askBack agent workflow, designed for user clarifications, yielded the lowest success rates, although it still produced some accurate geometries at 22/60 successful attempts. In cases where prompts were already well-defined, this mechanism appeared to introduce distractions, potentially impairing subsequent planning and execution. It is notable that the zeroShot agent workflow, which operates without intermediate planning or clarification steps, slightly outperformed the askBack agent workflow with a 26/60 success rate. Similarly, the stepPlanning agent workflow offered marginal improvement, also with 26/60 successful attempts. In contrast, the VisualInspection agent workflow, incorporating an automated visual feedback loop, had a success rate of over 50% for all geometries except the u-profile. This agent workflow enabled the LLM to identify and correct errors effectively, especially in cases like the corner plate, where it successfully generated 11/15 instances, in contrast with the stepPlanning agent workflow's 4/15.

Table 1. Success rate of agent workflows using chatGPT-4 Turbo

	box	corner plate	u-profile	toy car	Total
zeroShot	12/15 (80%)	1/15 (7%)	3/15 (20%)	**10/15 (67%)**	26/60 (43%)
stepPlanning	13/15 (87%)	4/15 (27%)	**6/15 (40%)**	3/15 (20%)	26/60 (43%)
askBack	12/15 (80%)	1/15 (7%)	5/15 (33%)	4/15 (27%)	22/60 (37%)
visualInspection	**14/15 (93%)**	**11/15 (73%)**	4/15 (27%)	8/15 (53%)	37/60 (62%)
Total	**51/60 (85%)**	*17/60 (28%)*	*18/60 (30%)*	*25/60 (42%)*	*111/240 (46%)*

To investigate the influence of the underlying LLMs on geometry generation performance, we compared off-the-shelf models using the stepPlanning agent workflow along with the predefined test geometries. The stepPlanning workflow was chosen because it functioned with both language-only models and multimodal models. Additionally, it is the only agent workflow, apart from the baseline zeroShot agent workflow, that is fully automated, not requiring human input beyond the initial prompt. Five different LLMs were compared, namely Mistral Large, Claude 3 Opus and generations of ChatGPT (ChatGPT-3.5 Turbo, ChatGPT-4 Turbo and ChatGPT-4o). As shown in Table 2, Mistral Large demonstrated the lowest performance with just 1/60 successful attempts, whereas Anthropic's Claude 3 Opus delivered results comparable to ChatGPT-4 Turbo at 27/60 and 26/60 successful attempts respectively. Among the three generations of ChatGPT tested, a significant improvement in performance is evident with each newer version. Notably, the most recently released ChatGPT-4o generally outperformed all other models, exhibiting enhanced spatial awareness and reasoning capabilities. However, it consistently failed to generate the right-angle geometry accurately, failing all 15 attempts.

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

Table 2. Success rates of LLMs across test geometries using the stepPlanning agent workflow

	box	corner plate	u-profile	toy car	Total
Mistral Large	0/15 (0%)	0/15 (0%)	1/15 (7%)	0/15 (0%)	*1/60 (2%)*
GPT-3.5 Turbo	3/15 (20%)	0/15 (0%)	0/15 (0%)	0/15 (0%)	*3/60 (5%)*
GPT-4 Turbo	**13/15 (87%)**	4/15 (27%)	6/15 (40%)	3/15 (20%)	*26/60 (43%)*
Claude 3 Opus	**13/15 (87%)**	**6/15 (40%)**	7/15 (47%)	1/15 (7%)	*27/60 (45%)*
GPT-4o	12/15 (80%)	0/15 (0%)	**9/15 (60%)**	**10/15 (67%)**	31/60 (52%)
Total	41/75 (55%)	*10/75 (13%)*	*23/75 (31%)*	*14/75 (19%)*	*88/300 (29%)*

4.2. End-to-end case study

The modular architecture of our framework extends beyond basic geometry creation, enabling the definition of arbitrary variables and constraints to enable the automation of complex design pipelines. To illustrate this capability, we present a use case involving topology optimization of a standard benchmark geometry: the Flight Crew Rest Cabin (FCRC) bracket (Schmidt, 2016). The FCRC bracket connects to the primary aircraft structure and serves as a widely cited reference for topological optimization and additive manufacturing studies. The requirements for this use case are shown in Fig. 3 Unlike previous experiments that mainly focused on geometry creation, the agent workflow now employs the *Forces* and *Fixtures* functions to also define the simulation constraints. This advanced shape is meant to challenge the LLM's capacity to handle patterns and mirroring, potentially leveraging its code generation capabilities. In combination with these inputs, the agent workflow generates a bounding box that defines the design space for subsequent topology optimization.

