NURBGen: High-Fidelity Text-to-CAD Generation through LLM-Driven NURBS Modeling

Muhammad Usama[1,2,3*], **Mohammad Sadil Khan**[1,2,3*†], **Didier Stricker**[1,2], **Muhammad Zeshan Afzal**[1,3]

[1]German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
[2]RPTU Kaiserslautern-Landau (RPTU), Germany
[3]MindGarage, Kaiserslautern, Germany
{muhammad.usama, mohammad.khan, didier.stricker, muhammad_zeshan.afzal}@dfki.de

Abstract

Generating editable 3D CAD models from natural language remains challenging, as existing text-to-CAD systems either produce meshes or rely on scarce design-history data. We present NURBGen, the first framework to generate high-fidelity 3D CAD models directly from text using Non-Uniform Rational B-Splines (NURBS). To achieve this, we fine-tune a large language model (LLM) to translate free-form texts into JSON representations containing NURBS surface parameters (*i.e*, control points, knot vectors, degrees, and rational weights) which can be directly converted into BRep format using Python. We further propose a hybrid representation that combines untrimmed NURBS with analytic primitives to handle trimmed surfaces and degenerate regions more robustly, while reducing token complexity. Additionally, we introduce partABC, a curated subset of the ABC dataset consisting of individual CAD components, annotated with detailed captions using an automated annotation pipeline. NURBGen demonstrates strong performance on diverse prompts, surpassing prior methods in geometric fidelity and dimensional accuracy, as confirmed by expert evaluations.

Code — https://github.com/SadilKhan/NURBGen
Project —
https://muhammadusama100.github.io/NURBGen

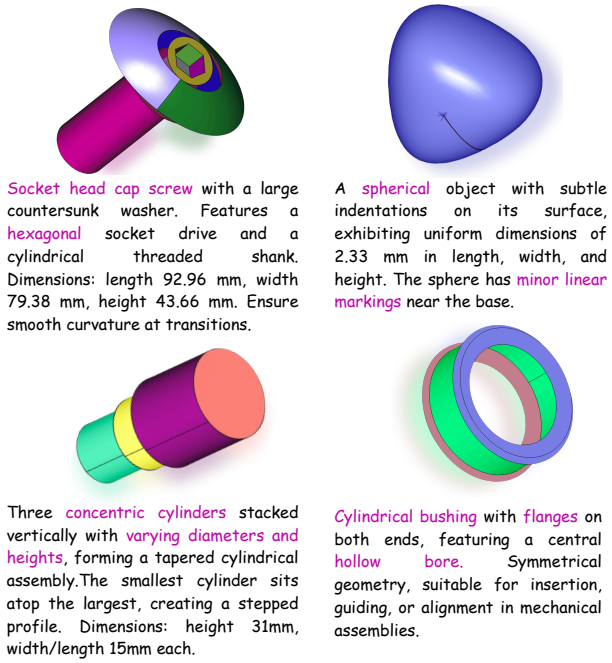


Figure 1: Text-to-CAD generation results from NURBGen, showcasing reconstructed CAD models from text prompts.

Introduction

Computer-Aided Design (CAD) plays a fundamental role in modern engineering, product design, and digital manufacturing workflows (Vido et al. 2024; Gao et al. 2015). It enables precise, parametric modeling of complex mechanical and architectural components. However, creating detailed CAD models typically requires expert knowledge of professional design software-such as Onshape (https://www.onshape.com) or AutoCAD (https://www.autodesk.com/products/autocad/overview), and remains a labor-intensive and, time-consuming task.

Researchers have therefore proposed deep learning-based approaches for automatic CAD modeling from high-level inputs such as natural language (Khan et al. 2024b), images (Chen et al. 2024), or point clouds (Liu et al. 2023). Among these, text-to-CAD generation offers a simple, intuitive interface that allows designers to describe 3D objects in natural language, bypassing the need for expert modeling skills. However, nearly all prior methods (Li et al. 2025; Kapsalis 2024; Khan et al. 2024b) rely on design-history-based representations (Wu, Xiao, and Zheng 2021; Khan et al. 2024a; Zhou, Tang, and Zhou 2023), where shapes are constructed via sequences of parametric operations—extrusions, and 2D sketches. While intuitive and highly editable, these methods are trained on small-scale datasets like DeepCAD (Wu, Xiao, and Zheng 2021), which mostly contain low-complexity parts (e.g., cuboids, cylinders), limiting generalization in real-world scenarios.

In contrast, the ABC dataset (Koch et al. 2019), which con-

*Equally contributing first authors.
†Corresponding Author.

tains over a million 3D CAD models, remains comparatively underutilized in text-to-CAD research due to two key limitations. First, ABC represents geometry in Boundary Representation (BRep) form, which lacks design history. BReps define solids using analytic surface patches—most commonly NURBS, the industry standard for their precision and parametric control. However, NURBS-based modeling is rarely explored in deep generative research due to the challenge of efficient representation (Yang, Wang, and Wang 2024), non-differentiability of knot vectors (Prasad et al. 2022), high parameter variability, and trimming complexity inherent to NURBS geometry. NeuroNURBS (Fan et al. 2024) partially addresses this by learning latent codes for untrimmed NURBS surfaces via a non-autoregressive transformer VAE, but it does not support language-based generation and cannot model complex shapes due to the trimming issues. Second, ABC lacks high-quality text descriptions, making it difficult to train or evaluate text-conditioned generative models.

In this work, we present **NURBGen**, the first framework for generating 3D CAD models from natural language using structured, symbolic NURBS representations. Unlike prior work that learns dense latent codes (Fan et al. 2024), we treat each NURBS surface as a language-aligned object: a sequence of tokens encoding control points, degrees, weights, and knot vectors in JSON format. This allows us to formulate text-to-CAD as a language modeling task. We fine-tune a large language model (Qwen3-4B) to map textual descriptions to these NURBS parameters, producing outputs that are editable and directly compatible to BRep format. To support this, we construct **partABC**, a curated dataset of more than 300k part-level CAD models from the ABC dataset, each represented as a sequence of NURBS surfaces and serialized with manageable context lengths ($\leq$ 8k tokens). We also generate high-quality natural language descriptions of the CAD models using an automatic annotation pipeline for the supervised fine-tuning task.

A key design choice to manage context length is our use of untrimmed NURBS surfaces similar to NeuroNURBS (Fan et al. 2024). However, this introduces a limitation it cannot capture trimmed geometry precisely. To address this, we propose a hybrid symbolic representation that replaces NURBS with primitive analytic curves (e.g., circles, B-splines, arcs, and lines) to accurately model such faces. This maintains the structural format required for LLM fine-tuning and inference. Our experiments demonstrate that NURBGen can outperform the current state-of-the-art methods in high-fidelity text-to-CAD generation as shown in Figure 1. Our contributions can be summarized as follows

- We propose NURBGen, the first framework for LLM-driven NURBS-based text-to-CAD framework.
- We introduce partABC, a large-scale multi-modal dataset of 300k CAD parts from the ABC dataset with NURBS annotations and high-quality captions using an automatic annotation pipeline.
- We design a hybrid representation combining untrimmed NURBS with analytic primitives to accurately model trimmed and degenerate surfaces while maintaining structural compatibility for LLM fine-tuning.
- Our extensive experiments demonstrate NURBGen's superior performance over existing baselines.

Related Work

CAD Generation: Earlier approaches to CAD generation primarily focused on low-level geometry tasks such as surface fitting (Sharma et al. 2020; Liu et al. 2023), point cloud classification (Qi et al. 2017a,b), BRep segmentation (Dupont et al. 2022; Lee et al. 2023; Mallis et al. 2023), or BRep structure prediction (Guo et al. 2022; Ali, Khan, and Stricker 2024), rather than generating fully parametric CAD models. A major shift came with DeepCAD (Wu, Xiao, and Zheng 2021), which introduced a design-history-based representation where CAD models are expressed as sequences of 2D sketches and 3D operations (e.g., extrusions). This formulation enabled sequence-to-sequence modeling of CAD generation. Building on this, later works explored cross-modal CAD synthesis from point clouds (Khan et al. 2024a; Dupont et al. 2024; Rukhovich et al. 2024), images (Chen et al. 2024), natural language (Li et al. 2025; Lv and Bao 2025; Khan et al. 2024b; Li et al. 2024; Govindarajan et al. 2025; Wang et al. 2025), or combinations thereof (Kolodiazhnyi et al. 2025; Xu et al. 2025). While design-history representations are highly interpretable and editable, their reliance on proprietary CAD operation data presents a major bottleneck for large-scale public research. Public datasets like DeepCAD-170k (Wu, Xiao, and Zheng 2021), Fusion360-8k (Willis et al. 2021), and CADParser-50k (Zhou, Tang, and Zhou 2023) are limited in size and complexity, often consisting of simple, synthetic parts (Govindarajan et al. 2025), which restricts generalization to real-world scenarios. Alternative approaches for CAD generation operate directly on BRep geometry (Lambourne et al. 2021) or leverage SDF supervision (Ren et al. 2022; Li et al. 2023; Yu et al. 2022). However, these methods don't generalizes well. Recent work BrepGen (Xu et al. 2024b), for example, generates BRep topology including vertices, edges, and faces via a hierarchical latent diffusion model. In contrast, we represent BReps as sequences of structured NURBS surfaces, allowing us to frame text-to-CAD as a language generation task. This enables fine-tuning an LLM on partABC, which is more diverse and larger than those used in prior work.

