Automated CAD Modeling Sequence Generation from Text Descriptions via Transformer-Based Large Language Models

Jianxing Liao[1]*, Junyan Xu[1]*, Yatao Sun[1] , Maowen Tang[1] , Sicheng He[1] , Jingxian Liao[1], Shui Yu[1]†, Yun Li[1]†, Xiaohong Guan[2]
[1]Shenzhen Institute for Advanced Study,
University of Electronic Science and Technology of China,
Shenzhen, 518000, China
[2]i4AI Ltd, London WCIN3AX, United Kingdom
†Shui.Yu@i4AI.org, Yun.Li@ieee.org

Abstract

Designing complex computer-aided design (CAD) models is often time-consuming due to challenges such as computational inefficiency and the difficulty of generating precise models. We propose a novel language-guided framework for industrial design automation to address these issues, integrating large language models (LLMs) with computer-automated design (CAutoD).Through this framework, CAD models are automatically generated from parameters and appearance descriptions, supporting the automation of design tasks during the detailed CAD design phase. Our approach introduces three key innovations: (1) a semi-automated data annotation pipeline that leverages LLMs and vision-language large models (VLLMs) to generate high-quality parameters and appearance descriptions; (2) a Transformer-based CAD generator (TCADGen) that predicts modeling sequences via dual-channel feature aggregation; (3) an enhanced CAD modeling generation model, called CADLLM, that is designed to refine the generated sequences by incorporating the confidence scores from TCADGen. Experimental results demonstrate that the proposed approach outperforms traditional methods in both accuracy and efficiency, providing a powerful tool for automating industrial workflows and generating complex CAD models from textual prompts. The code is available at `https://jianxliao.github.io/cadllm-page/`

1 Introduction

Computer-aided design (CAD) is important in industrial design and additive manufacturing. It is a key tool from the early stages of design to prototype production (Marchesi et al., 2021; Rapp et al., 2021). Computer-Automated Design (CAutoD) represents an evolution of the traditional CAD by integrating automation and machine learning algorithms to assist in the design process (Li, 2022). With the fast growth of AI, especially large language models (LLMs), combining CAD and AI is becoming a major factor in improving design speed and creativity (Guo et al., 2022; Dai and Hong, 2024). This development speeds up the design process and increases the possible uses of CAD systems, making them more intelligent and personalized. In recent years, more research focuses on combining AI with CAD (Xu et al., 2024a; Wu et al., 2024). Methods such as generating parametric CAD models from the textual description and improving automated design tools with AI are proposed (Yavartanoo et al., 2024). These studies show that AI in CAD can speed up design and make CAD tools more flexible and smart. The integration of AI is pushing CAD towards CAutoD, representing a shift from the traditional, computer-aided design to computer-automated design processes.

Most current research on combining LLMs with CAD focuses on the data synthesis ability of these models, such as generating design information by processing large datasets. However, design data synthesized by LLMs often lacks sufficient accuracy (Fan et al., 2025; Wu et al., 2021). Without parameter review, converting this data into high-quality, editable CAD models is difficult, especially in industrial design, where precision is critical. While current research emphasizes the ability of LLMs to understand complex problems, their application in complex CAD tasks still faces several challenges (Plaat et al., 2024). First, LLMs are computationally expensive and inefficient (Luo et al., 2023). Second, it is difficult for designers to guide these LLMs to generate reasonable CAD models using simple language or visual descriptions, limiting their practical use in CAD design. Despite these challenges, LLMs still have significant potential in CAD design and warrant further research. One promising solution is to combine

* These authors are equal first authors
† Corresponding authors

finely tuned small generative models (with fewer parameters) with large models (with massive parameters) (Xu et al., 2023). The small generative models can provide more accurate guidance for specific tasks, compensating for the limitations of the large models in detailed reasoning and thus improving overall accuracy (Chen and Varoquaux, 2024). This approach can enhance the design synthesis capabilities of large models and reduce the uncertainties that often arise in the reasoning process (Fang et al., 2024). Using small models as auxiliary modules in CAD design can improve the accuracy and efficiency of design, offering a more reliable solution for complex tasks.

