Learning From Design Procedure To Generate CAD Programs for Data Augmentation

Yan-Ying Chen, Dule Shu, Matthew Hong, Andrew Taber, Jonathan Li, Matthew Klenk
Toyota Research Institute Pittsburgh, PA 15213
yan-ying.chen@tri.global

Abstract

Large Language Models (LLMs) have demonstrated impressive capabilities in a wide range of code generation tasks. However, generating code for certain domains remains challenging. One such domain is Computer-Aided Design (CAD) program, where the goal is to produce scripted parametric models that define object geometry for precise design and manufacturing applications. A key challenge in LLM-based CAD program generation is the limited geometric complexity of generated shapes compared to those found in real-world industrial designs. This shortfall is in part due to the lack of diversity in the available CAD program training data. To address this, we propose a novel data augmentation paradigm that prompts an LLM to generate CAD programs conditioned on a reference surface program and a modeling procedure - an idea inspired by practices in industrial design. By varying the reference surface using a collection of organic shapes, our method enriches the geometric distribution of generated CAD models. In particular, it introduces edges and faces defined by spline-based curvature, which are typically missing or underrepresented in existing open-source CAD program datasets. Experiments show that our method produces CAD samples with significantly greater geometric diversity and a higher resemblance to industry-grade CAD designs in terms of the proportion of organic shape primitives. This enhancement makes our CAD data augmentation approach a useful tool for training LLMs and other deep learning models in CAD generation.

1 Introduction

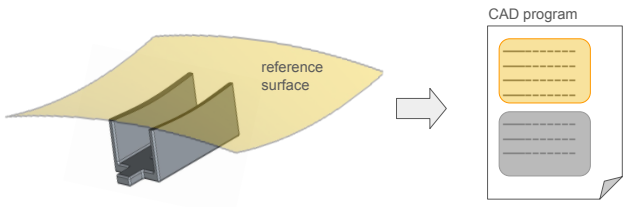

Figure 1: It is common to use a reference surface to guide CAD creation for specific design intent, e.g., compatibility to other component. Motivated by this idea, we propose to a new design procedure prompting to guide the CAD program generation toward more organic shapes.

Computer-Aided Design (CAD) is important for product design and engineering because it carries high-fidelity, parametric 3D information such as parts and topological structures that are essential for engineering and manufacturing physical products. Designers have been using CAD software, e.g.,

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Deep Learning for Code in the Agentic Era.

Solidworks and Fusion 360, to create CAD models through a sequence of operations such as sketch and extrude. These operations can be interpreted to different formats of CAD model representation (e.g., B-rep, boundary representation, in a STEP file) or 3D artifacts (e.g., mesh in a STL file) for a variety of downstream work such as physical simulation [20] and shape optimization [14].

CAD is a classic example of visual programming, where a CAD program is created by a sequence of operations that define a 3D shape. As a vital step towards automated CAD programming, larger CAD program datasets [26, 25] have been proposed for training CAD program generation models. However, it is much more challenging to scale up than other visual data formats such as images or videos, because CAD programming involves multiple modalities and requires domain-specific expertise [9] to create data. In addition, CAD data are mostly in closed systems and spread across different CAD tools without organically growing data sources similar to repositories for software programming, e.g., GitHub. Hence, the existing CAD program datasets are mostly limited in a small set of operation commands and result in simple CAD data that are far from industrial designs [10].

We argue that a key recipe for scaling is a sustainable eco-system to synthesize CAD programs for data augmentation. We leverage recent efforts in bridging CAD programming to generic programming languages, such as the python script libraries CadQuery[3] and Build123d[1]. This exciting progress enables the application of LLM-based knowledge of common programming languages (e.g., python) to CAD programming. This advantage drives the trend of finetuning approaches [21, 27, 5] built upon those libraries and LLMs. This trend brings a turning point of architecting a data simulation and augmentation system to scale up the data needed for training CAD generation models. However, how to prompt these LLMs to simulate the CAD design process and synthesize data with geometric properties similar to industrial design standards remains an underexplored topic.

Like software engineers, CAD engineers/designers embed deep expertise and tacit knowledge in both the design processes and domains they operate in. A common strategy used by CAD designers is a modeling procedure where they create a sequence of operations from a chosen reference surface; for example, a wall that a bracket is supposed to support as illustrated the left in Figure 1. The reference surface significantly impacts the resulting design because the sequential operations can create ripple effects, e.g., all curves adjacent to a surface must align with a part of its curvature. Hence, an internal bracket may comprise organic shapes (e.g., free form B-Spline shapes) in addition to standard primitives (e.g., flat rectangles and cylinders) because it needs to match the design's exterior. In addition, designers use field-specific design languages (e.g., double-spoke, Y-spoke, the number of spokes in a car wheel) to guide their design creation [1]. These semantically encoded design patterns hold significant meaning for industrial design practices as they embody human-interpretable representations that play a critical role in making the generated program align more closely with design intent. The combination of reference surface, design language and procedure constitute a multimodal expression to bridge human creativity and computer-aided design, enabling a more seamless translation between conceptual ideas and visual form.