Nurbs Modeling: The adoption of analytic surfaces like NURBS in learning-based systems remained limited (Böhm, Farin, and Kahmann 1984; Mykhaskiv et al. 2018). NURBSDiff (Prasad et al. 2022) introduced differentiable NURBS fitting for geometry optimization and reconstruction. (Worchel and Alexa 2023) proposed differentiable rendering of NURBS surfaces for inverse graphics tasks. The most relevant prior work is NeuroNURBS (Fan et al. 2024), which encodes untrimmed NURBS surfaces using a non-autoregressive transformer autoencoder into latent vectors for supporting tasks like reconstruction or segmentation, but not text-conditioned generation. However, the exclusive use of untrimmed NURBS surfaces limits generalization, as not all CAD models can be accurately represented without trim-

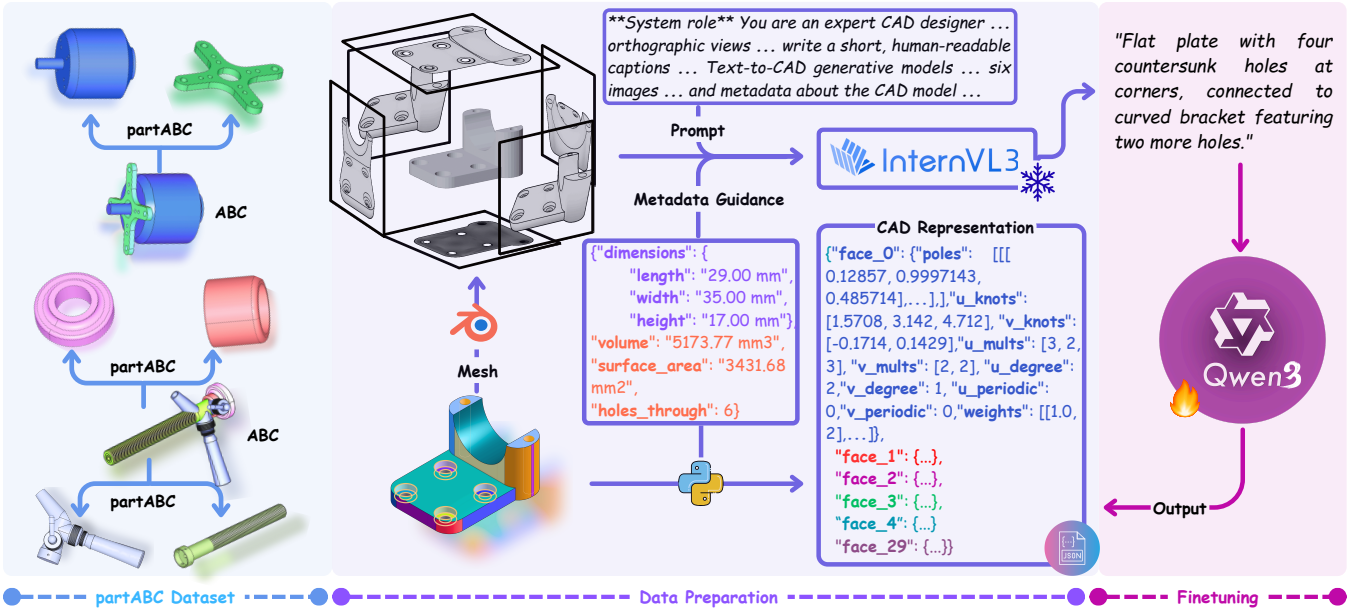


Figure 2: Overview of our partABC dataset, data preparation and fine-tuning pipeline. **Left:** We extract part-level CAD models from the ABC dataset by decomposing CAD assemblies into individual components. **Middle:** Each part is represented using a hybrid format—faces are encoded as untrimmed NURBS surfaces, with analytic primitives used where NURBS fitting fails. We also generate high-quality captions using InternVL3-13B with a metadata-guided annotation pipeline. **Right:** We fine-tune Qwen3-4B to map text captions to structured hybrid CAD representations, which can be directly converted to BRep models.

ming. To address this, we adopt a hybrid strategy: while untrimmed NURBS serve as our primary representation, we replace them with analytic primitives such as lines, arcs, and B-splines for faces where NURBS fitting fails.

LLM for 3D Generation: LLMs have been widely adopted across domains such as robotics (Zeng et al. 2023) and 3D scene understanding or grounding (Xu et al. 2024a; Hong et al. 2023). Their application to 3D generation is a relatively new but promising direction, as LLMs offer strong spatial priors and multimodal reasoning capabilities. A central challenge, however, lies in encoding 3D geometry into a sequential format compatible with language modeling. Recent work such as LLaMA-Mesh (Wang et al. 2024) fine-tunes LLaMA (Grattafiori and et al 2024) to generate mesh vertices and faces as plain text, demonstrating the potential of autoregressive text-based 3D synthesis. In the CAD domain, existing LLM-driven text-to-CAD methods primarily rely on design-history-based representations (Li et al. 2025; Rukhovich et al. 2024; Xu et al. 2025; Zhang et al. 2025). However, these methods are constrained by the scarcity and simplicity of the public datasets. In contrast, our method introduces a structured NURBS-based representation that enables symbolic, surface-level generation. This formulation aligns naturally with language modeling and allows us to leverage the large-scale and geometrically diverse partABC dataset for fine-tuning LLMs for the text-to-CAD task.

Background

Before presenting our method, we briefly review the fundamentals of NURBS. Non-Uniform Rational B-Splines (NURBS) are the standard representation for curves and surfaces in CAD and geometric modeling. They extend B-splines by assigning weights to control points, enabling both free-form and analytic shapes (e.g., circles, ellipses). Their compactness, smoothness, and precise parametric control make them central to modern CAD systems. A NURBS curve of degree p is defined by:

- A set of $n+1$ control points $\{\mathbf{P}_i \in \mathbb{R}^d\}_{i=0}^n$ with weights $\{w_i \in \mathbb{R}^+\}_{i=0}^n$,
- Knot vector $\mathbf{U} = \{u_0, \ldots, u_{n+p+1}\}$, $u_i \geq u_j, \forall i > j$
- Basis functions $N_{i,p}(u)$ defined recursively.

The NURBS curve is then given by:

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}, \quad u \in [u_p, u_{n+1}] \tag{1}$$

The B-spline basis functions $N_{i,p}(u)$ are defined recursively using the Cox-de Boor formula:

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1}, \\ 0 & \text{otherwise}, \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) \\ &\quad + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \end{aligned} \tag{2}$$

A NURBS surface is defined similarly, as the tensor product of two NURBS curves in parameters u and v. Given control points $\mathbf{P}_{ij}$, weights w_{ij}, knot vectors $\mathbf{U}$ and $\mathbf{V}$, and degrees

p, q, the NURBS surface is:

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)M_{j,q}(v)w_{ij}\mathbf{P}_{ij}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)M_{j,q}(v)w_{ij}}, \quad (3)$$
$$(u,v) \in [u_p, u_{n+1}] \times [v_q, v_{m+1}]$$

Here, $N_{i,p}(u)$ and $M_{j,q}(v)$ are the B-spline basis functions as defined in Eq. 2 in the u- and v-directions, respectively.

Data Preparation

In this section, we describe our data preparation pipeline, illustrated in Figure 2 (left and middle column). Our objective is to extract a NURBS-based surface representation for a BRep model in JSON format, along with a high-quality textual caption, which will serve as supervision to fine-tune an LLM for precise and editable text-to-CAD generation. To this end, we construct a new dataset, **partABC**, derived from the unlabeled, assembly-level ABC dataset. The following subsections detail our NURBS representation format and explain the motivation and processing steps used to build partABC.