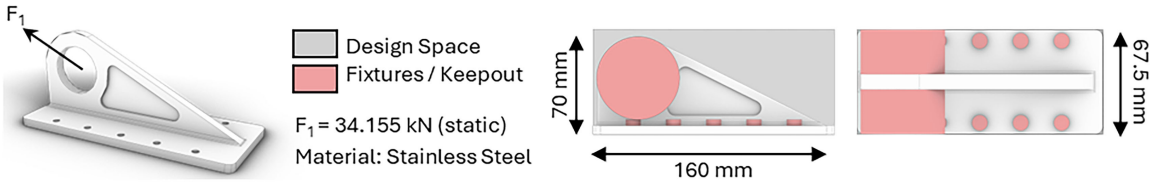


Figure 3. FCRC bracket Benchmark Properties

Within Rhino/Grasshopper, an automated workflow (Fig. 4) utilizes this design space to perform topology optimization based on the SIMP methodology (Bendsøe, 1989) using the tOpos plugin (Bialkowski, 2017). The resulting optimized geometry is automatically cleaned, re-meshed and prepared for manufacturing. In this case study, the part was successfully produced on a Colibrium Concept Laser Mlab Cusing R machine using stainless steel 316L.

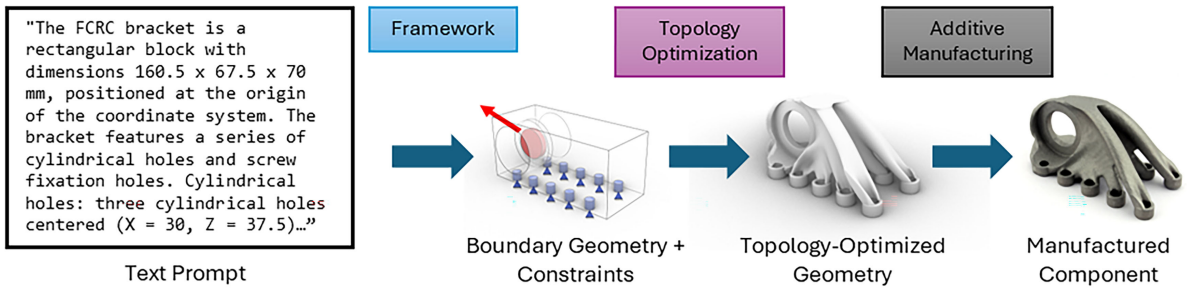


Figure 4. Complete FCRC process from text prompt to final additively manufactured geometry

The performance of the agent workflows in generating the FCRC bracket geometry was evaluated using ChatGPT-4 Turbo, as in the test geometries in Section 3.2. The askBack, stepPlanning, and visualInspection agent workflows outperformed the zeroShot agent workflow (Table 3). However, the visualInspection agent workflow did not achieve improvements as significant as those observed for other test geometries. While visualInspection generated 11/15 successful instances for the corner plate—7 more than the next best workflow (Table 1)—it produced only 5/15 successful instances for the FCRC bracket, outperforming stepPlanning by just 3 instances and askBack by only 2 (Table 3).

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

Table 3. Success rate of agent workflows using ChatGPT-4 Turbo for the FCRC bracket

askBack	zeroShot	stepPlanning	visualInspection	Total
2/15 (13%)	0/15 (0%)	3/15 (20%)	**5/15 (33%)**	10/60 (17%)

The impact of the LLMs on geometry generation was also compared using the stepPlanning agent workflow. The complexity of the FCRC bracket and the additional topology optimization requirements led to a low overall success rate of the geometry generation (Table 4). Less capable LLMs like Mistral Large and ChatGPT-3.5 Turbo completely failed to generate any successful geometries. Similarly, ChatGPT-4 Turbo had a very low success rate at 3/15 successful attempts. Claude 3 Opus offered some improvement at 6/15 successful instances, but only ChatGPT-4o performed well at this task, generating the desired geometry in 10/15 or 67% of instances.