Inspired by this design scenario, we propose a designer-centered data augmentation paradigm by leveraging (1) a reference surface program prompt and (2) a design procedure prompt to synthesize CAD programs with more organic shapes. Our experiments demonstrate that, in comparison to the benchmark CAD program datasets, the CAD programs generated by our approach, after being interpreted to 3D CAD B-rep format, have improved diversity of shape complexity. In addition, the geometric properties (e.g., B-Spline ratio) of our generated data are more similar to those of industrial-grade CAD designs. Finally, our ablation study highlights the importance of the proposed reference surface prompt in driving the generation of B-Spline shapes and the advantage of program modularization for reflecting the proposed design procedures.

2 Related Work

CAD program generation [26, 24, 2, 21, 27] has gained increased attention for generating 3D CAD designs. While parallel efforts such as primitive fitting [17, 18] and B-rep generation models [28, 12] are focused on 3D shape and topology representation, CAD program describes a CAD design as a sequence of design operations, and is more programmable and convertible to other CAD representation formats.

[1] `https://github.com/gumyr/build123d`

Earlier works such as Fusion360 [25], DeepCAD [26], and GenCAD [29] leverage Domain Specific Languages (DSLs) (e.g., FeatureScript [6], structured text representation [30], CAD assembly JSON [9]) to parse CAD operations and parameters. Based on a DSL, the generation of CAD programs can be performed by a deep reinforcement learning agent [22] or by a generative model such as GAN [7] and diffusion model [8] in a latent space learned by a transformer [23]. In addition, DSLs have been used to fine-tune LLMs for text-to-CAD generation [24, 9] and VLMs for CAD generation conditioned by image and text [2, 27]. To improve the compatibility of DSL representation with LLMs, recent work [13] has explored an augmented representation of CAD generation process which combines natural language-based text descriptions and parametric annotations in a hierarchical semantic description.

Python-based parametric CAD scripts such as CadQuery [3] and OpenSCAD [19] have been incorporated by many popular foundational LLMs, including OpenAI o3, Gemini, and Claude, into their code generation features. Taking advantage of this built-in feature, recent work such as CAD-recode [21] for point cloud-to-CAD generation and Cad-coder [5] for image-to-CAD generation has proposed fine-tuning open-source foundational LLM models such as Qwen [4] and LLaVA [16] for conditional CAD generation. The rise of foundation LLM models in conditional CAD generation has created new opportunities for CAD data augmentation.

Despite the active research on CAD program generation mentioned above, current deep generative models (including LLMs) for CAD generation are still far from meeting industry's need for automated engineering design. Previous work [10] has indicated most 3D CAD generation datasets such as DeepCAD [26] and ABC [11] have been intentionally restricted in sketch-extrude operations that do not align with most real manufactured shapes. To narrow this gap and increase the shape complexity of publicly available CAD datasets, GenCAD [2] and CAD-MLLM [27] propose to develop multimodality datasets by using image prompts for data augmentation. However, the conditioning images used in those works are primarily rendered from open-source CAD program samples (i.e., DeepCAD), therefore constraining the shape distribution of the training dataset to shapes covered by existing CAD program datasets and preventing the deep learning models from accessing more sophisticated geometric features such as B-Spline geometries commonly seen in industry-level CAD objects.

In comparison, our proposed approach ensures the generation of B-Spline geometries by including in our LLM prompts a Python script that defines a B-Spline-shaped reference surface. By combining this Python script with a natural language-based description of a design procedure, we command the CAD-generating LLMs to conform to the reference surface and sequentially influence other connected primitives, thus synthesizing CAD objects with B-Spline geometries. Our proposed design prompts which combines text description and reference surface CAD programs is compatible with any off-the-shape code-generating LLM and provides users with a conditioning mechanism for CAD generation that is both parameterizable and visualization.

3 Proposed Method

The goal of this work is to develop a CAD program generation approach for data augmentation which enables the generation of complicated geometric properties commonly found in industrial practice. The core idea of our approach is a novel prompting strategy for LLM-based CAD program generation that is inspired by CAD model design practices from industry.

3.1 Design Procedure Prompting

A formative study conducted with industrial designers to understand their design practices has introduced to us the following design procedure. The creation of a 3D CAD object starts from a given reference surface that the target CAD object is supposed to match. All subsequent operations that lead to the final CAD object will therefore influence the choice of the reference surface. In many industrial practices, the reference surface is chosen as a free-form B-spline shape for aesthetic or functional purposes. First, in exterior styling, B-spline organic curves are often used by designers to create smooth and flowing surfaces. Second, for interior components, ergonomic considerations are another key reason for using B-splines because these curves conform more naturally to a human body. In addition, from the perspective of structure, B-spline geometries are advantageous. In automotive body design, sharp creases or abrupt curvature transitions can lead to stress concentrations and