1. CAD Representation

A BRep solid models geometry as a collection of topologically connected faces, each defined by a bounded parametric surface. In modern CAD systems, these surfaces are most commonly represented using NURBS surfaces due to their ability to accurately model both analytic primitives (e.g., planes, cylinders, tori) and complex free-form geometry with high continuity and compactness. To reconstruct a BRep solid in a symbolic and editable form, it is essential to extract the full set of NURBS surface parameters for each face. Using pythonOCC, we propose a robust pipeline for converting BReps into parametric NURBS representation.

Given a BRep solid, we begin by normalizing the geometry to fit within a $2 \times 2 \times 2$ bounding box centered at the origin, ensuring consistent scale and alignment across all samples. We then apply `BRepBuilderAPI_NurbsConvert` to convert each face into its untrimmed NURBS representation. This step standardizes all underlying analytic and freeform surfaces—such as planes, cylinders, and spline patches—into rational B-splines, providing a uniform surface representation. Next, we traverse each face using `TopExp_Explorer` and extract its surface parameters via the `Geom_BSplineSurface` API. For each face, we retrieve the control points (also called poles), knot vectors in both parametric directions, knot multiplicities, degrees in u and v, rational weights, and periodicity flags. Knot multiplicities specify how many times each knot value appears in the knot vector. Periodicity flags indicate whether the surface is seamlessly closed in the u and/or v direction, as in cylindrical or toroidal geometries. With all these parameters extracted, the original surface can be exactly reconstructed using the `Geom_BSplineSurface` constructor.

However, Not all surfaces can be robustly represented by untrimmed NURBS. In particular, thin regions around holes or fillets often introduce geometric artifacts or reconstruction errors (see Fig. 3). To address such degenerate cases,

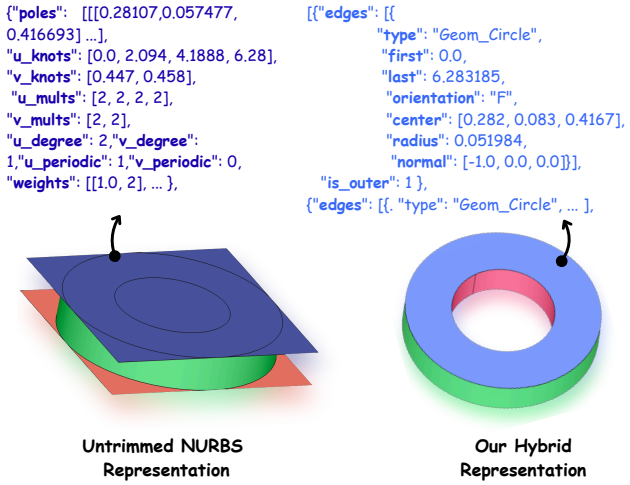


Figure 3: Our proposed hybrid representation. **Left:** Untrimmed NURBS surfaces introduce artifacts in hole-like or thin regions. **Right** We resolve this by substituting their NURB representation with analytic curves (*e.g.*, lines, circles) for improved geometric fidelity.

we adopt a hybrid representation: instead of enforcing a NURBS fit, we revert to simpler analytic primitives such as lines, circles, B-splines, ellipses, parabolas, and hyperbolas. These primitives are extracted from the original BRep faces prior to NURBS conversion. We detect degenerate or poorly reconstructed faces by comparing each reconstructed surface f_n with its ground-truth counterpart f_{gt} using the Chamfer Distance (CD) between their sampled point clouds ($CD(f_n, f_{gt}) \leq \epsilon$). CD measures the average squared distance from points in one set to their nearest neighbors in another. If it is below a threshold ϵ, the NURBS approximation is deemed acceptable; otherwise, we retain the original analytic primitive. We empirically set $\epsilon = 6 \times 10^{-4}$.

We represent each face using either its extracted NURBS parameters or its analytic primitive definition, based on reconstruction quality. In practice, about 70% of faces are modeled using NURBS, while 30% fall back to analytic primitives. This hybrid representation stored in structured JSON offers a more expressive and compact alternative to the purely NURBS-based format used in NeuroNURBS (Fan et al. 2024). While analytic primitives can represent simple geometry, they lack the flexibility to capture free-form surfaces. NURBS, on the other hand, provide a unified framework that can model both standard analytic shapes and complex free-form surfaces within a single patch—often replacing multiple primitives such as segmented arcs or partial cylinders. By combining both representations, our hybrid approach improves robustness and reduces parameter count for simpler shapes, resulting in shorter and more token-efficient inputs for LLM fine-tuning.

2. Annotation Pipeline

Supervised fine-tuning of our text-to-CAD model requires paired textual descriptions, but the ABC dataset lacks cap-

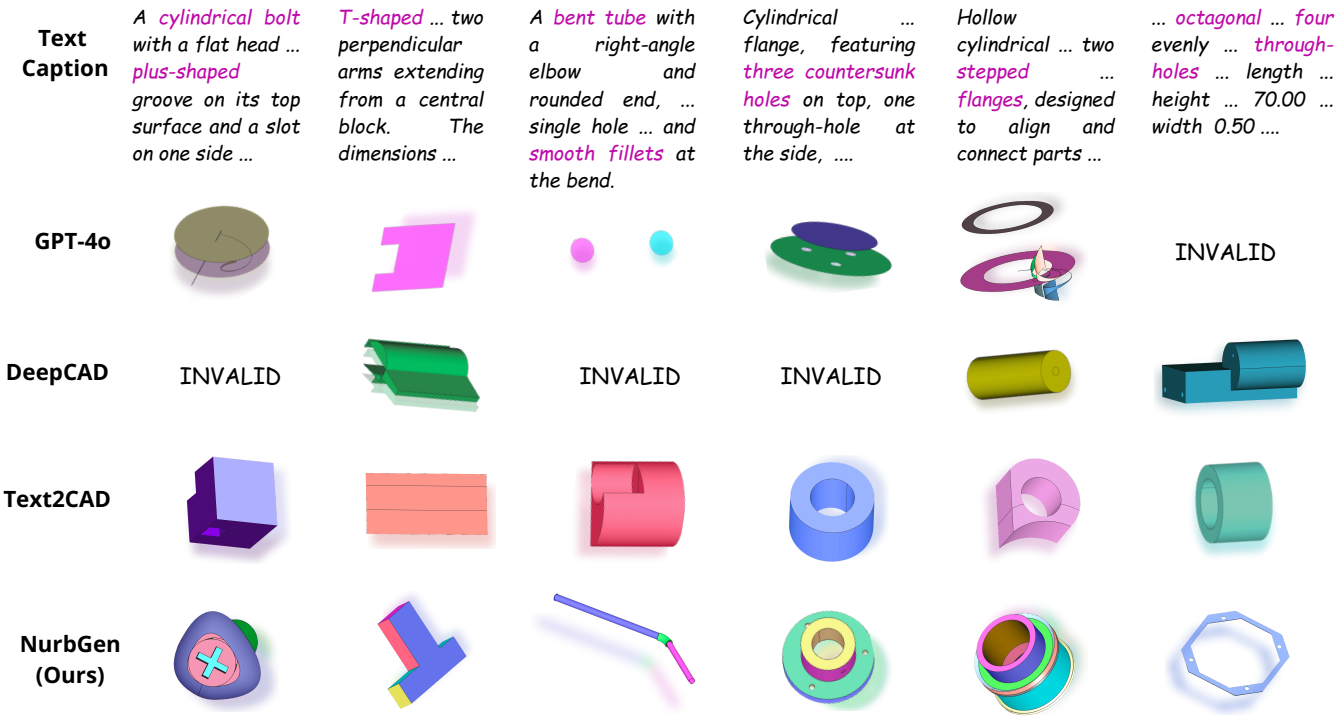


Figure 4: Qualitative comparison of reconstructed CAD models from text prompts. From top to bottom, we show generations from GPT-4o, DeepCAD, Text2CAD, and our proposed NURBGen. NURBGen consistently produces more detailed and structurally coherent results, with higher fidelity to the input prompt and fewer geometric artifacts compared to baselines.

tions. To address this, we design an automated annotation pipeline using a VLM to generate high-quality captions for CAD models at scale as shown in Figure. 2 (middle).

