

Formalizing Machining Feature Knowledge for Feature-Based Design Automation in CAD

Elias Berger^{1,2}, Felix Pusch³, Kevin Herrmann³, Christian Becker³, Jan Mehlstäubl², Roland Lachmayer³, Kristin Paetzold-Byhain¹

¹Dresden University of Technology

²MAN Truck & Bus SE

³Leibniz University Hannover

Abstract: Parametric CAD models encode engineering design knowledge through feature-based representations, yet generative AI approaches to design automation treat these models as geometric shapes, discarding functional semantics and manufacturing constraints. Text-to-CAD methods rely on vision-language models to generate descriptions from renders, producing annotations that capture visual appearance but lack engineering content, limiting their ability to generate functionally valid outputs. We present a methodology for capturing machining feature knowledge in parametric CAD representations. Our approach enriches models with structured metadata linking geometric features to functional purposes and manufacturing requirements while maintaining traceability to design operations. We validate this on 1,770 shaft components with dual annotations—VLM-only baseline versus feature-augmented. Training generative models on feature-augmented data yields substantial improvements in geometric accuracy and functional correctness, demonstrating that structured engineering metadata enables better capture of design intent. We release our code for reproducibility with the camera-ready paper.

Keywords: computer-aided design, design automation, generative artificial intelligence, machining features

1 Introduction

Parametric CAD models encode engineering design knowledge through feature-based representations that capture functional requirements and manufacturing constraints. Knowledge-based engineering (KBE) approaches formalize this knowledge through template-based methods and rule-based systems (Bode et al., 2023; Herrmann et al., 2021), but remain constrained to predefined solution spaces requiring labor-intensive customization for novel requirements.

Recent generative AI approaches, particularly large language models (LLMs), offer new pathways for design automation by translating natural language into executable CAD code (Doris et al., 2025; Khan et al., 2024; Xu et al., 2025). However, current Text-to-CAD methods face a critical limitation: training data systematically omits engineering semantics that distinguish functional CAD models from geometric shapes (Berger et al., 2026). Existing annotation pipelines rely on vision-language models (VLMs) to generate descriptions from 2D renders capturing visual appearance but discarding machining features, dimensional specifications, and functional relationships that encode design intent. Consequently, models are evaluated using shape-similarity metrics that fail to assess functional validity or manufacturability.

Research Questions. We address this short coming by presenting a methodology for enriching CAD training data with explicit machining feature knowledge. This work is guided by two research questions: (RQ1) Can machining feature metadata be systematically extracted and incorporated into CAD training annotations in a reproducible manner? (RQ2) Does training on feature-augmented annotations improve model performance on geometric accuracy and functional correctness compared to VLM-only baselines?

Building on established feature recognition methods (Pusch et al., 2026a, 2026b), our approach generates training data linking geometric features to functional purposes and manufacturing requirements while maintaining traceability to originating CAD operations. We demonstrate feasibility through a proof-of-concept dataset of 1,770 shaft components with dual annotations, baseline VLM-only and feature-augmented, enabling controlled evaluation of annotation strategies.

Contributions. To answer the research questions, we contribute: (1) a systematic, reproducible methodology for incorporating machining feature metadata into CAD training data; (2) a proof-of-concept dataset with parallel annotations demonstrating practical implementation; and (3) empirical validation showing that models trained on feature-augmented data achieve substantial improvements in geometric accuracy and functional correctness.

2 Background and Related Work

2.1 Generative Methods for CAD

This section reviews learning-based approaches for CAD generation with a focus on two orthogonal dimensions: training strategy and input modality. Recent approaches leverage large language models (LLMs) to generate parametric CAD

models, motivated by their strong performance in code and sequence generation (Radford et al., 2019). These methods can be systematically categorized along two axes: (i) the *training method* and (ii) the *CAD data representation*. Training strategies range from zero-shot prompting to fine-tuning and full training from scratch, while representations include JSON-based formats, Python code (e.g., CadQuery), and CAD-specific tokenized sequences (Wu et al., 2021). A structured overview is provided in Table 1.

Table 1: Comparison of CAD generation approaches using LLMs grouped by training method and data representation.

Training Method	Data Representation	Publication
Zero-Shot	Code	(Alrashedy et al., 2025; Mallis et al., 2025; Ocker et al., 2025)
Fine-Tune	JSON	(Wang et al., 2025; Xu et al., 2025)
	Code	(Doris et al., 2025; He et al., 2025)
Full Training	Tokens	(Alam and Ahmed, 2025; Khan et al., 2024)

2.1.1 No Training (Zero-Shot)

Commercial LLMs and VLMs, such as GPT-4, have been used in a zero-shot manner to generate CAD code without task-specific training (Alrashedy et al., 2025; Mallis et al., 2025; Ocker et al., 2025). These approaches are attractive due to their rapid deployment and multi-modal capabilities. However, lacking CAD-specific supervision, they frequently produce syntactically or semantically invalid CAD outputs and therefore rely on iterative verification or visual feedback loops. Empirical results show that single-shot success rates are close to zero in many cases (Mallis et al., 2025), leading to high computational cost and slow inference.

2.1.2 Fine-Tuning

Fine-tuning approaches adapt 7B-scale LLMs or VLMs to CAD generation tasks by training on domain-specific datasets. Typical outputs are Python code (e.g., CadQuery) or JSON-based CAD representations (Wang et al., 2025; Xu et al., 2025). Vision-language models have also been fine-tuned to map images to CAD construction sequences (Doris et al., 2025). Despite improved validity compared to zero-shot prompting, these methods remain computationally expensive and are constrained by the input modalities supported by the underlying foundation models.