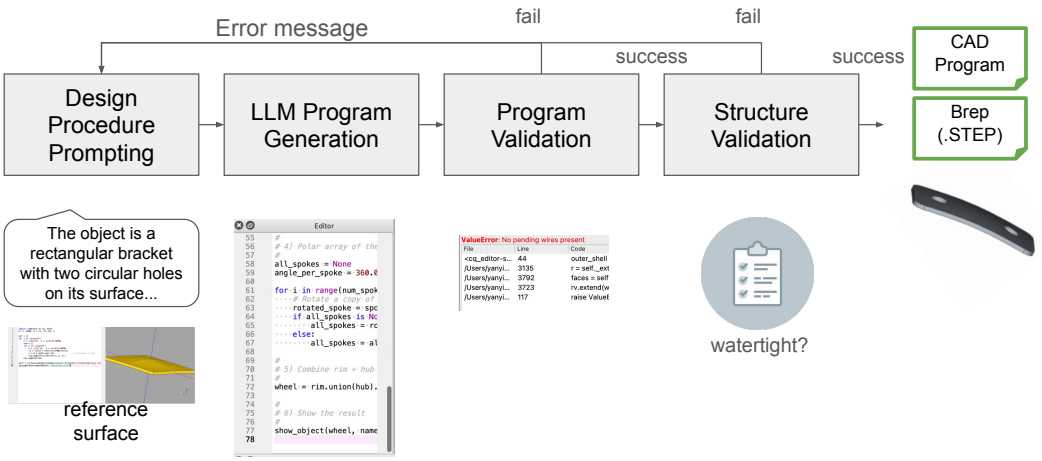

Figure 2: System overview: (1) Design procedure prompting takes a design description and a reference surface program as an input to formulate the design procedure. (2) The design prompt then condition an LLM to generate a CAD program. (3) Program validation executes the generated program to visualize a CAD Brep. (4) Structure validation checks the validity of the CAD Brep.

starting points for buckling. A smooth, continuous B-spline surface helps distribute stress more evenly. Unlike a simple design procedure starting from basic sketch and extrusion (commonly found in public CAD program datasets), the B-Spline reference surface induces organic shaped primitives (e.g., faces and edges) parametrized by higher-order polynomial functions in the final design of the CAD object. The reference surface will be removed after the CAD object is created.

To approximate this design procedure, we prompt an LLM CAD program generation model with a text description of the intended design and a reference surface input. The text prompt template is formatted as follows.

text prompt := [prefix system prompt, design description, design context, postfix system prompt]

Prefix system prompt: We instruct the program generation model to write a python-based CAD program as follows: *"Use Python CadQuery library to write a CAD program of a bracket that is described as follows."*
Design description: We provide the generation model with a description of an intended design, e.g., *"The object is a rectangular bracket with two circular holes on its surface."* as shown in Figure 2.
Design context: we offer more detailed instructions on design requirement and procedure: *"The shapes of the bracket look smooth. The bracket should conform to the curvature of the reference surface in the CAD program below. After the bracket is created, the reference surface should be removed."*
Postfix system prompt: We require that the generated CAD object is watertight solid and specify the path of the output STEP file after program execution. The details of postfix system prompt is reported in Appendix A.

The content of both prefix and postfix system prompts remain the same for each instance of CAD generation while the design description varies to create diverse designs. The design context remains constant for the same target category of design (e.g., bracket) and is updated as a new target category (e.g., car wheel) is introduced. The example full prompts are reported in Appendix B

3.2 Reference Surface Program

Instead of using an image or a point cloud to represent a reference surface, we use a Python script representation of CAD program as an input prompt as shown in Figure 2. The reason for using a script-based program to represent a reference surface is that the script provides an accurate and expressive parametric description of 3D geometry that is well-understood by off-the-shelf LLMs extensively trained for Python code generation. In contrast, directly prompting a CAD program generating LLM with visual content tends to produce inaccurate geometry due to LLMs' limitation

in learning cross-modality correlation, especially if the surface has a more organic shape instead of typical primitives.

To ensure a higher level of shape diversity in CAD data augmentation, we included four types for reference surfaces: Gaussian surface, saddle surface, wave surface and ripple surface - all of which are B-Spline surfaces. Each surface type is represented in a Python script using the CADquery library. To further increase shape diversity, we vary the parameters of each reference surface script to create more surface geometries; for example, we vary the curvature of a saddle surface from shallow to deep, with the variation automated by updating shape parameters in the Python script. In each instance of CAD generation, we select a reference surface and pair it with a design description to formulate a prompt. The design description data are provided in the experiment setting in Sec 4.1.

3.3 Program Generation and Validation

As illustrated in Figure 2, the formatted prompt is fed to an LLM for CAD program generation. The LLM can be almost any popular off-the-shelf foundational LLM, and we chose to use OpenAI o3 [2] in our experiments. Each generated program goes through two agentic validation stages: (1) program validation to verify whether it can be converted to a boundary representation (B-rep) and exported to a STEP file and (2) structure validation to verify whether the generated B-rep is watertight and structurally feasible using the validity check proposed in DTGBrepGen [15]. Any identified errors in the CAD program are fed back to the prompt in an iterative process for self-correction. Once validated as a successful generation, the CAD program script is finalized and then converted to a B-rep.