Multi-View Rendering: Each BRep is first converted into a textureless triangular mesh and rendered from six viewpoints at a resolution of 512×512 using Blender. Four of the camera views follow the orientation strategy proposed in (Sinha and Khan et al. 2025), while the remaining two capture the top and bottom perspectives. To enhance geometric perception and visual clarity, we enable Blender's Freestyle renderer to overlay clean silhouette and edge contours on each image.

Metadata Guidance for Caption Generation: High-quality captions for CAD models should go beyond simple naming and incorporate essential geometric features such as the number of through-holes, overall dimensions, surface area, and volume. Previous work like Text2CAD (Khan et al. 2024b) leverages minimal JSON-based design history to guide vision-language models (VLMs), while MARVEL (Sinha and Khan et al. 2025) uses structured metadata for fine-grained 3D annotation. Building on these ideas, we extract geometric metadata that is often inaccessible to VLMs—specifically, length, width, height, surface area, volume, and the number of topological holes (genus). We compute overall dimensions by fitting an axis-aligned bounding box to the CAD geometry using OpenCascade's `Bnd_Box`. Volume and surface area are obtained using OpenCascade's built-in mass property computation (`brepgprop.VolumeProperties` and `SurfaceProperties`).

To estimate the number of through-holes, we first generate a watertight mesh from BRep. We then compute the Euler characteristic, $\chi = V - E + F$, where V, E, F are the mesh vertices, edges, and faces. Using the Euler–Poincaré formula for closed 2-manifolds, the genus is given by $g = 0.5 \times (2 - \chi)$, which corresponds to the number of topological through-holes (Hliněný 2021). This metadata is then injected into the annotation prompt, which guides the VLM to generate captions with precise measurements.

Caption Generation: We use InternVL3-13B (Zhu et al. 2025), a multi-view VLM, which takes six rendered views of the CAD model along with the metadata-augmented annotation prompt as input. It then processes multi-view images of the CAD model simultaneously to generate a coherent and geometry-aware caption. Rather than focusing solely on object category names, we prioritize shape-centric descriptions that capture structural characteristics (*"a bent tube.."*,.. *cylindrical bolt .. six holes..*). The inclusion of dimensional metadata and hole counts further grounds the captions in precise geometric details, resulting in more informative and reliable annotations as shown in Figure 5.

3. partABC Dataset

In this section, we describe the construction of the partABC dataset as shown in Figure 2 (Left Column). While our data processing pipeline supports any CAD model, we focus on

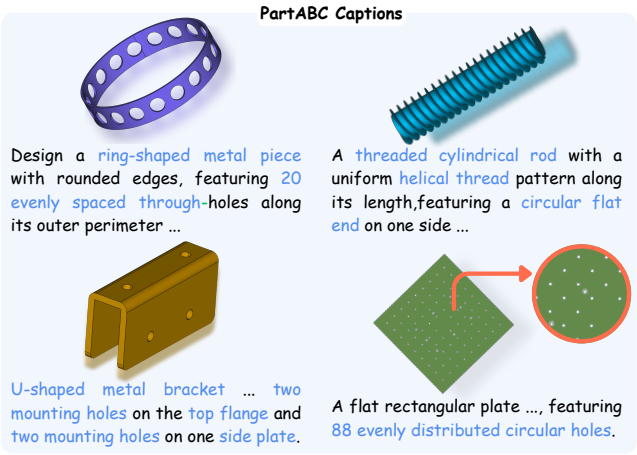


Figure 5: Captions from partABC dataset generated using our automatic captioning pipeline.

the ABC dataset due to its large scale and geometric diversity. ABC contains 1M CAD models, but processing the full set is computationally expensive and time-consuming. Therefore, we limit our preprocessing to 200k models for this project. However, many of these are assembly-level designs with a large number of faces, resulting in JSON representations that can exceed 100k tokens well beyond the context window and training budget of our project. To address this, we leverage the fact that BReps in ABC often encode part-level substructures within these assemblies. Using PythonOCC, we programmatically extract these individual parts, each representing a self-contained and geometrically coherent component. From the 200k processed assemblies, this provides us with 3M part-level CAD instances.

Filtering: However, extracting part-level CAD models from larger assemblies introduces a key challenge: many of the resulting shapes tend to be geometrically simple such as cuboids or cylinders. This can lead to an imbalanced training set and bias the fine-tuned LLM toward generating trivial geometry. To address this, we apply a complexity-aware filtering strategy using a weighted scoring function that prioritizes geometrically rich and structurally diverse parts. Each part-level model is scored using

$$\mathrm{w}(B) = l_1 \times \texttt{token\_count} + l_2 \times \texttt{through\_holes} + l_3 \times \frac{\texttt{surface\_area}}{\texttt{volume}} + l_4 \times \texttt{bbox\_diag}$$

where l_1=0.35, l_2=0.3, l_3=0.25, l_4=0.1 are selected using empirical experiment on 100 samples. `token_count` refers to the size of tokens after tokenizing the JSON using Qwen3 Tokenizer (Yang et al. 2025), `through_holes` counts the number of holes that pass through the entire part, and `bbox_diag` is the length of the diagonal of the part's axis-aligned bounding box. Based on $\mathrm{w}(B)$, we categorize parts into simple (≤ 0.12), moderate (0.12–0.23), and complex (> 0.23) tiers. From 3M extracted parts, we retain 10% simple, 50% moderate, and 40% complex models, forming the final partABC dataset of ∼300k samples (Figure 6).

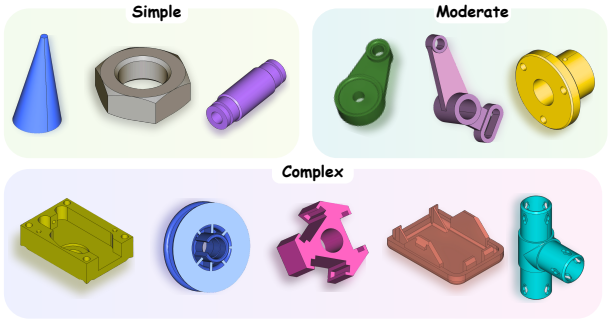


Figure 6: Example CAD parts from the partABC dataset across complexity tiers- simple (top-left), moderate (top-right), and complex (bottom).

Experimental Results

In this section, we provide details of our experiments and discuss evaluation results with baselines.

Datasets: We use the curated partABC dataset for supervised fine-tuning of our model, with 95%–2.5%-2.5% split for training, validation, and testing. To reduce context length and improve token efficiency, we round the NURBS control point coordinates to 6 decimal places. Additionally, we compress control point weights using a *(value, frequency)* representation scheme.

Implementation Details: We fine-tune Qwen3-4B model (Yang et al. 2025) using AdamW (Loshchilov and Hutter 2019) with a learning rate of 5×10^{-5} and linear warm-up. LoRA (Hu et al. 2021) is applied with rank 64 and α=128. Training runs for 180k steps with batch size 1 on 4×H200 GPUs over 3 days. The context window is 8192 during training and 14k during inference, with temperature 0.3. On RTX 3090, the model achieves a generation throughput of ∼ 800 tokens per second. Figure 2 (Right column) shows the finetuning task.

Baselines: We compare against strong open-source baselines for text-to-CAD generation. While recent models like CAD-LLaMA (Li et al. 2025) and CADFusion (Wang et al. 2025) report promising results, their implementations are not publicly available. Moreover, to the best of our knowledge, there are no open-source models capable of generating NURBS-based CAD representations from text. Hence, we focus our comparison with open-source methods, including Text2CAD (Khan et al. 2024b), DeepCAD (Wu, Xiao, and Zheng 2021), and GPT-4o. We use official pretrained weights for Text2CAD and retrain DeepCAD for 100 epochs following the Text2CAD protocol. GPT-4o is evaluated using 2-shot prompting with example caption–JSON pairs.

Metrics: We assess both geometric fidelity and visual alignment of the generated CAD models. For geometry, we compute Chamfer Distance (CD), Hausdorff Distance (HD), Jensen–Shannon Divergence (JSD), and Minimum Matching Distance (MMD) on 7,500 test samples using 8,192 uniformly sampled points normalized to a unit cube. For visual alignment, we evaluate prompt fidelity through human and GPT-4o preference studies on 1k and 5k samples, respec-

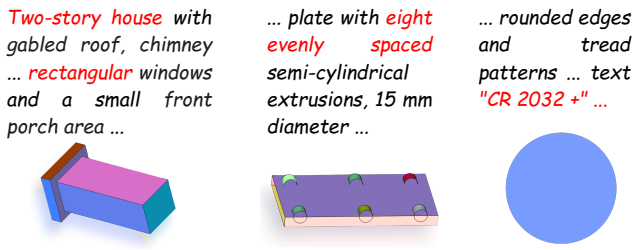


Figure 7: Failure cases of NURBGen illustrating limitations in handling complex prompts, geometric artifacts like self-intersections, and challenges in text engraving.