2.1.3 Full Training

In contrast, full training approaches develop dedicated Transformer models for CAD generation from scratch (Alam and Ahmed, 2025; Khan et al., 2024). This is feasible because CAD construction sequences are compact: typical representations fit within a 512-token context window and require fewer than 100M trainable parameters. These models are more efficient and specialized, but require carefully curated datasets and are typically limited to the modalities seen during training.

2.2 Data and Evaluation Limitations

A shared limitation across most generative CAD approaches is the scarcity of high-quality training data. Public datasets such as DeepCAD (Wu et al., 2021), Fusion360 Gallery (Willis et al., 2021), and CADParser (Zhou et al., 2023) contain only a few tens of thousands of relatively simple models, which is small compared to datasets in other AI domains. The quality of generative CAD models depends critically on training data characteristics. Current public datasets exhibit distinct properties: DeepCAD provides approximately 20,000 construction sequences with basic shape descriptions, Fusion360 Gallery contains around 8,700 parametric models with multi-view renderings, CADParser offers about 45,000 B-Rep models with operation sequences, and MFCAD++ (Colligan et al., 2022) provides machining feature annotations on B-Rep but lacks construction sequences.

While these datasets have advanced the field, they share a common limitation in their annotation approach. Text descriptions are typically generated using vision-language models applied to 2D renders or derived from basic metadata extraction (Khan et al., 2024; Xu et al., 2025). To our knowledge, no existing dataset provides structured machining feature metadata including feature types, dimensions, and poses; feature-to-operation traceability that links geometric entities to their originating CAD operations; or parallel annotations under different methodologies to enable controlled comparison of annotation strategies. This gap motivates our development of a feature-augmented annotation pipeline that addresses these limitations.

2.3 Machining Feature Recognition

CAD models contain different types of information, including the design sequence (e.g., sketch, extrude). The geometrical and topological information is stored in the boundary representation (B-Rep), which is a surface representation of the 3D model, consisting of faces, edges, and vertices (Shi et al., 2020). The geometry of a CAD model can be decomposed into different parts (features) to cluster faces, edges, and vertices into combined geometric entities. This decomposition is used in feature recognition methods to classify machining features. Machining features can be used to capture shapes for computer-aided manufacturing (Subrahmanyam and Wozny, 1995) and to identify specific features for further use, such as parallel keyways and center holes, which contain functional references (Pusch et al., 2026a). Examples of machining features are shown in Figure 1.

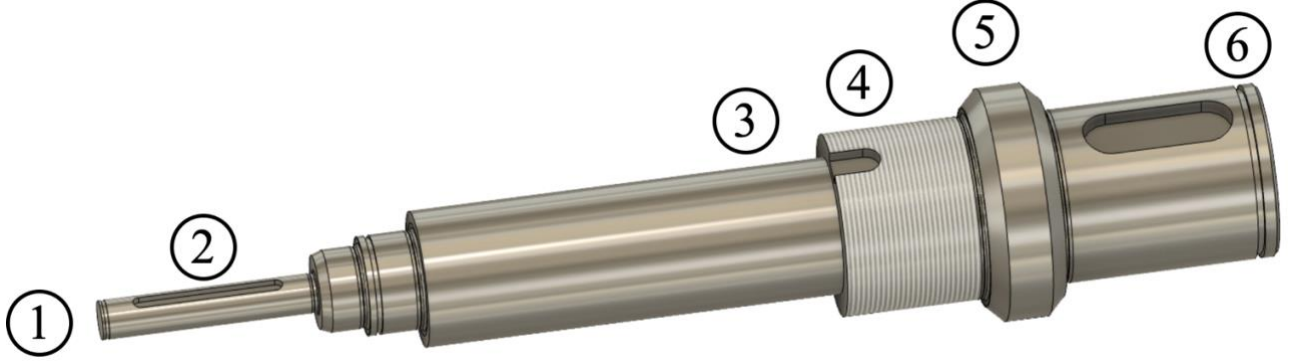


Figure 1: The six machining feature classes detectable by the algorithm: (1) center bore, (2) keyway, (3) feather keyway, (4) thread, (5) chamfer, (6) ring groove.

2.4 Hybrid AI for Engineering

Knowledge-based engineering (KBE) tools use explicit physical models and formalized design knowledge that, compared to LLMs, have already reached a mature state (Herrmann et al., 2021). Established tools such as topology optimization and finite element analysis (FEA) use physics-based objectives and constraints to reliably produce functionally optimized designs, albeit within restricted domains and without learning from past data. The fundamental gap between generative creativity and validated engineering tools motivates hybrid approaches that combine data-driven AI with knowledge-based methods.

3 Method

We use design science research (Hevner et al., 2004) as our overarching research framework, guided by Hevner's three-cycle view comprising the relevance cycle, rigor cycle, and design cycle. We have chosen this approach because the central contribution of this work is artifact related. At its core, a new research artifact is created, namely a processing pipeline including a dataset that explicitly maps functional content in CAD, thus addressing a gap resulting from current generative CAD approaches and their data- and metric-driven limitations. The relevance cycle in this paper is characterized by the derivation of requirements from the application context. For this purpose, "functional content" is structured into four levels, and the level at which combinations of primitives fulfill a clear technical purpose (such as parallel keyways for mounting machine elements on a shaft) is selected as the scope.