4 Experiments

The purpose of the experiments is to evaluate whether the proposed data augmentation method can synthesize CAD designs that contain a larger portion of shape primitives with free-form organic shapes resembling actual designs for industry application. We first explain the evaluation metrics and existing datasets as the baselines, followed by a comparison to the geometric properties of the baselines and our generated CAD data. An ablation study is reported to investigate the effectiveness of different prompting strategies for generating organic shapes, and also illustrates how the proposed prompting strategies affect the programmed modules. Finally, we show the potential for extending the proposed approach to multiple design targets.

4.1 Experiment Setting

Evaluation metrics: The metrics include several geometric properties: (1) The number of lines of content in each STEP file. More lines of content in a STEP file generally indicates a more complex CAD design, as STEP files contain a formatted ASCII text description of geometric elements and their connectivity in a CAD object. (2) the number of faces and curves as defined in a Brep. (3) the proportion of B-rep STEP files with B-Spline faces and curves out of all generated STEP files. In addition, the B-Spline ratio β_i of each CAD object c_i is calculated as,

$$\beta_i = [(f_i^b/f_i) + (e_i^b/e_i|)]/2, \tag{1}$$

where f_i is the number of faces, f_i^b is the number of B-Spline faces, e_i is the number of curves and e_i^b is the number of B-Spline curves. B-Spline, or more generally the non-uniform rational basis spline (NURBS), are commonly used in computer graphics and CAD design to represent a wide range of parameterized geometrical shapes, where the flexibility allows it to define more free-form shapes than standard primitives. Higher B-Spline Ratio means that more proportion of B-Spline shapes appear in the geometry. Note that these proxy metrics may not completely reflect true industrial-grade quality. Real-world quality also depends on factors like curvature continuity, fillet robustness, manufacturing tolerances, and feature intent, which are not assessed in the scope of this paper.

Baselines: In our experiments, the bracket is chosen as the target category for the generation of CAD, because brackets are a class of common mechanical parts and can be conveniently retrieved from

[2]The snapshot o3-2025-04-16 at `https://platform.openai.com/docs/models/o3` was used. The parameter of reasoning effort was set to be high. The cost of the API was $8 per 1M output tokens.

Table 1: A comparison of geometric properties with a commercial industry bracket dataset and baseline CAD program datasets DeepCAD-b, GenCAD and CAD-MLLM. (*) indicates that the data also include non-bracket objects.

Bracket Data	Industry	DeepCAD-b	GenCAD*	CAD-MLLM*	Ours
avg. #lines (STEP)	10099	1783	991	2402	4494
avg. #faces	82.91	21.48	12.55	28.65	26.57
avg. #curves	259.4	50.30	27.86	69.26	67.57
w/ B-Spline faces	100%	0%	0%	0%	77%
w/ B-Spline curves	100%	1%	1%	1%	89%
B-Spline Ratio	0.5352	0.0004	0.0009	0.0008	0.2217

the existing collection of commercial industry CAD designs (referred to as "Industry") and from the open-source CAD program dataset DeepCAD [26] with keyword labels for filtering [9] (referred as "DeepCAD-b"). In addition to commercial industry designs and DeepCAD samples, we also included GenCAD dataset [3] and CAD-MLLM dataset [4] as benchmark baselines for a quantitative evaluation. Note that GenCAD and CAD-MLLM samples in our analysis includes CAD objects outside the bracket category as no keyword filtering is available. In addition, our approach requires design descriptions as part of the text prompt as mentioned in Sec 3. For the experiments of bracket generation, the text descriptions [9] associated with each of the DeepCAD bracket data are used as our design descriptions.

4.2 Geometric Properties of Programmed CAD Data

As reported in the first column from the right of Table 1, our approach generates CAD programs where the corresponding B-rep data have a higher or similar average number of STEP lines, surfaces and curves in comparison to the existing CAD program datasets: DeepCAD-b, GenCAD and CAD-MLLM. This suggests that our approach is able to synthesize more designs with an organic shape without compromising geometric complexity. In particular, 77% of the CAD objects generated by our approach contain B-Spline faces, whereas none of the CAD objects from DeepCAD-b, GenCAD and CAD-MLLM contain any B-Spline face. Similarly, 89% of the CAD objects generated by our approach contain B-Spline curves, in contrast to only 1% of the CAD objects from DeepCAD-b, GenCAD and CAD-MLLM containing B-Spline curves. In terms of the mean B-Spline Ratio in a CAD object, our generated CAD objects also outperform DeepCAD-b, GenCAD, and CAD-MLLM by a large margin, as shown in the bottom row of the table. This result is aligned with previous work [10] that the existing public CAD datasets are much simpler than the commercial CAD data used in industry. It also demonstrates that our generated CAD objects have a more organic shape than the current open-source CAD program datasets, featuring a much closer resemblance to the CAD designs from industry where every sample contains B-Spline geometry and over 50% of the shape primitives in an object are B-Splines.