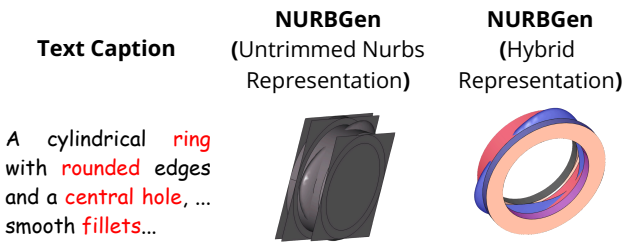


Figure 8: Comparison of NURBS-only (left) and hybrid (right) models, showing improved handling of thin and hole-adjacent regions.

tively. In the human study, five CAD designers of varying expertise choose the reconstruction that best matches each prompt, and we report Top-1 accuracy via majority vote. For GPT-4o, we present a 2×2 grid of multi-view renderings and the prompt, asking it to select the most faithful output or label cases as "Undecided." We additionally report the Invalidity Ratio (IR), defined as the percentage of generated models that fail to convert into valid B-Rep structures.

Model	User(1k)↑	GPT↑	IR↓	CD↓	HD↓	JSD↓	MMD↓
Undecided	2.7	3.2	–	–	–	–	–
GPT-4o	1.5	1.9	0.17	7.2	0.36	72.87	4.17
DeepCAD	5.6	6.1	0.32	10.28	0.45	89.77	4.43
Text2CAD	26.1	27.2	0.05	9.66	0.42	85.27	4.54
NURBGen	**64.1**	**61.6**	**0.018**	**4.43**	**0.25**	**57.94**	**2.14**

Table 1: Quantitative comparison of text-to-CAD models. CD, JSD and MMD are multiplied by 10^2.

Results: Table 1 lists the quantitative comparison of NURBGen with other baselines. Our model outperforms the prior baselines by a significant margin in both geometric and visual alignment. Notably, we achieve 60.8% top-1 preference in human evaluation and 63.7% in GPT-4o evaluation. Text2CAD ranks 2nd, followed by DeepCAD and GPT-4o. Notably, NURBGen also has the lowest invalidity ratio (0.01), indicating strong geometric correctness in its output. In contrast, DeepCAD suffers from a higher rate (0.3), reflecting challenges in generating complete and consistent BRep geometry. As shown in Fig 4, NURBGen generates CAD models that are better aligned with the input text, outperforming baselines in both fidelity and consistency.

Caption Quality: Because VLMs can occasionally hallucinate (Liu et al. 2024), we evaluate the reliability of our automatically generated captions. We randomly sample 1,000 captions and provide each, together with its 6 rendered views and CAD metadata, to GPT-4o for verification. GPT-4o reports an accuracy of $\sim$ 85%, indicating that our captioning pipeline produces generally accurate and semantically meaningful descriptions.

Ablation Study

We perform an ablation study to assess the contribution of our hybrid representation. We fine-tune Qwen3-4B using only the untrimmed NURBS representation, removing the fallback to analytic primitives, and evaluate it with both human judges and GPT-4o following the same protocol as before. The hybrid model achieves 72% (human) and 79% (GPT-4o), whereas the NURBS-only model exhibits noticeable geometric artifacts—especially around holes, sharp transitions, and regions where NURBS fitting is less precise (Figure 8). These results demonstrate that the hybrid design is crucial for stable and accurate CAD reconstruction.

Limitation

Despite its strong performance, our approach has certain limitations. Figure 7 illustrates a few representative failure cases. For instance, in response to complex prompts (e.g., "*Two-story house with gabled roof...*"), NURBGen struggles to capture fine-grained architectural structure. In rare cases, we also observe geometric artifacts such as self-intersections or topological inconsistencies, as seen in the second example. Additionally, NURBGen has difficulty reconstructing prompts with engraving text (third example).

Conclusion

We present **NURBGen**, the first framework for text-to-CAD generation using NURBS surfaces. NURBGen generates structured, editable NURBS representations from text prompts, which can be directly converted into B-Rep format using a fine-tuned LLM. To enable this, we generate **partABC**, a large-scale dataset of 300k part-level models from ABC with NURBS annotations and high-quality generated captions. We hope that this dataset will be a valuable resource for future research. We further propose a hybrid representation that combines untrimmed NURBS with analytic primitives to address trimming artifacts while enhancing geometric robustness and token efficiency. Empirical results show that NURBGen surpasses existing state-of-the-art methods in geometric fidelity, as confirmed by expert evaluators. While the current model is constrained by a context window of 8192, future work will explore long-context training and multimodal extensions to handle more complex assemblies. We believe that our work will position NURBS-based representations as a compelling alternative to design-history-based methods for future research in the evolving text-to-CAD domain.

Acknowledgements

This work was co-funded by the European Union under Horizon Europe, grant number 101135724, project LUMINOUS. However, the views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible.

References

Ali, S. A.; Khan, M. S.; and Stricker, D. 2024. BRep Boundary and Junction Detection for CAD Reverse Engineering. In *IEEE International Conference on Computing and Machine Intelligence (ICMI)*. IEEE.

Böhm, W.; Farin, G.; and Kahmann, J. 1984. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1(1): 1–60.

Chen, T.; Yu, C.; Hu, Y.; Li, J.; Xu, T.; Cao, R.; Zhu, L.; Zang, Y.; Zhang, Y.; Li, Z.; and Sun, L. 2024. Img2CAD: Conditioned 3D CAD Model Generation from Single Image with Structured Visual Geometry. arXiv:2410.03417.

Dupont, E.; Cherenkova, K.; Kacem, A.; Ali, S. A.; Arzhannikov, I.; Gusev, G.; and Aouada, D. 2022. CADOpsNet: Jointly Learning CAD Operation Types and Steps from Boundary-Representations. In *2022 International Conference on 3D Vision (3DV)*, 114–123.

Dupont, E.; Cherenkova, K.; Mallis, D.; Gusev, G.; Kacem, A.; and Aouada, D. 2024. TransCAD: A Hierarchical Transformer for CAD Sequence Inference from Point Clouds. arXiv:2407.12702.

Fan, J.; Gholami, B.; Bäck, T.; and Wang, H. 2024. NeuroNURBS: Learning Efficient Surface Representations for 3D Solids. arXiv:2411.10848.

Gao, W.; Zhang, Y.; Ramanujan, D.; Ramani, K.; Chen, Y.; Williams, C. B.; Wang, C. C.; Shin, Y. C.; Zhang, S.; and Zavattieri, P. D. 2015. The status, challenges, and future of additive manufacturing in engineering. *Computer-Aided Design*, 69: 65–89.

Govindarajan, P.; Baldelli, D.; Pathak, J.; Fournier, Q.; and Chandar, S. 2025. CADmium: Fine-Tuning Code Language Models for Text-Driven Sequential CAD Design. arXiv:2507.09792.

Grattafiori, A.; and et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.

Guo, H.; Liu, S.; Pan, H.; Liu, Y.; Tong, X.; and Guo, B. 2022. ComplexGen: CAD Reconstruction by B-Rep Chain Complex Generation. *ACM Trans. Graph. (SIGGRAPH)*, 41(4).

Hliněný, P. 2021. A Short Proof of Euler–Poincaré Formula. arXiv:1612.01271.

Hong, Y.; Zhen, H.; Chen, P.; Zheng, S.; Du, Y.; Chen, Z.; and Gan, C. 2023. 3D-LLM: Injecting the 3D World into Large Language Models. *NeurIPS*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Kapsalis, T. 2024. CADgpt: Harnessing Natural Language Processing for 3D Modelling to Enhance Computer-Aided Design Workflows. arXiv:2401.05476.

Khan, M. S.; Dupont, E.; Ali, S. A.; Cherenkova, K.; Kacem, A.; and Aouada, D. 2024a. CAD-SIGNet: CAD Language Inference from Point Clouds using Layer-wise Sketch Instance Guided Attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4713–4722.

Khan, M. S.; Sinha, S.; Uddin, S. T.; Stricker, D.; Ali, S. A.; and Afzal, M. Z. 2024b. Text2CAD: Generating Sequential CAD Designs from Beginner-to-Expert Level Text Prompts. In *Advances in Neural Information Processing Systems*, volume 37, 7552–7579. Curran Associates, Inc.