In this paper, we propose a novel framework, as shown in Fig. 1, for generating high-quality CAD modeling sequences, where LLMs generate CAD operation commands and the corresponding parameters to facilitate CAD modeling. Specifically, we focus on the detailed design phase, where precise parameter and appearance descriptions are transformed into CAD models without human intervention. Through this automation, our work represents a significant step toward fully automating the CAD design process, enhancing both speed and accuracy in industrial applications, and supporting the vision of CAutoD. Our approach has three key innovations:

- We propose a semi-automated annotation pipeline with LLMs and VLLMs, enabling high-quality CAD description generation through automated validation and verification.
- We present TCADGen, a Transformer-based dual-channel architecture for transforming descriptive annotations into CAD commands, achieving improved sequence accuracy.
- We introduce a CAD-specific LLM enhancement framework, demonstrating its effectiveness in refining command types and parameter values for industrial applications.

2 Related Works

Extending the traditional CAD, CAutoD has witnessed significant advancements in recent years across various industries, including ship design, architectural design, and mechanical engineering (Ang et al., 2016; Bye et al., 2017). These advancements are largely driven by progress in dataset (Wu et al., 2021; Yavartanoo et al., 2024; Fan et al., 2025), sequence generation (Yavartanoo et al., 2024), and large language models (Rapp et al., 2021). ModelNet is the first large dataset of 3D CAD mesh models (Vishwanath et al., 2009). However, it does not include modeling process data. The Fusion 360 Gallery dataset later adds modeling history and assembly data (Willis et al., 2021). The Mechanical Components Benchmark then provides complete modeling histories for mechanical parts (Kim et al., 2020). These datasets focus on geometric features but lack natural language descriptions of CAD models. Deep learning creates new ways to approach CAD modeling. DeepCAD uses a Transformer model to generate CAD sequences (Wu et al., 2021). It breaks down CAD modeling into sketches and extrusion steps. However, it generates sketches before extrusion parameters, often creating disconnected sequences. It also supports only basic modeling operations. Text2CAD proposes a different approach that generates CAD sequences from text description and visual features (Yavartanoo et al., 2024). However, they do not consider that large models excel in generative and reasoning abilities to assist in design.

LLMs lead to new CAD modeling methods. BlenderLLM uses LLMs to generate CAD sequences through self-improvement (Du et al., 2024). It creates the BlendNet dataset and CAD-Bench for evaluation. However, its sequences lack geometric constraints, which affect model accuracy. Query2CAD uses LLMs to generate CAD commands and improves designs through iterations (Badagabettu et al., 2024). However, it often misses steps when handling complex tasks. CAD-CodeVerify combines vision and language models to generate and improve CAD code (Alrashedy et al., 2024). However, the multiple rounds of visual checking and code fixing needed make it slow to use. The AutoForma framework (Liao et al., 2024) , which leverages a large language model-based multi-agent system for the automated generation of CAD models. However, the method is costly and suffers from low generation efficiency.

3 Methodology

3.1 Problem Definition

Given a CAD model's appearance description text $T_{\text{appear}} \in \mathcal{V}_{\text{appear}}$ and its corresponding parameter modeling description text $T_{\text{param}} \in \mathcal{V}_{\text{param}}$, where $\mathcal{V}_{\text{appear}}$ and $\mathcal{V}_{\text{param}}$ represent the vocabulary for appearance descriptions and parameter descrip-