4.3 Diversity of Shape Complexity

As shown in Figure 3, the CAD data generated by our approach are more evenly distributed over different B-Spline ratio, which suggests the capability to generate programmed CAD data that covers a wider range of shape complexity than DeepCAD-b, GenCAD and CAD-MLLM. Note that the distributions of DeepCAD-b, GenCAD, and CAD-MLLM over B-Spline ratio are very similar in a sense that a majority of the samples have zero B-Spline components. This is likely because GenCAD and CAD-MLLM are both built upon DeepCAD and therefore inherit the same limitation on the B-Spline-free operations adopted by DeepCAD. Compared with GenCAD and CAD-MLLM, our approach does not require image prompts and instead uses a script-based CAD program for reference surface to condition CAD program generation. Since both the script-based CAD program and the natural language-based text description are processed as text tokens by an LLM, no multimodal data processing of the LLM or VLM is required, which allows a more efficient and

[3] downloaded from `https://github.com/ferdous-alam/GenCAD`

[4] downloaded from `https://huggingface.co/datasets/jingwei-xu-00/Omni-CAD/resolve/main/Omni-CAD.zip`

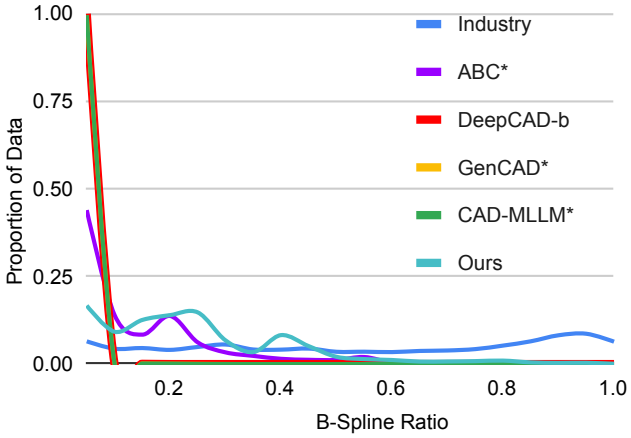

Figure 3: Distribution of B-Spline ratio over data samples: Industry, DeepCAD-b and ours only contain bracket data. ABC, GenCAD and CAD-MLLM include brackets and other objects.

coherent generation of CAD designs and avoids the potential generation artifacts due to a suboptimal contrastive language-image pre-training.

In addition to the existing CAD program datasets, we also include the ABC dataset[11] (referred to as ABC-misc), by far the largest public CAD dataset. ABC-misc includes miscellaneous CAD STEP files without the corresponding CAD programs. Note that all publicly available baselines 4.1 are direct or indirect derivatives of ABC dataset by reversing a subset of the ABC 3D CAD objects to CAD programs using a limited set of CAD operations while discarding those that cannot be parsed successfully. ABC as a larger collection is considered to have more geometric diversity than the CAD program datasets. We conduct experiments on the commonly used test split of ABC [17, 18]. In comparison to the ABC dataset, our generated data is less skewed at low B-Spline region, offers a more significant representation in the higher B-Spline region, and is closer to the more evenly-distributed industrial bracket datasets (Industry).

4.4 Ablation Study on Design Procedure Prompting

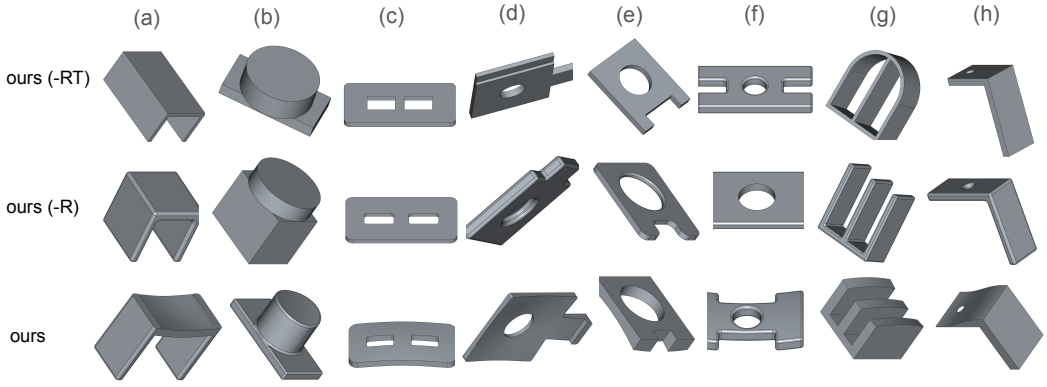

Figure 4: Examples of CAD B-rep visualization of our approach using reference surfaces, and the alternatives (-RT) and (-R) excluding reference surfaces. (-R) include a text guidance to prompt smooth and organic shapes. Ours with reference surfaces can generate more B-Spline shapes.