In the design cycle, the artifact is constructed iteratively and made verifiable. Specifically, it comprises a manufacturing-supported feature taxonomy and a multi-stage pipeline that takes CAD or B-Rep as input and consistently links geometry, machining features, and design history so that each feature remains traceable back to the generating CAD operation. The pipeline is then operationalized in two annotation workflows. A baseline workflow generates text descriptions from renderings and CAD history, while the augmented workflow additionally uses geometry-based, graph- and rule-based feature recognition, assigns feature IDs, establishes feature-to-surface and feature-to-operation traceability, and feeds this information into the text generation as structured machining feature JSON. The influence of augmentation is discussed in the results section by comparing both workflows and their outcomes.

The rigor cycle is implemented by linking to existing procedures and benchmarks. To this end, the pipeline is mirrored on established datasets and operationalized as a reproducible LLM benchmark by fine-tuning a decoder-only code LLM on text-code pairs and limiting the loss calculation to code tokens, implemented as a QLoRA setup with a 90/5/5 train/validation/test split, 4-bit NF4 quantization, and a fixed inference strategy.

4 Data and Annotation Process

To create machining feature augmented CAD data, two elements are required: (1) a feature taxonomy linking functional labels to the CAD operations that create them, and (2) CAD models containing the desired functional content. To move

beyond pure shape generation and capture engineering functionality, we outline four levels of functional content in geometry.

Following a Text-to-CAD-style pipeline similar to (Pusch et al., 2026a), we generate human-interpretable textual annotations that describe the CAD construction workflow of each shaft model. In addition, we extend this process with explicit machining-feature information obtained from our geometry-aware recognition algorithm (Pusch et al., 2026a). Concretely, we compare two annotation workflows: *baseline* workflow that derives text only from rendered views and CAD construction history as implemented by previous publications (Khan et al., 2024; Xu et al., 2025). And our *feature-augmented* workflow that additionally conditions the text generation on a structured machining-feature JSON file (taxonomy, dimensions, pose, and face-to-operation traceability). The goal is to produce textual descriptions that not only summarize overall shape and modeling steps but also explicitly verbalize manufacturing-relevant features (e.g., through holes, pockets, slots, keyways, fillets, chamfers) in a consistent, machine-readable manner. The workflows are visualized in Figure 2.