Koch, S.; Matveev, A.; Jiang, Z.; Williams, F.; Artemov, A.; Burnaev, E.; Alexa, M.; Zorin, D.; and Panozzo, D. 2019. ABC: A Big CAD Model Dataset For Geometric Deep Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kolodiazhnyi, M.; Tarasov, D.; Zhemchuzhnikov, D.; Nikulin, A.; Zisman, I.; Vorontsova, A.; Konushin, A.; Kurenkov, V.; and Rukhovich, D. 2025. cadrille: Multimodal CAD Reconstruction with Online Reinforcement Learning. *arXiv preprint arXiv:2505.22914*.

Lambourne, J. G.; Willis, K. D.; Jayaraman, P. K.; Sanghi, A.; Meltzer, P.; and Shayani, H. 2021. BRepNet: A Topological Message Passing System for Solid Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12773–12782.

Lee, J.; Yeo, C.; Cheon, S.-U.; Park, J. H.; and Mun, D. 2023. BRepGAT: Graph neural network to segment machining feature faces in a B-rep model. *Journal of Computational Design and Engineering*, 10(6): 2384–2400.

Li, J.; Ma, W.; Li, X.; Lou, Y.; Zhou, G.; and Zhou, X. 2025. CAD-Llama: Leveraging Large Language Models for Computer-Aided Design Parametric 3D Model Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, P.; Guo, J.; Zhang, X.; and ming Yan, D. 2023. SECAD-Net: Self-Supervised CAD Reconstruction by Learning Sketch-Extrude Operations. arXiv:2303.10613.

Li, X.; Song, Y.; Lou, Y.; and Zhou, X. 2024. CAD Translator: An Effective Drive for Text to 3D Parametric Computer-Aided Design Generative Modeling. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, 8461–8470. New York, NY, USA: Association for Computing Machinery. ISBN 9798400706868.

Liu, H.; Xue, W.; Chen, Y.; Chen, D.; Zhao, X.; Wang, K.; Hou, L.; Li, R.; and Peng, W. 2024. A Survey on Hallucination in Large Vision-Language Models. arXiv:2402.00253.

Liu, Y.; Obukhov, A.; Wegner, J. D.; and Schindler, K. 2023. Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds. arXiv:2312.04962.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Lv, C.; and Bao, J. 2025. CADInstruct: A multimodal dataset for natural language-guided CAD program synthesis. *Computer-Aided Design*, 188: 103926.

Mallis, D.; Aziz, A. S.; Dupont, E.; Cherenkova, K.; Karadeniz, A. S.; Khan, M. S.; Kacem, A.; Gusev, G.; and Aouada, D. 2023. SHARP Challenge 2023: Solving CAD History and pArameters Recovery from Point Clouds and 3D Scans. Overview, Datasets, Metrics, and Baselines. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 1786–1795.

Mykhaskiv, O.; Banović, M.; Auriemma, S.; Mohanamuraly, P.; Walther, A.; Legrand, H.; and Müller, J.-D. 2018. NURBS-based and parametric-based shape optimization with differentiated CAD kernel. *Computer-Aided Design and Applications*, 15(6): 916–926.

Prasad, A. D.; Balu, A.; Shah, H.; Sarkar, S.; Hegde, C.; and Krishnamurthy, A. 2022. NURBS-Diff: A Differentiable Programming Module for NURBS. *Computer Aided Design*, 146: 103199.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. arXiv:1612.00593.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv preprint arXiv:1706.02413*.

Ren, D.; Zheng, J.; Cai, J.; Li, J.; and Zhang, J. 2022. ExtrudeNet: Unsupervised Inverse Sketch-and-Extrude for Shape Parsing. arXiv:2209.15632.

Rukhovich, D.; Dupont, E.; Mallis, D.; Cherenkova, K.; Kacem, A.; and Aouada, D. 2024. CAD-Recode: Reverse Engineering CAD Code from Point Clouds. *arXiv preprint arXiv:2412.14042*.

Sharma, G.; Liu, D.; Kalogerakis, E.; Maji, S.; Chaudhuri, S.; and Měch, R. 2020. ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds. arXiv:2003.12181.

Sinha, S.; Khan, M. S.; Usama, M.; Sam, S.; Stricker, D.; Ali, S. A.; and Afzal, M. Z. 2025. MARVEL-40M+: Multi-Level Visual Elaboration for High-Fidelity Text-to-3D Content Creation. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 8105–8116.

Vido, M.; de Oliveira Neto, G. C.; Lourenço, S. R.; Amorim, M.; and Rodrigues, M. J. F. 2024. Computer-Aided Design and Additive Manufacturing for Automotive Prototypes: A Review. *Applied Sciences*, 14(16).

Wang, R.; Yuan, Y.; Sun, S.; and Bian, J. 2025. Text-to-CAD Generation Through Infusing Visual Feedback in Large Language Models. In *Forty-second International Conference on Machine Learning*.

Wang, Z.; Lorraine, J.; Wang, Y.; Su, H.; Zhu, J.; Fidler, S.; and Zeng, X. 2024. LLaMA-Mesh: Unifying 3D Mesh Generation with Language Models. *arXiv preprint arXiv:2411.09595*.

Willis, K. D. D.; Pu, Y.; Luo, J.; Chu, H.; Du, T.; Lambourne, J. G.; Solar-Lezama, A.; and Matusik, W. 2021. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences. *ACM Transactions on Graphics (TOG)*, 40(4).

Worchel, M.; and Alexa, M. 2023. Differentiable Rendering of Parametric Geometry. *ACM Transactions On Graphics.*, 42(6).

Wu, R.; Xiao, C.; and Zheng, C. 2021. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 6772–6782.

Xu, J.; Zhao, Z.; Wang, C.; Liu, W.; Ma, Y.; and Gao, S. 2025. CAD-MLLM: Unifying Multimodality-Conditioned CAD Generation With MLLM. arXiv:2411.04954.

Xu, R.; Wang, X.; Wang, T.; Chen, Y.; Pang, J.; and Lin, D. 2024a. PointLLM: Empowering Large Language Models to Understand Point Clouds. In *European Conference on Computer Vision*.

Xu, X.; Lambourne, J.; Jayaraman, P.; Wang, Z.; Willis, K.; and Furukawa, Y. 2024b. Brepgen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4): 1–14.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; and et al. 2025. Qwen3 Technical Report. arXiv:2505.09388.

Yang, S.; Wang, J.; and Wang, K. 2024. NURBS-OT: An Advanced Model for Generative Curve Modeling. *Journal of Mechanical Design*, 147(3): 031703.

Yu, F.; Chen, Z.; Li, M.; Sanghi, A.; Shayani, H.; Mahdavi-Amiri, A.; and Zhang, H. 2022. CAPRI-Net: Learning Compact CAD Shapes With Adaptive Primitive Assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11768–11778.

Zeng, F.; Gan, W.; Wang, Y.; Liu, N.; and Yu, P. S. 2023. Large Language Models for Robotics: A Survey. arXiv:2311.07226.

Zhang, L.; Le, B.; Akhtar, N.; Lam, S.-K.; and Ngo, T. 2025. Large Language Models for Computer-Aided Design: A Survey. arXiv:2505.08137.

Zhou, S.; Tang, T.; and Zhou, B. 2023. CADParser: a learning approach of sequence modeling for B-Rep CAD. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23. ISBN 978-1-956792-03-4.

Zhu, J.; Wang, W.; Chen, Z.; Liu, Z.; Ye, S.; Gu, L.; Tian, H.; Duan, Y.; Su, W.; Shao, J.; Gao, Z.; Cui, E.; Wang, X.; Cao, Y.; Liu, Y.; Wei, X.; Zhang, H.; Wang, H.; Xu, W.; Li, H.; Wang, J.; Deng, N.; Li, S.; He, Y.; Jiang, T.; Luo, J.; Wang, Y.; He, C.; Shi, B.; Zhang, X.; Shao, W.; He, J.; Xiong, Y.; Qu, W.; Sun, P.; Jiao, P.; Lv, H.; Wu, L.; Zhang, K.; Deng, H.; Ge, J.; Chen, K.; Wang, L.; Dou, M.; Lu, L.; Zhu, X.; Lu, T.; Lin, D.; Qiao, Y.; Dai, J.; and Wang, W. 2025. InternVL3: Exploring Advanced Training and Test-Time Recipes for Open-Source Multimodal Models. arXiv:2504.10479.