Table 4. Success rates of LLMs using the stepPlanning agent workflow for the FCRC bracket

Mistral Large	GPT-3.5 Turbo	GPT-4 Turbo	Claude 3 Opus	GPT-4o	Total
0/15 (0%)	0/15 (0%)	3/15 (20%)	6/15 (40%)	**10/15 (67%)**	19/75 (25%)

5. Discussion

This paper addresses the need for a highly automated design framework that incorporates diverse LLM approaches and models while demonstrating its application through a comprehensive end-to-end case study. The proposed framework integrates LLMs with function calling and agent workflows to automate CAD geometry generation. Evaluation results highlight the effectiveness of varying levels of agency provided to LLMs through these agent workflows. Notably, the combination of the visualInspection agent workflow and ChatGPT-4o, with its multimodal capabilities, shows potential in enabling a fully automated design pipeline with minimal human intervention. Furthermore, the framework's ability to interpret and generate CAD geometries was validated through the design and manufacturing of a benchmark geometry in an end-to-end case study. Thus, this framework offers both expert and novice designers an intuitive way to convey design intent, reducing their reliance on traditional CAD tools.

Moreover, the framework is both LLM- and CAD-agnostic. Any LLM supporting API-based interaction can be integrated, and once the necessary operations and objects for geometry creation are defined in text, they can be executed in any CAD software with scripting capabilities. This adaptability demonstrates the potential of our framework to automate even complex design tasks and be transferred across diverse design domains, including mechanical studies and topology optimization. Further, the design automation demonstrated potential applications beyond the FCRC bracket. By solely modifying the initial prompt, this framework can generate topology-optimized geometries for a wide range of use cases.

Nevertheless, the framework holds considerable room for improvement. The visualInspection agent workflow is currently limited by its reliance on a single perspective of the generated geometry. Providing multiple views with spatial annotations could enhance the LLM's ability to detect and correct errors. Additionally, introducing a visual interface for user feedback would enable more intuitive corrections by allowing users to point out errors directly, reducing reliance on text-based descriptions. Similarly, augmenting prompts with visual aids, such as sketches or example geometries, could guide the LLM, improve its interpretative accuracy and support user interaction.

Beyond these enhancements, the framework's current focus on geometry creation overlooks key aspects such as manufacturability or the practical application of each geometry. Expanding the agent workflows to include design-for-manufacturing guidelines, validated through case studies for different manufacturing technologies like additive manufacturing, injection molding, and milling, would significantly broaden its utility.

Another area for improvement lies in the evaluation methodology, more specifically in the definition of the inputs provided to the LLM, which should be made more robust. Prompts should be standardized with objective metrics, such as token length or number of data points, to facilitate comparisons and quantify their effects on the LLM's interpretation. Furthermore, the evaluation approach for the framework could benefit from established prompt-and-CAD benchmarking suites. These suites, which

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

typically include ground-truth data for each prompt, could help replace the human-based visual assessment used in this paper with more nuanced methods such as point cloud distance.
Despite these opportunities for refinement, challenges persist that are intrinsic to the LLMs themselves. While there is promise shown by newer multimodal models like ChatGPT-4o, all models struggled with descriptive spatial terms. In fact, the conversational capabilities of the askBack agent workflow did not lead to outperforming the stepPlanning or zeroShot agent workflows. This suggests a deficiency in current LLMs' spatial reasoning and their ability to resolve semantic ambiguities like “right angle” when multiple interpretations are possible. While visual feedback can enhance error detection, it does not appear to improve the spatial awareness of the model. These results align with recent research suggesting that general improvements in LLM capabilities do not guarantee uniform success across specialized tasks. In fact, studies seem to indicate that LLMs fundamentally struggle with reasoning and are prone to large deviations from intended outputs if non-essential information is present in prompts (Mirzadeh et al., 2024). Even though newer models such as OpenAI O1 are reportedly designed and trained with improved reasoning capabilities, it remains to be seen whether these advancements will apply to spatial problems like those explored in this paper. Future research should explore the reasoning capabilities of new released models within this framework.