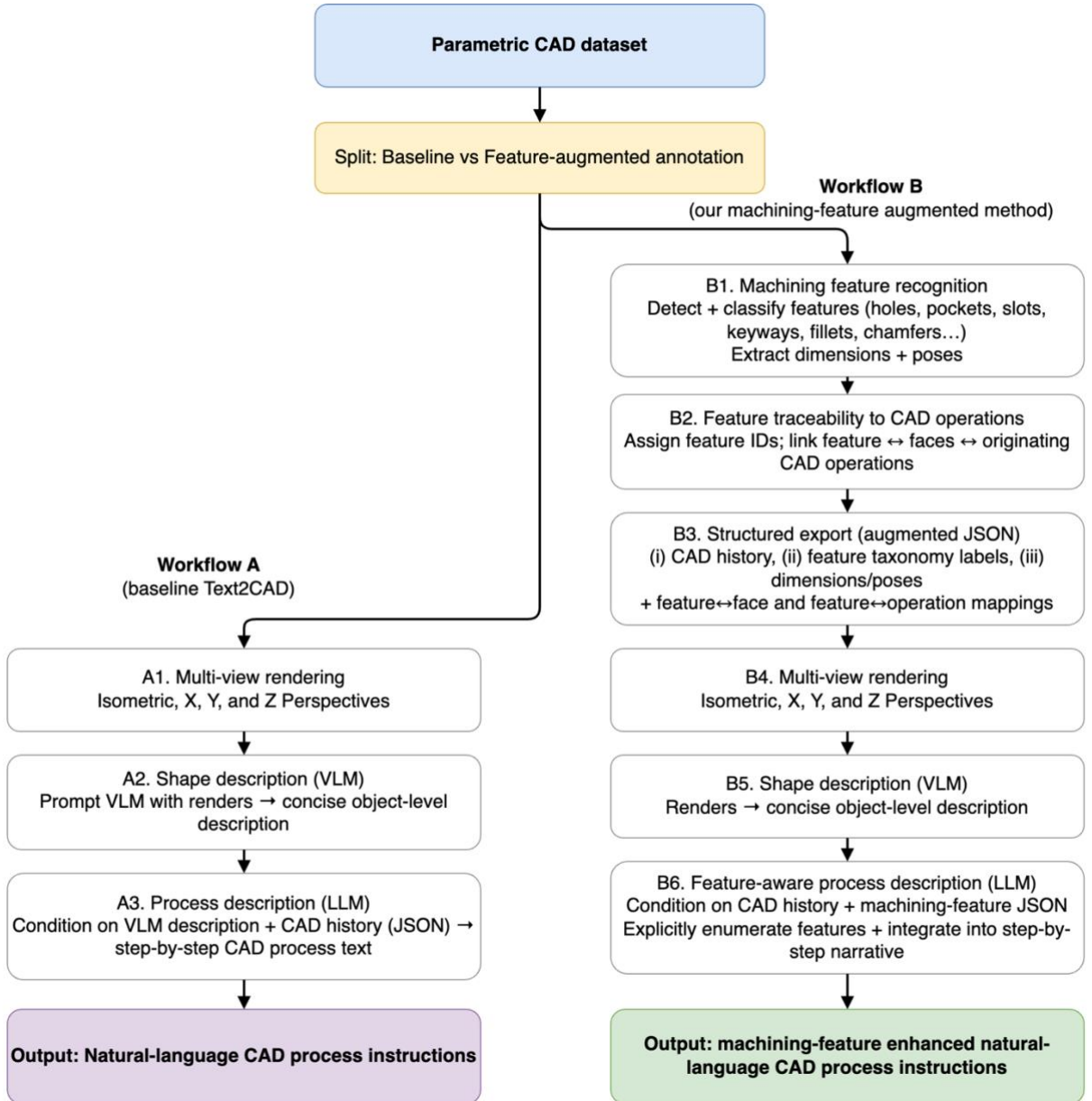


Figure 2: Overview of the two annotation workflows: (A) Baseline Text-to-CAD annotation using VLM-generated descriptions from renders and CAD construction history, and (B) feature-augmented workflow incorporating machining feature recognition, traceability, and structured metadata.

Workflow A. Baseline Text-to-CAD annotation with renders and CAD history. This workflow implements the baseline as used by previous publications (Khan et al., 2024; Xu et al., 2025).

1. Multi-view rendering. Render standardized isometric (and optionally multi-view) images for each shaft model.
2. Shape description (VLM). Prompt a vision-language model (VLM) with the rendered views to generate a concise object-level description of the final shape.
3. Process description (LLM). Provide the VLM description together with the CAD construction sequence (design history in JSON) to a large language model (LLM) to generate natural-language CAD process instructions

Workflow B. Feature-augmented annotation with renders + CAD history + machining features.

1. Machining feature recognition. Detect and classify machining features (e.g., through holes, pockets, slots, keyways, fillets, chamfers) using a geometry-aware, graph- and rule-based hybrid algorithm while extracting feature dimensions and poses.
2. Feature traceability to CAD operations. Assign unique feature IDs and link each feature to the corresponding faces and originating CAD operations, enabling face-level traceability from geometry to construction history.
3. Structured export. Export an augmented JSON representation containing (i) CAD design history (construction sequence), (ii) feature taxonomy labels, and (iii) dimensional/positional parameters, including the feature-to-face and feature-to-operation mappings.
4. Multi-view rendering. Render standardized isometric (and optionally multi-view) images for each shaft model.
5. Shape description (VLM). Generate a concise object-level description from the rendered views.
6. Feature-aware process description (LLM). Condition the LLM on the CAD construction history and the machining-feature JSON to generate a textual description that explicitly enumerates recognized features and integrates them into the step-by-step CAD process narrative

4.1 Source Data

We evaluated our pipeline on the DeepCAD and MFCAD++ datasets but found that they contain predominantly non-functional geometric objects without value for engineering design (see Figure 3). Therefore, we created a novel dataset of shaft components resembling the geometry and complexity of industrial-grade data.



Figure 3: The left 2x2 block shows randomly sampled CAD objects from DeepCAD (Wu et al., 2021), which underpins many current generative CAD models. The center 2x2 block shows randomly sampled objects from MFCAD++ dataset (Colligan et al., 2022). The right 2x2 block presents samples generated by our pipeline.

To create a large dataset that retains meaningful functional intent, CAD models must be generated automatically. We address this by developing a shaft generator that emulates key aspects of human design processes. The shaft's base geometry is created by revolving a synthesized profile, while machining features are added through CAD operations. Through this approach, the system enables automated and consistent synthesis of complex, function-driven shaft geometries. We make the source code available for reproducibility.

5 Proof-of-Concept by fine-tuning an LLM

We demonstrate that our process is superior training a generative model to create machine elements with rich machining features. To this end, we use an LLM to convert the feature descriptions into natural language prompts.

5.1 Architecture

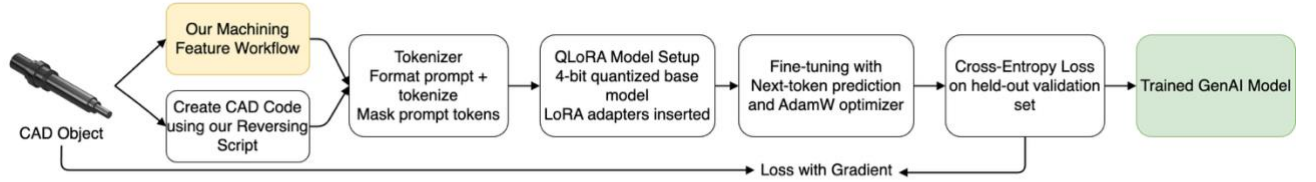
We employ the decoder-only causal language model *Qwen2.5-Coder-1.5B-Instruct* (Qwen Team, 2024) as our foundational architecture to synthesize parametric CAD Python programs from natural-language descriptions. To adapt the model for this downstream task while maintaining computational efficiency, we utilize the QLoRA (Dettmers et al.,

2023) fine-tuning method. The base model is loaded with 4-bit NF4 quantization, and Low-Rank Adapters (LoRA) (Hu et al., 2021) are injected into the attention and MLP projection modules (q_proj , k_proj , v_proj , o_proj , $gate_proj$, up_proj , $down_proj$). The learnable adapters are configured with a rank $r=16$, scaling factor $\alpha=32$, and a dropout rate of 0.2.