Appendix

CAD Representation

We represent a BRep solid using a sequence of faces. Each face is either a NURB surface or contains analytical primitives (lines, circle, bsplines, and so on). Below, we provide how to parameterize them in our CAD representation. Figure 9 showcases an example CAD representation.

1. NURBS:

- `Poles`: Control points, represented as a 2D array of 3D points defining the control net of the surface.
- `Weights`: A 2D array of real numbers associated with the poles. Defines the rational nature of the surface. Optional for non-rational surfaces.
- `u_knots`, `v_knots`: Non-decreasing sequences of real numbers defining the knot vectors in the u and v parametric directions.
- `u_mults`, `v_mults`: Integer sequences representing the multiplicity of each knot in the u and v directions, respectively.
- `u_degree`, `v_degree`: Degree of the B-spline basis functions in the u and v directions.
- `u_periodic`, `v_periodic`: Boolean flags indicating whether the surface is periodic in each parametric direction.

2. Line

- `start`: (x, y, z) coordinates of the starting point.
- `end`: (x, y, z) coordinates of the ending point.

3. Circle

- `center`: (x, y, z) coordinates of the circle's center.
- `normal`: (x, y, z) direction vector normal to the plane of the circle.
- `radius`: Radius of the circle.
- `first`, `last`: Start and end angles (in radians) defining an arc on the circle. For a semicircle, `first = 0.0`, `last =` π.

4. Ellipse

- `center`: (x, y, z) coordinates of the ellipse's center.
- `normal`: (x, y, z) direction vector normal to the ellipse's plane.
- `major_radius`: Length of the major axis.
- `minor_radius`: Length of the minor axis.
- `first`, `last`: Start and end angles (in radians) defining an arc on the ellipse. For a full ellipse, `first = 0.0`, `last =` 2π.

5. Bezier Curve

- `poles`: List of control points (x, y, z).
- `degree`: Degree of the Bezier curve.
- `first`, `last`: Parametric domain range.

6. B-spline Curve

- `poles`: List of control points (x, y, z).
- `degree`: Degree of the B-spline curve.
- `knots`: Knot vector (non-decreasing real numbers).
- `mults`: Corresponding multiplicities of knots.
- `weights` (optional): Weights for rational B-splines (if omitted, assumed to be 1.0).
- `is_periodic`: Boolean flag indicating if the curve is periodic.
- `first`, `last`: Parametric range.

Impact of Metadata on Caption Quality

Figure 10 and 11 highlight the crucial role of metadata in driving accurate and informative captions. With access to structural cues such as *dimensions* and *hole count*, InternVL3-13B produces descriptions that are both precise and grounded in geometry. In contrast, GPT-4o, when prompted without metadata, often overlooks or misstates these critical features.

More Qualitative Results

Figure 12 and Figure 13 showcase additional results from our text-to-CAD generation using NURBGen. Notably, examples such as (Column 1, Row 1) and (Column 2, Row 3) in Figure 12, and (Column 1, Row 3) in Figure 13, show shapes that are currently infeasible to generate using existing design-history-based text-to-CAD approaches due to lack of CAD operations such as loft, sweep and revolution in the training datasets.

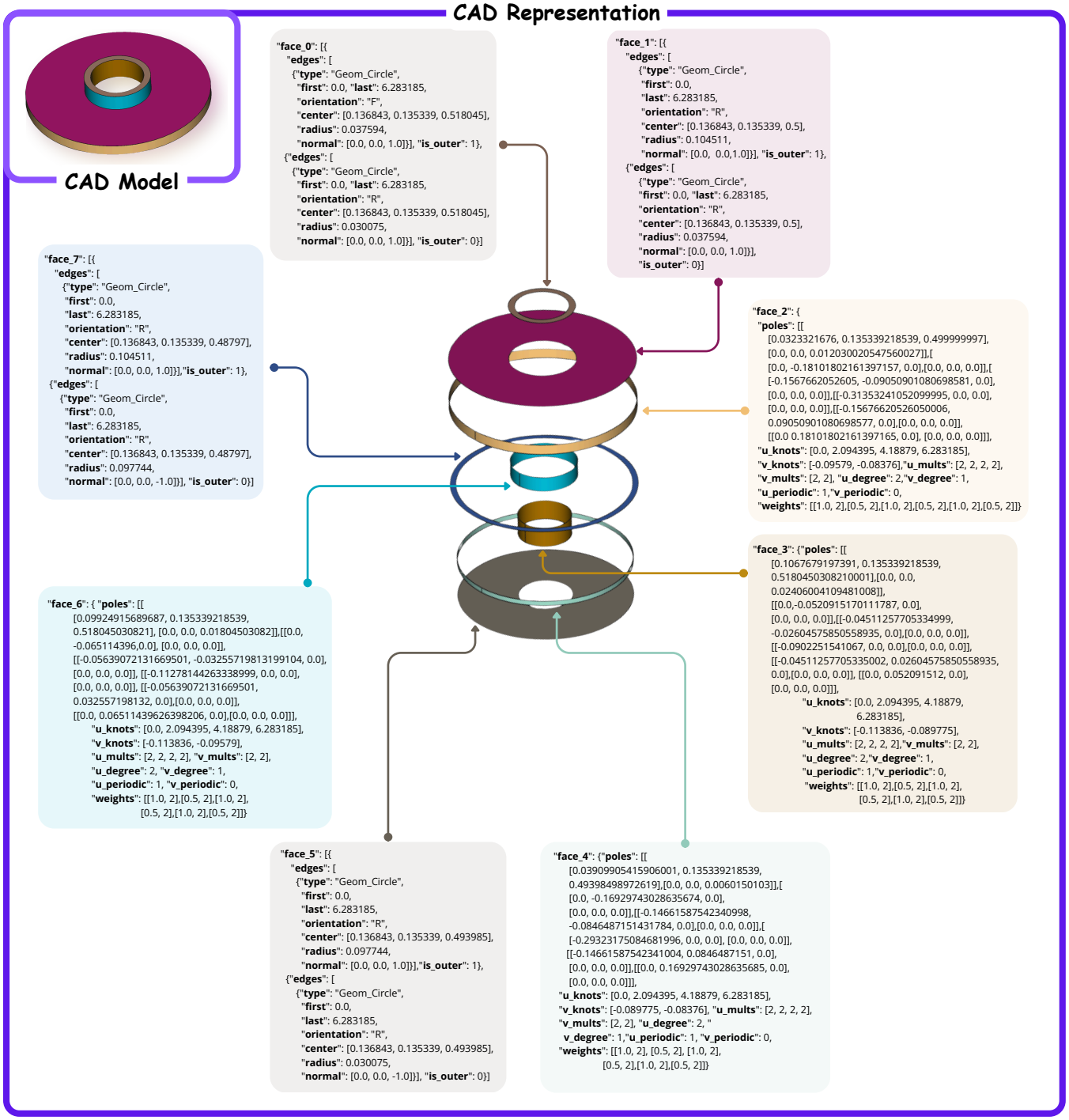


Figure 9: Our proposed hybrid CAD representation.

Figure 10: **Impact of metadata-guided annotation in our annotation pipeline**. From left to right: CAD model, caption generated by GPT-4o without metadata, and caption from our pipeline using metadata such as dimensions, and hole count.