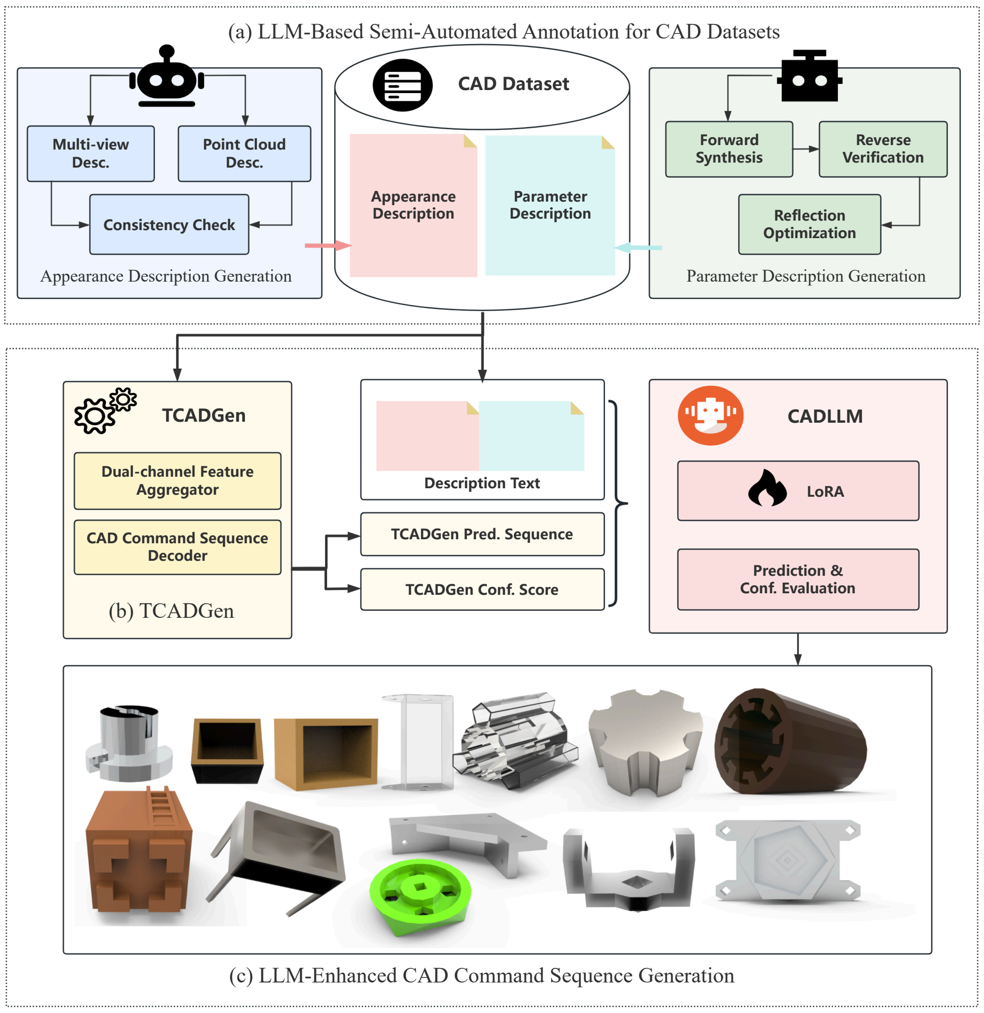

(a) LLM-Based Semi-Automated Annotation for CAD Datasets
Multi-view Desc.
Point Cloud Desc.
Consistency Check
Appearance Description Generation
CAD Dataset
Appearance Description
Parameter Description
Forward Synthesis
Reverse Verification
Reflection Optimization
Parameter Description Generation
TCADGen
Dual-channel Feature Aggregator
CAD Command Sequence Decoder
(b) TCADGen
Description Text
TCADGen Pred. Sequence
TCADGen Conf. Score
CADLLM
LoRA
Prediction & Conf. Evaluation
(c) LLM-Enhanced CAD Command Sequence Generation

Figure 1: The overall framework of automated CAD modeling from text descriptions, leveraging Transformer-based sequence generation and LLM-driven refinement. The innovations proposed in our work are highlighted in (a), (b), and (c).

tions respectively. Our goal is to generate a complete CAD Command Sequence(CCS)[1]. $\mathbf{M} = \{(c_1, \mathbf{p}_1), (c_2, \mathbf{p}_2), ..., (c_N, \mathbf{p}_N)\}$, where $c_i \in \mathcal{C}$ represents the type of the i-th modeling command, $\mathbf{p}_i \in \mathbb{R}^d$ represents the corresponding parameter vector in d-dimensional vector space, and $\mathcal{C}$ is the predefined set of CAD commands. This task can be divided into two stages:

TCADGen Sequence Generation: We define the TCADGen sequence generation problem as a mapping from appearance and parameter descriptions to CAD command sequences and their confidence scores: $f_{\text{TCADGen}}(T_{\text{appear}}, T_{\text{param}}) = (\mathbf{M}, \mathbf{S})$, where T_{appear} is the appearance description text, T_{param} is the parameter description text, $\mathbf{M}$ is the predicted CAD Command Sequence (CCS), $\mathbf{S} = s_1, s_2, ..., s_N$ represents the confidence scores for each command and its parameters, $s_i = (s_i^{\text{cmd}}, s_i^{\text{args}})$ represents the confidence scores for command type and parameters respectively, N is the total number of commands in the sequence, s_i^{cmd} represents the confidence score for the i-th command type, and s_i^{args} represents the confidence score for the parameters of the i-th command.