The training objective is autoregressive next-token prediction. Each input sequence concatenates the natural-language instruction and the target code, formatted as: `# Description: ... # Python Code:.` To ensure the model focuses solely on code generation, we apply a masking strategy to the supervision signal, setting the labels for tokens corresponding to the natural-language prompt and padding to -100, thereby computing the loss exclusively over the valid code tokens. An overview of the training and inference pipelines is illustrated in Figure 4.

5.2 Experimental Setup

Training Pipeline



Inference Pipeline

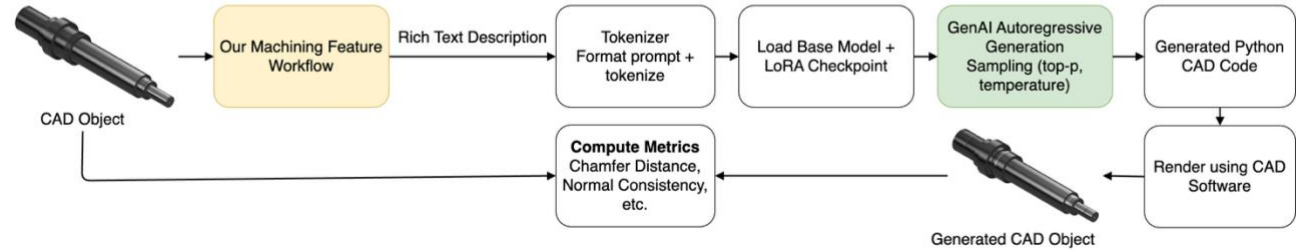


Figure 4: Visualization of the training and inference processes we used to train and evaluate our fine-tuned LLM. Using this setup we aim to prove the hypothesis that our machining-feature augmented data is superior for CAD generation compared to previous approaches.

5.2.1 Dataset and Preprocessing

We utilize the shaft generator and to generate 1770 annotated shaft components. To facilitate code generation, the CAD geometry is converted into executable Python scripts using the *RapidCADPy* library (Berger, 2025). This preprocessing aligns with contemporary frameworks that treat CAD synthesis as a code generation task. The processed dataset is partitioned into training, validation, and testing subsets using a 90/5/5 split. All sequences are padded or truncated to a fixed context length of 3321 tokens, which covers over 95% of inputs.

5.2.2 Implementation Details

Training is performed using PyTorch Lightning with bf16-mixed precision to optimize memory usage. We use the AdamW optimizer (Kingma and Ba, 2017) with a learning rate of 2×10^{-4} . Gradient checkpointing is enabled to further reduce the memory footprint of the 4-bit quantized base model. Experiments were conducted on a compute cluster consisting of four NVIDIA L40S GPUs, with a total training duration of approximately 6 hours. Checkpoint selection is based on the lowest validation loss, and experiment metadata is tracked via Weights & Biases. To prevent overfitting, we set the dropout probability to 20% and added early stopping when validation loss rises for three consecutive epochs.

5.2.3 Inference and Evaluation

For inference, we generate Python programs using nucleus sampling with a temperature of 0.7 and top- p of 0.9, generating up to half the maximum context length in new tokens. To rigorously assess the quality of the generated outputs, we report a suite of fidelity metrics: Chamfer Distance (CD) (Fan et al., 2016), Intersection-over-Union (IoU) (Cignoni et al., 1998), F-score (Tatarchenko et al., 2019), Normal Consistency (NC) (Henderson and Ferrari, 2020), invalid ratio (IR), and CodeBLEU (Ren et al., 2020).

6 Results and Discussion

6.1 Rich CAD-to-Text Samples

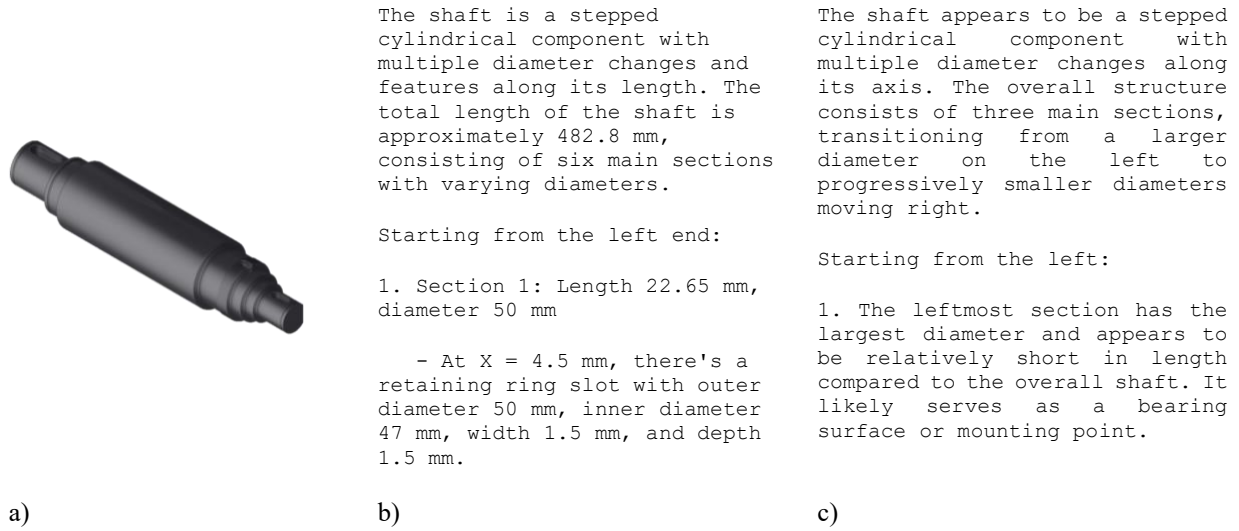


Figure 5: Qualitative example of our machining feature augmented CAD descriptions. Figure a) shows the input CAD object. Figure b) displays the output of our method, c) displays the output following the method outlined by (Khan et al., 2024; Xu et al., 2025).