PartABC

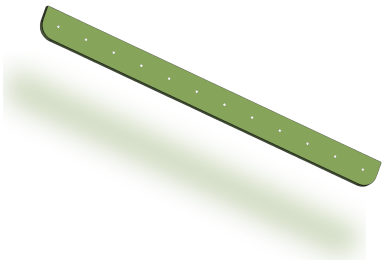
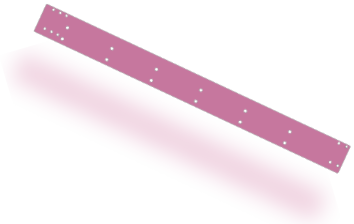
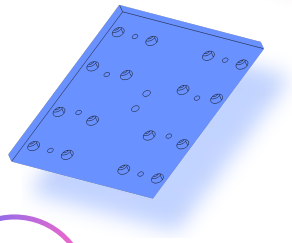
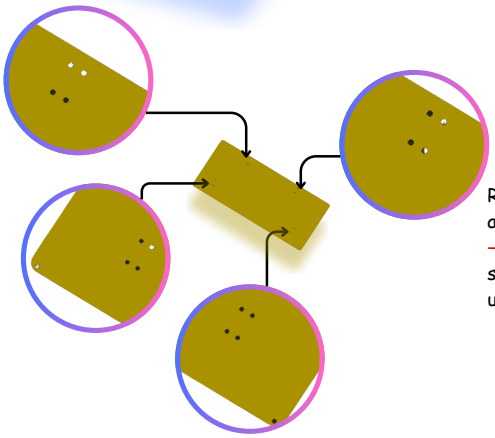
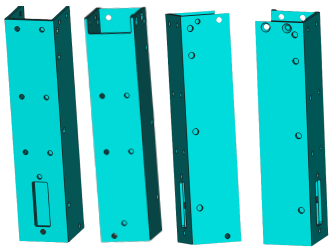
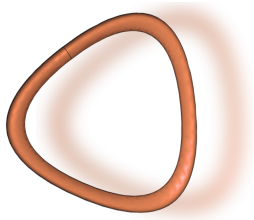
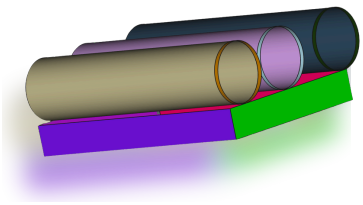
CAD Model	GPT-4o (Without Metadata)	Ours (With Metadata)
	Long, slender cylindrical rod featuring 10 evenly spaced circular holes along one side; opposite face is smooth with rounded ends and uniform diameter.	Rectangular metal bar with rounded ends, featuring 12 evenly spaced holes along its length, and smooth edges. Dimensions: 1506 mm long, 24 mm wide, 125 mm tall."
	Tapered rectangular plate with 10 circular through-holes in a linear array and 6 smaller holes clustered near each end. Flat top and bottom faces; uniform thickness.	Design a rectangular metal bar measuring 1369mm x 125mm x 10mm with 22 evenly spaced through-holes along its length. Ensure uniform hole distribution for structural attachment purposes.
	Rectangular plate with symmetrical pattern of through holes; 5 rows by 5 columns, with varying diameters concentrated toward edges and corners.	Rectangular plate with rounded corners, measuring 135x158x8mm. Features 26 holes: 16 with circles around them and 10 solid circles.
	Rectangular tray-like component with angled side walls and 16 circular holes—4 per corner—on the inclined inner surfaces; base is flat and unperforated.	Create a rectangular metal bracket with dimensions 1270mm x 584.20mm x 3.18mm. Include rounded edges with fillets, four corners, and 18 evenly distributed mounting holes along the perimeter.
	Hollow tapered rectangular column with open ends and 23 circular holes—8 on each longer face, 5 and 5 on the shorter faces—plus one rectangular cutout near the base.	Rectangular metal frame with dimensions 50.8mm x 50.8mm x 228.6mm. Features 29 through-holes. Includes mounting brackets at top corners.

Figure 11: **Impact of metadata-guided annotation in our annotation pipeline**. From left to right: CAD model, caption generated by GPT-4o without metadata, and caption from our pipeline using metadata such as dimensions, and hole count.

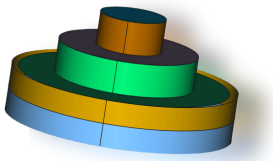
(Text-to-CAD using NURBGen)



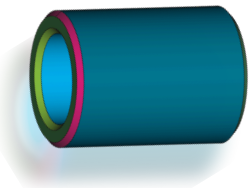
Closed triangular ring with smooth, rounded tubular profile, three equal curved edges, consistent diameter



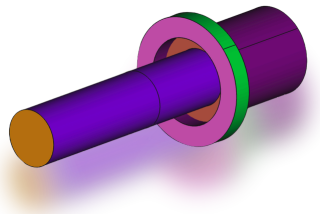
Rectangular block with three through-holes aligned horizontally on one face. Dimensions: Length 26.22 mm, width 3.18 mm, height 16.44 mm..



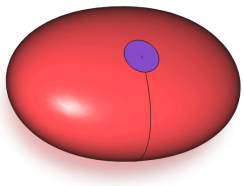
Cylindrical component with spherical caps on both ends, overall height 168.71 mm, diameter 22.71 mm. Features an elongated central section and a single horizontal slot at mid-height along its length.



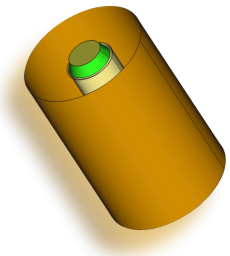
A concentric cylindrical assembly showing multiple nested components, illustrating precise fit and structural configuration.



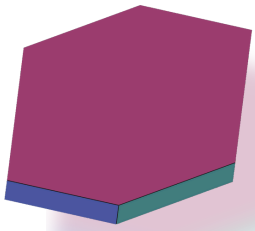
Assembly model featuring a stepped cylindrical shaft, fitted through a central ring and washer, terminating in a larger hollow cylinder.



Design a cylindrical component with hemispherical top and bottom ends. Include fillets at both ends. Dimensions: length 74.61 mm, width 107.95 mm, height 74.61 mm. Ensure smooth transitions between cylindrical and spherical sections.



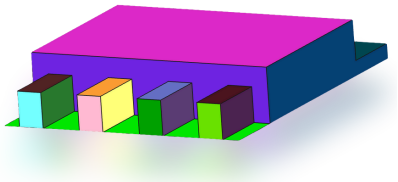
Cylindrical component with hemispherical ends and two flanged faces. Features rounded edges and symmetrical design. Total length 14.32 mm, diameter 7.90 mm at widest points. Fillets present on transitions between cylindrical and spherical sections. No visible holes or threading.



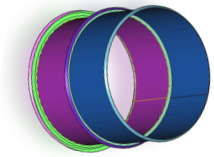
Design a hexagonal prism, measuring 39.91 mm in length, 38.44 mm in width, and 5.08 mm in height.

Figure 12: Text-to-CAD generation using NURBGen.

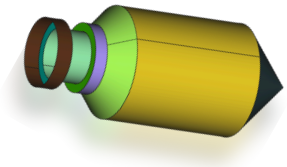
(Text-to-CAD using NURBGen)



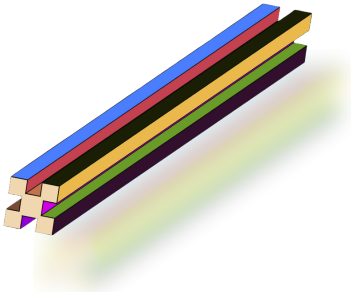
Rectangular body, four rectangular protrusions at corners on one side. Dimensions: length 5.00 mm, width 6.20 mm, height 1.10 mm. No through holes present. Flat top surface. Symmetrical layout.



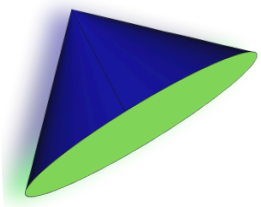
Create two concentric hollow cylinders with visible cross-section, varying wall thicknesses, chamfered edges, and aligned along a shared central axis.



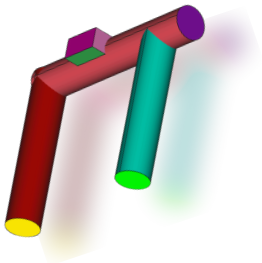
Create a cylindrical bottle with dimensions 26mm x 26mm x 62.3mm. Include a rounded cap on top. Add a horizontal groove near the neck.



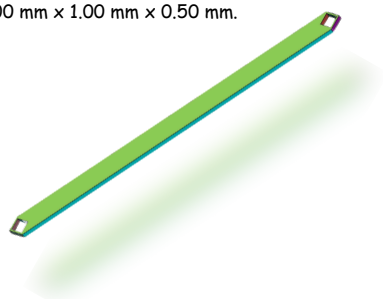
Design a 400mm-long bundle of interlocking, cross-shaped bars arranged in parallel. Maintain uniform spacing, and consistent wall thickness.



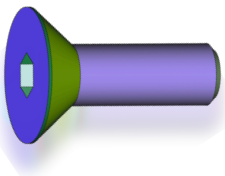
A pyramid with a flat circular base. The dimensions are 1.00 mm x 1.00 mm x 0.50 mm.



A T-shaped assembly with cylindrical legs and a horizontal beam featuring rectangular cutouts on the top tube.



Design a rectangular plate with dimensions 330.20 mm x 233.40 mm x 6.00 mm. Include two square through-holes near each end.



The CAD model is a tapered cylindrical shaft with a hexagonal base, featuring rounded edges and a conical tip. It has a central groove running along its length.

Figure 13: Text-to-CAD generation using NURBGen.