CADLLM CCS enhancement: We formulate the CADLLM CCS enhancement stage as: $f_{\text{CADLLM}}(T_{\text{appear}}, T_{\text{param}}, \mathbf{M}, \mathbf{S}) = \mathbf{M}^*$, where $\mathbf{M}^*$ is the final optimized modeling sequence, f_{CADLLM} represents the industrial design domain large model CADLLM. The optimization objective is formulated as follows:

$$\begin{aligned} \max_{\mathbf{M}^*} \quad & p(\mathbf{M}^* | T_{\text{appear}}, T_{\text{param}}) \\ \text{s.t.} \quad & \forall (c_i, \mathbf{p}_i) \in \mathbf{M}^*, c_i \in \mathcal{C}, \mathbf{p}_i \in \mathbb{R}^d, \end{aligned} \tag{1}$$

[1]An introduction to CCS can be found in Appendix A.1

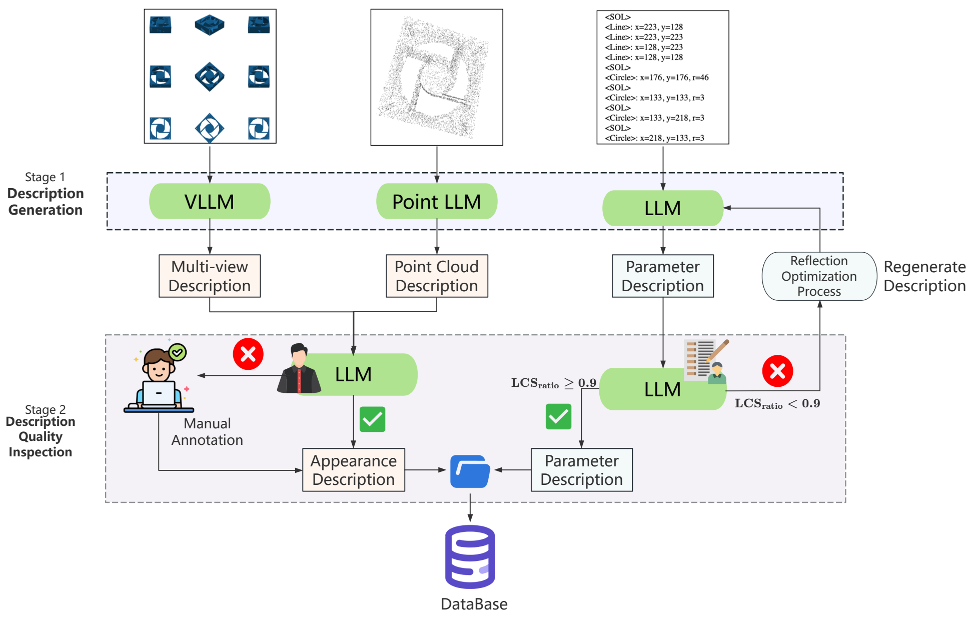

```
<SOL>
<Line>: x=223, y=128
<Line>: x=223, y=223
<Line>: x=128, y=223
<Line>: x=128, y=128
<SOL>
<Circle>: x=176, y=176, r=46
<SOL>
<Circle>: x=133, y=133, r=3
<SOL>
<Circle>: x=133, y=218, r=3
<SOL>
<Circle>: x=218, y=133, r=3
```

Figure 2: LLM-Based Semi-Automated Annotation for CAD Datasets Process. This pipeline primarily generates appearance and parameter descriptions for CAD datasets, focusing on the Description Generation and Description Quality Inspection stages.

where $\mathbf{M}^*$ is constrained to be a valid CAD modeling sequence, c_i represents the i-th command type, $\mathbf{p}_i$ represents the parameters for the i-th command, $\mathcal{C}$ is the set of all valid CAD commands, d is the dimension of the parameter space, and $p(\mathbf{M}|T_{\text{appear}}, T_{\text{param}})$ represents the probability of generating sequence $\mathbf{M}^*$ given the input descriptions.

The proposed framework aims to generate high-quality CAD modeling sequences through this two-stage process, where TCADGen generates the initial modeling sequences and the corresponding confidence assessment, followed by the optimization and improvement of the sequences in CADLLM.

3.2 Proposed Method

3.2.1 LLM-Based Semi-Automated Annotation for CAD Datasets

To efficiently create high-quality annotations for large CAD datasets, we propose a semi-automated annotation pipeline based on LLMs, utilizing multiple LLMs in a coordinated workflow. Annotating the appearance of CAD models requires human-machine collaboration, while parameter description annotation can be fully automated. Each annotation pipeline includes a description generation stage and a description quality control stage. The process is described in Fig. 2.