From this sample it is already evident that our approach (b) contains superior details while (c) is void of dimensional details.

Using the proposed data processing pipeline, we generated an intermediate dataset comprising a comprehensive set of paired parametric CAD objects and rich natural-language descriptions. In contrast to prior datasets, which typically provide only coarse or sparsely annotated textual metadata, our dataset contains substantially more detailed descriptions that capture both geometric and functional characteristics. Most importantly, while dimensional information is frequently lost or omitted in related work, our representation explicitly preserves key dimensional parameters, enabling downstream models to learn not only shape structure but also scale- and tolerance-relevant design intent. Figure 5 shows a qualitative comparison of our data to previous approaches.

6.2 Text-to-CAD Reconstruction

Table 2: Comparison of 3D reconstruction accuracy metrics between feature-augmented (Workflow B) and baseline (Workflow A) annotation workflows with statistical significance analysis. Lower is better for Chamfer Distance; higher is better for all other metrics. * denotes statistical significance at $p < 0.05$. Values shown as mean $\pm$ std.

Metric	Workflow B (ours)	Workflow A	Significant	Winner
CD ($\times 1000$) $\downarrow$	0.27 ± 0.07	1.37 ± 1.42	*	B
F-score $\uparrow$	0.99 ± 0.01	0.95 ± 0.13	*	B
Normal Consistency $\uparrow$	0.87 ± 0.03	0.80 ± 0.04	*	B
CodeBLEU $\uparrow$	0.85 ± 0.40	0.51 ± 0.16		B
IoU $\uparrow$	0.99 ± 0.01	0.47 ± 0.18	*	B
IR $\downarrow$	0.17	0.00		A

The feature-augmented approach achieves superior performance on four of five metrics, with particularly striking improvements in Chamfer Distance (80% reduction in reconstruction error, from 1.37 ± 1.42 to $0.27 \pm 0.07 \times 10^{-3}$) and IoU (111% improvement, from 0.47 ± 0.18 to 0.99 ± 0.01). The near-perfect F-score (0.99 ± 0.01) and IoU (0.99 ± 0.01) for the feature-augmented workflow indicate that the generated CAD code produces geometries that are virtually identical to the ground truth models. This suggests that explicit machining-feature annotations enable the model to capture not only overall shape but also fine-grained geometric details essential for manufacturing.

At the 0.05 significance level, the feature-augmented approach achieves statistically significant improvements in Chamfer Distance (0.27 ± 0.07 vs. $1.37 \pm 1.42 \times 10^{-3}$), F-score (0.99 ± 0.01 vs. 0.95 ± 0.13), Normal Consistency (0.87 ± 0.03 vs. 0.80 ± 0.04), and IoU (0.99 ± 0.01 vs. 0.47 ± 0.18). CodeBLEU and Invalid Rate differences, while favoring the feature-augmented and baseline workflows respectively, do not reach statistical significance due to the high standard deviation.