An ablation study is conducted to test the effectiveness of using reference surface prompts and the alternatives, referred to as "ours(-RT)" and "ours(-R)". Ours(-RT) completely removes the design context (instructing the design procedure) in the text prompt (Sec. 3.1) and excludes the input reference surface program. Ours(-R) is similar to ours(-RT), but replaces the original design context

Table 2: An ablation study of our approach, testing the alternative prompts (-RT) and (-R) without using reference surfaces.

Approach	Ours(-RT)	Ours(-R)	Ours
avg. #lines (STEP)	1225	2992	4494
avg. #faces	14.86	34.56	26.57
avg. #curves	35.17	75.89	67.57
w/ B-Spline faces	2%	18%	77%
w/ B-Spline curves	6%	27%	89%
B-Spline Ratio	0.0085	0.0478	0.2217

with a text-based shape guidance, i.e., "The shapes of the bracket look smooth and organic." This is to test to what extent a simple text-based shape guidance may affect the generated results.

As shown in Table 2, the B-Spline ratio is largely decreased if the reference surface prompt or the design procedure prompt are not used, i.e., ours vs. ours(-RT), suggesting the importance of the proposed design procedure prompting. Ours(-R), by excluding the reference surface prompt and using text-based shape guidance, only slightly improves the B-Spline ratio, i.e., ours(-R) vs. ours(-RT). This suggests surface program is the key factor to guide the program generation to yield more B-Splines.

Interestingly, the average number of faces and curves of ours(-R) are higher than ours. We suspect that the LLMs program generation model attempts to generate more standard primitives to illustrate the text guidance regarding smooth and organic shapes, instead of using B-Splines to concisely represent the intended shapes. This suggests the limitation of natural language guidance for complicated shapes.

Figure 4 presents the generated CAD B-rep examples. The same design description is applied to each design in a column. Ours has a more variety of curvatures than ours(-RT) and ours(-R), attributed to the intent to match varied reference surfaces.

4.5 Program Modularization for Design Procedure

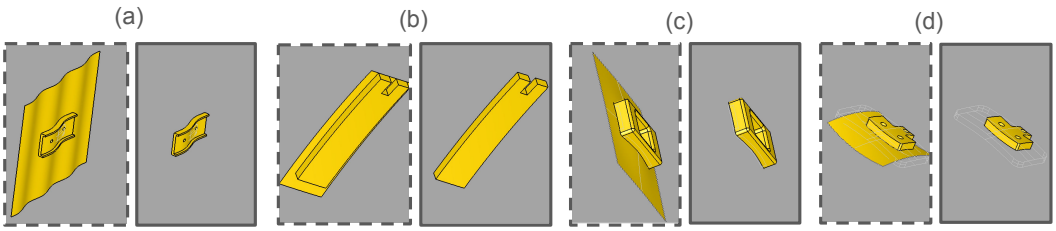

Figure 5: Visualization of the generated program modules. The left side of each column is a generated bracket with its reference surface, and the right side is the bracket after the reference surface is removed.

The design procedure conditioned on a reference surface is visualized in Figure 5, where CQ-Editor [5] is used to compare the programmed modules with (left) and without (right) the presence of the reference surface. Each created bracket naturally comprises more B-Splines as a result of matching the given reference surface. The variation of reference surfaces creates the necessary shape diversity for data augmentation. Unlike voxel-based deformation of 3D objects, the deformation by our approach is implemented at the CAD program level using LLM prompts, which provides a smoother and more precise shape control for procedural generation of 3D shapes.

4.6 Generalizability to other Design Targets

To test the generalizability to more design targets, a qualitative experiment is conducted on car wheels using our approach. We leverage the design patterns used in the prior paper [1] to formulate the

[5] https://github.com/CadQuery/CQ-editor

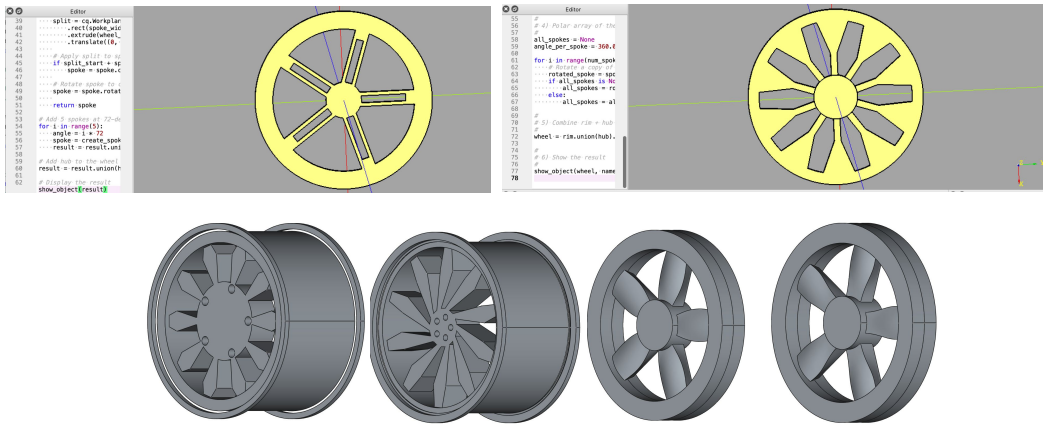

Figure 6: Examples of generated wheels. The top row are the program visualization. The bottom row are the CAD Breps. Reference surfaces are used to generate the two wheels in the bottom right.