6. Conclusion and outlook

In this work, we have introduced a novel framework for automated CAD geometry generation using LLM function calling and agent workflows. By integrating function calling and iterative workflows, this approach simplifies design for both novice and experienced users. We have tested this proposed framework for a set of test geometries with diverse levels of complexity in combination with several recent, widely used LLMs. The positive results from the visualInspection agent workflow, an automated visual feedback mechanism, and ChatGPT-4o highlight the potential of future LLM generations. In particular, multimodal capabilities could enhance performance in tasks requiring spatial awareness. Furthermore, we have fully incorporated this framework into an end-to-end case study in which a component has been automatically designed, optimized, and additively manufactured with minimal input from a human user. However, the framework remains focused on pure geometry creation and the agent workflows are limited in scope; additional work to increase the capabilities of the agent workflows and validate them through further case studies is needed. Integrating geometries of higher complexity is similarly required to test the framework and fully explore its capabilities. Furthermore, the LLMs themselves are affected by limitations in spatial reasoning and prompt dependency, necessitating the investigation of the performance of newer off-the-shelf models and the improvement of prompt definition and evaluation criteria.

Acknowledgements

Many thanks to Patrick Beutler, Urs Hofmann, and Jasmin Jansen for their invaluable feedback.

References

Alshare, A. A., Calzone, F., & Muzzupappa, M. (2019) Hydraulic manifold design via additive manufacturing optimized with CFD and fluid-structure interaction simulations. *Rapid Prototyping Journal*, 25(9), 1516–1524. https://doi.org/10.1108/RPJ-03-2018-0064

Bendsøe, M. P. (1989). Optimal shape design as a material distribution problem. Structural Optimization, 1(4), 193–202. https://doi.org/10.1007/BF01650949

Bialkowski, S. (2017). tOpos - GPGPU Accelerated Structural Optimisation Utility for Architects. DESIGN TOOLS - ROBOTICS (679–688). eCAADe. https://doi.org/10.52842/conf.ecaade.2017.1.679

Biedermann, M., Beutler, P., & Meboldt, M. (2021). Automated design of additive manufactured flow components with consideration of overhang constraint. *Additive Manufacturing*, 46, 102119. https://doi.org/10.1016/j.addma.2021.102119

Cagan, J., Campbell, M. I., Finger, S., & Tomiyama, T. (2005). A Framework for Computational Design Synthesis: Model and Applications. Journal of Computing and Information Science in Engineering, 5(3), 171–181. https://doi.org/10.1115/1.2013289

Chen, L., Wang, Z., Zhou,Z., Gao, T., Su, H., Zhu, J., & Li, C. (2024). MicroDreamer: Efficient 3D Generation in ~20 Seconds by Score-based Iterative Reconstruction (No. arXiv:2404.19525). *arXiv*. https://doi.org/10.48550/arXiv.2404.19525