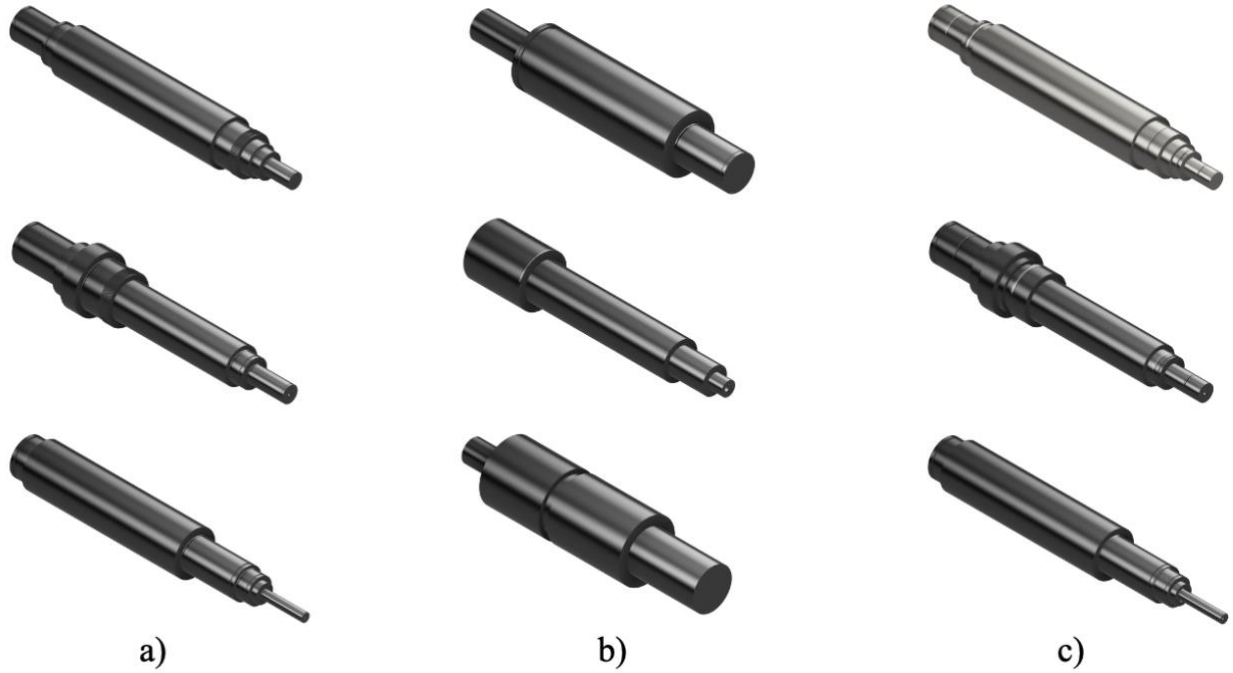


Figure 6: Column a) shows a render of the input shaft that we used our pipeline. Column b) shows the rendered result using the baseline approach (Workflow A in Figure 2“). Column c) shows the rendered result of our machining-feature enhanced approach (Workflow B in Figure 2). From the qualitative examples it is evident that with our improved method, the downstream LLM is capable of generating CAD objects with greater detail, aligning closer to the input. The center column (b) does not contain the detailed machining features, whereas using our workflow allows the LLM to reconstruct such critical details.

The feature-augmented workflow also achieves higher CodeBLEU scores (0.85 ± 0.40 vs. 0.51 ± 0.16), though this difference is not statistically significant. This suggests that feature-aware annotations improve not only geometric accuracy but also code quality and structural similarity to reference implementations. The one metric where the baseline workflow performs better is Invalid Rate (IR), with 0% invalid outputs compared to 17% for the feature-augmented approach.

6.3 Discussion

The results support our central hypothesis that *text descriptions incorporating explicit machining-feature metadata improve CAD code generation*. The feature-augmented workflow outperforms the baseline on five of six metrics, with the largest gains in geometric fidelity (Chamfer Distance, F-score, Normal Consistency, IoU). In particular, the 80% reduction in Chamfer Distance and near-perfect F-score and IoU values indicate that feature-aware supervision helps the model recover not only overall shape but also fine-grained, manufacturing-relevant details.

Notably, Workflow A achieves a lower invalid ratio compared to Workflow B. This may be because Workflow A produces simpler CAD geometries that leave less room for potential errors. Workflow B, while exhibiting a higher invalid ratio, produces more complex machining features that, if malformed, lead to invalid CAD geometries. Higher complexity is nonetheless more desirable than overly simple geometries, even at the cost of a higher invalidity rate.

From an industrial perspective, the machine learning method presented in this paper could enable engineers to rapidly converge on an initial prototype. This is valuable when developing a part from scratch proves more efficient than locating and adapting an existing design. Unlike template-based approaches, the proposed method offers greater flexibility and encourages more creative design exploration. The strong reconstruction performance further suggests that taxonomy-grounded feature annotations allow the model to reliably preserve critical functional elements such as bearing seats, chamfers, fillets, and keyways, laying a solid foundation for AI-assisted CAD workflows that remain firmly aligned with manufacturing constraints. In this paper we demonstrated our method using shaft components, however our method is applicable to other machining elements as well.

7 Conclusion and Outlook

This work demonstrates that augmenting text descriptions with explicit machining feature metadata substantially improves Text-to-CAD generation quality. Our feature-augmented annotation pipeline produces more geometrically accurate CAD models than standard VLM-based approaches. The publicly released data pipeline and code enable reproducible research in feature-aware generative CAD.

This work has acknowledged limitations that define the scope for future research. The dataset and experiments can be extended to other rotational or non-rotational component types would enhance generalizability and industrial applicability. Similarly, the feature taxonomy encompasses six machining feature classes, which is deliberately constrained to the functional level targeted in this work.

Future work should address scaling to complex assemblies, incorporating engineering validation beyond geometric similarity, evaluating functional reasoning capabilities, and generating negative examples to improve design validity.