Table 3: The proportion of generated data that requires more than 5 iterations of re-generation due to program execution errors or structure validation failures.

Approach	Ours(-RT)	Ours(-R)	Ours
require > 5 iterations	12%	18%	21%

design descriptions. The upper plots in Figure 6 shows the generated program and the visualization in CQ-Editor, where the design descriptions specify the number of spokes (5 vs. 10) and the type of spokes (double-spoke vs. Y-spoke). The lower plots shows a visualization of the STEP files of more generated wheels, prompted by the design description with regard to the hub and the barrel. A reference surface is used to prompt the spoke shapes of the two wheels counting from right to left.

5 Limitation and Discussion

While design procedure is demonstrated to be useful for CAD program generation, it also creates more complicated programs that may result in more execution errors and invalid structures. This highlights the importance of the program validation and the structure validation used in our proposed system (Fig. 2). Validation feedback and iterative generation largely improve validity, but also increase the iterations needed for a successful generation, as reported in Table 3.

Our approach still requires the programs of reference surfaces. While we can sample a variety of reference shapes for data augmentation purpose, script-based surface programs are less available for prompting precision control generation. Logging of operations is required when a designer is creating a reference surface; without logged operations, a reverse engineering process would be required to obtain/generate a reference surface program. Nevertheless, reference surface program generation is a reduced task of general CAD program generation and can be critical to progressively address the challenges.

Our approach cannot precisely control where the bracket is supposed to match the reference surface. In Figure 4 (a) a reference surface is applied to the base, while in (g) a reference surface is applied to the three feet. Precision control requires a deeper understanding for CAD parameterization, which might be assisted by a semi-automated way (e.g., human intervention) or LLM models finetuned by pairs of precise input and output. We hope our data augmentation work can contribute to scaling up diverse data for finetuning precision generation models.

6 Conclusion

In the presence of the current challenge of lacking CAD program datasets with complicated geometry, an effective data augmentation strategy to enrich CAD program datasets could be an essential piece

to scaling up the deep learning models' performance in CAD program generation. Inspired by CAD design practice from industry, we propose a novel procedure for LLM prompting which uses a reference surface program to guide the generation of CAD programs. Experiment results demonstrate that the generated CAD programs by our proposed approach has shown a significant improvement on the diversity of shape complexity compared with baseline CAD programs for LLMs. A comparison with practical CAD design from industry indicates that our method is able to narrow the gap in geometric complexity between synthesized CAD designs from machine learning and practical CAD designs used in industry applications.

References

[1] *Inspired by AI? A Novel Generative AI System to Assist Conceptual Automotive Design*, volume Volume 6: 36th International Conference on Design Theory and Methodology (DTM) of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 08 2024.

[2] Md Ferdous Alam and Faez Ahmed. Gencad: Image-conditioned computer-aided design generation with transformer-based contrastive representation and diffusion priors, 2025.

[3] AU, Jeremy Wright, thebluedirt, Marcus Boyd, Lorenz, Innovations Technology Solutions, Hasan Yavuz ÖZDERYA, Bruno Agostini, Jojain, Michael Greminger, Seth Fischer, Justin Buchanan, cactrot, huskier, Ruben, iulianOnofrei (U-lee aan), Miguel Sánchez de León Peque, Martin Budden, Hecatron, Peter Boin, Wink Saville, Pavel M. Penev, Bryan Weissinger, M. Greyson Christoforo, Jack Case, AGD, Paul Jurczak, nopria, moeb, and jdegenstein. Cadquery/cadquery: Cadquery 2.4.0, January 2024.

[4] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[5] Anna C Doris, Md Ferdous Alam, Amin Heyrani Nobari, and Faez Ahmed. Cad-coder: An open-source vision-language model for computer-aided design code generation. *arXiv preprint arXiv:2505.14646*, 2025.

[6] Onshape featurescript. `https://cad.onshape.com/FsDoc/`.

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[9] Mohammad Sadil Khan, Sankalp Sinha, Sheikh Talha Uddin, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential CAD designs from beginner-to-expert level text prompts. In *Advances in Neural Information Processing Systems*, volume 37, pages 7552–7579. Curran Associates, Inc., 2024.

[10] MA Kimmel, Mueed Ur Rehman, Yonatan Bisk, and Gary K. Fedder. Position: You can't manufacture a neRF. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025.