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press

Hao, S., Liu, T., Wang, Z., & Hu, Z. (2023). ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings. *Advances in Neural Information Processing Systems*, 36, 45870–45894.
Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., & Panozzo, D. (2019). *ABC: A Big CAD Model Dataset for Geometric Deep Learning. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 9593–9603). IEEE. https://doi.org/10.1109/CVPR.2019.00983
Lee, K., Sohn, K., & Shin, J. (2024). DreamFlow: High-Quality Text-to-3D Generation by Approximating Probability Flow (No. arXiv:2403.14966). *arXiv*. https://doi.org/10.48550/arXiv.2403.14966
Makatura, L., Foshey, M., Wang, B., HähnLein, F., Ma, P., Deng, B., Tjandrasuwita, M., Spielberg, A., Owens, C. E., Chen, P. Y., Zhao, A., Zhu, A., Norton, W. J., Gu, E., Jacob, J., Li, Y., Schulz, A., & Matusik, W. (2023). How Can Large Language Models Help Humans in Design and Manufacturing? (No. arXiv:2307.14377). *arXiv*. https://doi.org/10.48550/arXiv.2307.14377
Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., & Farajtabar, M. (2024). GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models (No. arXiv:2410.05229). *arXiv*. http://arxiv.org/abs/2410.05229
Nelson, M. D., Goenner, B. L., & Gale, B. K. (2023). Utilizing ChatGPT to assist CAD design for microfluidic devices. Lab on a Chip 23(17), 3778–3784. https://doi.org/10.1039/D3LC00518F
Oh, S., Jung, Y., Kim, S., Lee, I., & Kang, N. (2019). Deep Generative Design: Integration of Topology Optimization and Generative Models. *Journal of Mechanical Design*, 141(11), 111405. https://doi.org/10.1115/1.4044229
Poole, B., Jain, A., Barron, J. T., & Mildenhall, B. (2022). *DreamFusion: Text-to-3D using 2D Diffusion*. https://doi.org/10.48550/ARXIV.2209.14988
Rohrbach A., Hendricks, L. A., Burns,K., Darrell, T., & Saenko, K. (2019).Object Hallucination in Image Captioning (No. arXiv:1809.02156). *arXiv*. https://doi.org/10.48550/arXiv.1809.02156
Schmidt, T. (2016). Potentialbewertung generativer Fertigungsverfahren für Leichtbauteil. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-52996-6
Shi, Y., Wang, P., Ye, J., Long, M., Li, K., & Yang, X. (2024). MVDream: Multi-view Diffusion for 3D Generation (No. arXiv:2308.16512). *arXiv*.https://doi.org/10.48550/arXiv.2308.16512
Wang, H., Zhao, M.,Wang, Y., Quan, W., & Yan, D.-M. (2024). VQ-CAD: Computer-Aided Design model generation with vector quantized diffusion. *Computer Aided Geometric Design*, 111, 102327. https://doi.org/10.1016/j.cagd.2024.102327
Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y.,Zhao, W. X., Wei, Z., & Wen, J.-R. (2024). A Survey on Large Language Model based Autonomous Agents (No. arXiv:2308.11432). *arXiv*. https://doi.org/10.48550/arXiv.2308.11432
Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., & Wang, C. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation (No. arXiv:2308.08155). *arXiv*. https://doi.org/10.48550/arXiv.2308.08155/
Wu, R., Xiao, C., & Zheng, C.(2021). DeepCAD: A Deep Generative Network for Computer-Aided Design Models. *2021 IEEE/CVF International Conference on Computer VisualInspection (ICCV)*, (pp. 6752–6762). IEEE. https://doi.org/10.1109/ICCV48922.2021.00670
Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., … Gui, T. (2023). The Rise and Potential of Large Language Model Based Agents: A Survey (No. arXiv:2309.07864). *arXiv*. https://doi.org/10.48550/arXiv.2309.07864
Xu, X., Lambourne, J. G., Jayaraman, P. K., Wang, Z., Willis, K. D. D., & Furukawa, Y. (2024). BrepGen: A B-rep Generative Diffusion Model with Structured Latent Geometry (No. arXiv:2401.15563). *arXiv*. https://doi.org/10.48550/arXiv.2401.15563
Yuan, Z., Lan, H., Zou, Q., & Zhao, J. (2024). 3D-PreMise: Can Large Language Models Generate 3D Shapes with Sharp Features and Parametric Control? (No. arXiv:2401.06437). *arXiv*. https://doi.org/10.48550/arXiv.2401.06437
Zhu, J., Zhou, H., Wang, C., Zhou, L., Yuan, S., & Zhang, W. (2021). A review of topology optimization for additive manufacturing: Status and challenges. Chinese Journal of Aeronautics, 34(1), 91–110. https://doi.org/10.1016/j.cja.2020.09.020
Zhuang, Y.,Yu, Y., Wang,K., Sun, H., & Zhang, C. (2024). ToolQA: A dataset for LLM question answering with external tools. *Proceedings of the 37th International Conference on Neural Information Processing Systems*, (pp. 50117–50143). Curran Associates.

Appendix A

The code for framework, including the predefined set of prompts and the experimental data, can be found here: https://github.com/R-SMP/text-to-design

https://doi.org/10.1017/pds.2025.10203 Published online by Cambridge University Press