References

- Alam, M.F., Ahmed, F., 2025. GenCAD: Image-Conditioned Computer-Aided Design Generation with Transformer-Based Contrastive Representation and Diffusion Priors. <https://doi.org/10.48550/arXiv.2409.16294>
- Alrashedy, K., Tambwekar, P., Zaidi, Z., Langwasser, M., Xu, W., Gombolay, M., 2025. Generating CAD Code with Vision-Language Models for 3D Designs. <https://doi.org/10.48550/arXiv.2410.05340>
- Berger, E., 2025. <https://github.com/rapidcad/rapidcadpy>. <https://doi.org/10.5281/zenodo.20408820>
- Berger, E., Herrmann, K., Pusch, F., Mehlstäubl, J., Lachmayer, R., Paetzold-Byhain, K., 2026. From geometry to function: Towards context-aware generative AI for engineering design, in: Proceedings of the DESIGN Conference. Dubrovnik, Croatia. <https://doi.org/10.1017/pds.2026.10577>
- Bode, B., Herrmann, K., Reusch, J., Plappert, S., Ehlers, T., Gembarski, P.C., Hasse, C., Lachmayer, R., 2023. Thermo-elastic topology optimization for high temperatures gradients using load separation. *Procedia CIRP* 119, 576–581. <https://doi.org/https://doi.org/10.1016/j.procir.2023.03.113>
- Cignoni, P., Rocchini, C., Scopigno, R., 1998. Metro: Measuring error on simplified surfaces. *Comput. Graph. Forum* 17, 167–174. <https://doi.org/10.1111/1467-8659.00236>
- Colligan, A., Robinson, T.T., Nolan, D., Hua, Y., 2022. MFCAD++ dataset. <https://doi.org/10.17034/d1fec5a0-8c10-4630-b02e-b92dc81df823>
- Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L., 2023. QLoRA: Efficient Finetuning of Quantized LLMs. <https://doi.org/10.48550/arXiv.2305.14314>
- Doris, A.C., Alam, M.F., Nobari, A.H., Ahmed, F., 2025. CAD-Coder: An Open-Source Vision-Language Model for Computer-Aided Design Code Generation. <https://doi.org/10.48550/arXiv.2505.14646>
- Fan, H., Su, H., Guibas, L.J., 2016. A point set generation network for 3D object reconstruction from a single image. *CoRR abs/1612.00603*.
- He, C., Zhang, S., Zhang, L., Miao, J., 2025. CAD-Coder:Text-Guided CAD Files Code Generation. <https://doi.org/10.48550/arXiv.2505.08686>
- Henderson, P., Ferrari, V., 2020. Learning to predict 3D objects with an RGB-D camera. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 149–162. <https://doi.org/10.1109/TPAMI.2018.2868169>
- Herrmann, K., Altun, O., Wolniak, P., Mozgova, I., Lachmayer, R., 2021. Methodischer aufbau von entwicklungsumgebungen nach dem generative parametric design approach, in: Proceedings of the 32nd Symposium Design for X, DFX 2021. <https://doi.org/10.35199/dfx2021.14>
- Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design Science in Information Systems Research. *MIS Q.* 28, 75–105. <https://doi.org/10.2307/25148625>
- Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., 2021. LoRA: Low-Rank Adaptation of Large Language Models. <https://doi.org/10.48550/arXiv.2106.09685>
- Khan, M.S., Sinha, S., Sheikh, T.U., Stricker, D., Ali, S.A., Afzal, M.Z., 2024. Text2CAD: Generating Sequential CAD Models from Beginner-to-Expert Level Text Prompts. <https://doi.org/10.48550/arXiv.2409.17106>
- Kingma, D.P., Ba, J., 2017. Adam: A Method for Stochastic Optimization. <https://doi.org/10.48550/arXiv.1412.6980>
- Mallis, D., Karadeniz, A.S., Cavada, S., Rukhovich, D., Foteinopoulou, N., Cherenkova, K., Kacem, A., Aouada, D., 2025. CAD-Assistant: Tool-Augmented VLLMs as Generic CAD Task Solvers. <https://doi.org/10.48550/arXiv.2412.13810>
- Ocker, F., Menzel, S., Sadik, A., Rios, T., 2025. From Idea to CAD: A Language Model-Driven Multi-Agent System for Collaborative Design. <https://doi.org/10.48550/arXiv.2503.04417>
- Pusch, F., Berger, E., Gembarski, P.C., Mehlstäubl, J., Herrmann, K., Paetzold-Byhain, K., Lachmayer, R., 2026a. Labeling CAD parts to capture design intent for llms through machining feature recognition. *Procedia CIRP* 137.
- Pusch, F., Herrmann, K., Becker, C., Gembarski, P.C., Lachmayer, R., 2026b. Extending graph-based feature recognition to functional features for enhanced process planning. *Procedia CIRP* 137.
- Qwen Team, 2024. Qwen2.5-coder-1.5B-instruct.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., 2019. Language models are unsupervised multitask learners.
- Ren, S., Guo, D., Lu, S., Zhou, L., Liu, S., Tang, D., Sundaresan, N., Zhou, M., Blanco, A., Ma, S., 2020. CodeBLEU: a method for automatic evaluation of code synthesis. *CoRR abs/2009.10297*.
- Shi, Y., Zhang, Y., Xia, K., Harik, R., 2020. A critical review of feature recognition techniques. *Comput.-Aided Des. Appl.* 17, 861–899. <https://doi.org/10.14733/cadaps.2020.861-899>
- Subrahmanyam, S., Wozny, M., 1995. An overview of automatic feature recognition techniques for computer-aided process planning. *Comput. Ind.* 26, 1–21. [https://doi.org/10.1016/0166-3615\(95\)80003-4](https://doi.org/10.1016/0166-3615(95)80003-4)
- Tatarchenko, M., Dosovitskiy, A., Brox, T., 2019. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 967–979. <https://doi.org/10.1109/TPAMI.2018.2821896>
- Wang, S., Chen, C., Le, X., Xu, Q., Xu, L., Zhang, Y., Yang, J., 2025. CAD-GPT: Synthesising CAD Construction Sequence with Spatial Reasoning-Enhanced Multimodal LLMs. *Proc. AAAI Conf. Artif. Intell.* 39, 7880–7888. <https://doi.org/10.1609/aaai.v39i8.32849>

- Willis, K.D.D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J.G., Solar-Lezama, A., Matusik, W., 2021. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences. <https://doi.org/10.48550/arXiv.2010.02392>
- Wu, R., Xiao, C., Zheng, C., 2021. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. <https://doi.org/10.48550/arXiv.2105.09492>
- Xu, J., Zhao, Z., Wang, C., Liu, W., Ma, Y., Gao, S., 2025. CAD-MLLM: Unifying Multimodality-Conditioned CAD Generation With MLLM. <https://doi.org/10.48550/arXiv.2411.04954>
- Zhou, S., Tang, T., 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. International Joint Conferences on Artificial Intelligence Organization, Macau, SAR China, pp. 1804–1812. <https://doi.org/10.24963/ijcai.2023/200>