[11] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[12] Mingi Lee, Dongsu Zhang, Clément Jambon, and Young Min Kim. Brepdiff: Single-stage b-rep diffusion model. SIGGRAPH Conference Papers '25, New York, NY, USA, 2025. Association for Computing Machinery.

[13] Jiahao Li, Weijian Ma, Xueyang Li, Yunzhong Lou, Guichun Zhou, and Xiangdong Zhou. Cad-llama: Leveraging large language models for computer-aided design parametric 3d model generation. In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2025.

[14] Jichao Li, Xiaosong Du, and Joaquim R.R.A. Martins. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences*, 134:100849, 2022.

[15] Jing Li, Yihang Fu, and Falai Chen. Dtgbrepgen: A novel b-rep generative model through decoupling topology and geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[16] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.

[17] Yilin Liu, Jiale Chen, Shanshan Pan, Daniel Cohen-Or, Hao Zhang, and Hui Huang. Split-and-fit: Learning b-reps via structure-aware voronoi partitioning. *ACM Trans. Graph.*, 43(4), July 2024.

[18] Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. Point2cad: Reverse engineering cad models from 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3763–3772, 2024.

[19] Open Cascade S.A.S. Open Cascade Technology (OCCT), 1999. Open-source platform for 3D CAD/CAM/CAE; LGPL-2.1 with exception.

[20] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

[21] Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-recode: Reverse engineering cad code from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2025.

[22] Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In *International conference on machine learning*, pages 9367–9376. PMLR, 2020.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[24] Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. Text-to-CAD generation through infusing visual feedback in large language models. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

[25] Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.

[26] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6772–6782, October 2021.

[27] Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm, 2024.

[28] Xiang Xu, Joseph Lambourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024.

[29] Nomi Yu, Md Ferdous Alam, A. John Hart, and Faez Ahmed. Gencad-3d: Cad program generation using multimodal latent space alignment and synthetic dataset balancing. *Journal of Mechanical Design*, pages 1–17, 2025.

[30] Zhanwei Zhang, Shizhao Sun, Wenxiao Wang, Deng Cai, and Jiang Bian. Flexcad: Unified and versatile controllable cad generation with fine-tuned large language models. In *The Thirteenth International Conference on Learning Representations*, 2024.

A Postfix System Prompt

The details of postfix system prompt is reported as below.

"Make sure the generated CAD model is watertight solid. Please export the generated CAD model to output.stl file and output.step file. Please do not visualize it. Here is the document of CadQuery for your reference (`https://cadquery.readthedocs.io/en/latest/index.html`). Do not output explanation."

B Example Full Prompts

The prefix system prompt, the design description, the design context and the postfix system prompt are color coded in gray, blue, purple and brown, repspectively, followed by the program of a reference surface.

Example 1:
Use Python CadQuery library to write a CAD program of a bracket that is described as follows. The object is a U-shaped, open-ended, three-dimensional structure with a flat bottom and curved edges. It resembles a bracket or support structure. The shapes of the bracket look smooth. The bracket should conform to the curvature of the reference surface in the CAD program below. After the bracket is created, the reference surface should be removed. Make sure the generated CAD model is watertight solid. Please export the generated CAD model to output.stl file and output.step file. Please do not visualize it. Here is the document of CadQuery for your reference (https://cadquery.readthedocs.io/en/latest/index.html). Do not output explanation.

```
# saddle.py
import cadquery as cq, math
U, V, SPAN, CURV = 300, 300, 50, 0.004

net = []
for i in range(U):
    u = i/(U-1);   x = (u-0.5)*SPAN
    row = []
    for j in range(V):
        v = j/(V-1);   y = (v-0.5)*SPAN
        z = CURV*(x**2 - y**2)
        row.append(cq.Vector(x, y, z))
    net.append(row)

surf = cq.Face.makeSplineApprox(net)
cq.exporters.export(surf, "saddle.step")
```

Example 2:
Use Python CadQuery library to write a CAD program of a bracket that is described as follows. A rectangular bracket with two holes and two slots. The shapes of the bracket look smooth. The bracket should conform to the curvature of the reference surface in the CAD program below. After the bracket is created, the reference surface should be removed. Make sure the generated

CAD model is watertight solid. Please export the generated CAD model to output.stl file and output.step file. Please do not visualize it. Here is the document of CadQuery for your reference (https://cadquery.readthedocs.io/en/latest/index.html). Do not output explanation.

```
import cadquery as cq, math
U, V, SPAN, H = 100, 100, 100, 7

net = []
for i in range(U):
    u = i/(U-1);   x = (u-0.5)*SPAN
    row = []
    for j in range(V):
        v = j/(V-1);   y = (v-0.5)*SPAN
        r2 = (x**2 + y**2)/((SPAN/3)**2)
        z = H * math.exp(-r2)                    # Gaussian height
        row.append(cq.Vector(x, y, z))
    net.append(row)

surf = cq.Face.makeSplineApprox(net).thicken(2).translate((0,0,-1))
cq.exporters.export(surf, "gaussian.step")
```