(i). Semi-Automated Appearance Description Generation

To address the challenge of efficiently labeling large-scale CAD datasets, we leverage LLMs to automate parameter descriptions and assist in appearance annotation. For the description generation stage, we first sample multiple views of the CAD model and use a VLLM to generate an appearance description based on these views. Simultaneously, point cloud data from the CAD model is processed by a PointLLM (Xu et al., 2024b) to generate descriptions based on the point cloud. The models answer key questions regarding the 3D model's appearance features, material composition, specific details, and function.

For the description quality inspection stage, we use LLM to check the consistency between the point cloud and multi-view descriptions and point out conflicts (e.g., one calling it a cylinder, another a cube). If the point cloud and multi-view descriptions align well, they are merged into a comprehensive appearance description; otherwise, manual annotation is required (our experiments showed an automatic pass rate of 98.4%, with only a few samples needing human intervention). The final appearance description comes from combining both the automated and manual annotations.

(ii). Automated Parameter Description Generation

The parameter description is primarily generated based on the CCS. For the description generation stage, annotated CCS parameter data is input into the LLM along with a predefined prompt template. The prompt also specifies that descriptions must be "fluent and instructive paragraphs" detailing each step of the CAD modeling process and the corresponding parameters.

For the description quality inspection stage, we validate the accuracy of the generated CCS parameter description by reverse verification. It is deemed correct if the model can reconstruct the ground truth CCS from the generated description. The reliability of the description is measured by the longest common subsequence (LCS) ratio between the generated CCS and the ground truth CCS. The LCS metric effectively captures structural similarities between sequences while allowing for minor variations, making it well-suited for evaluating CAD modeling descriptions. To quantify the similarity between the generated description and the ground truth, we use the $\text{LCS}_{\text{ratio}}$:

$$\text{LCS}_{\text{ratio}} = \frac{\text{len}(\text{LCS}(g, r))}{\text{len}(g)}, \tag{2}$$

where g denotes the ground truth CCS and r represents the generated CCS. Descriptions with an $\text{LCS}_{\text{ratio}}$ above 0.9 are accepted, while those with an LCS ratio below 0.9 are considered low reliability and enter a reflection optimization process.

Reflection Optimization Process: For low-reliability descriptions, we follow a reflection optimization process:

1. Analyze potential issues in the description generation stage based on ground truth CCS.
2. Generate a new CCS parameter description based on the model's reflection feedback from Step 1.
3. Recheck the newly generated description.

This process repeats until the description reaches the reliability threshold (i.e., $\text{LCS}_{\text{ratio}} \geq 0.9$) or the maximum retry limit (twice). Note that this process is fully automated, requiring no human intervention for parameter description annotation.

3.2.2 TCADGen

We propose a novel CAD model generation framework called TCADGen, which innovatively adopts a dual-channel Transformer architecture to achieve an effective fusion of parameter modeling knowledge and visual appearance features, as shown in Fig. 3.

(i). Dual-Channel Feature Aggregator To fully preserve both parametric features and appearance features of the model, we propose a dual-channel feature aggregator that simultaneously embeds parameter descriptions and appearance descriptions. Parameter descriptions contain both basic geometric operations and specific parameter designs, while appearance descriptions include model appearance analysis. Based on DeBerta-Large-v3 (Tran et al., 2024), we first encode and project the input text through BERT and linear transformation:

$$\mathbf{h}_p = BERT(T_{param})\mathbf{W}_p + \mathbf{b}_p, \tag{3}$$

$$\mathbf{h}a = BERT(T_{appear})\mathbf{W}_a + \mathbf{b}_a, \tag{4}$$

where T_{param} and T_{appear} are the input parameter and appearance description texts respectively, d_p and d_a are the dimensions of parameter and appearance features, d is the dimension of the shared semantic space, $\mathbf{W}_p \in \mathbb{R}^{d_p \times d}$ and $\mathbf{W}_a \in \mathbb{R}^{d_a \times d}$ are projection matrices that map BERT embeddings into a shared d-dimensional semantic space, and $\mathbf{b}_p$, $\mathbf{b}_a$ represent bias vectors.

Inspired by capsule networks (Patrick et al., 2022), we design an adaptive feature fusion mechanism based on dynamic routing to fully utilize the normalized feature representations:

$$\mathbf{s}_j = \sum_i \text{softmax}(\mathbf{W}_r[\hat{\mathbf{h}}_p^i; \hat{\mathbf{h}}_a^j])\hat{\mathbf{h}}p^i\mathbf{W}ij, \tag{5}$$

where $\hat{\mathbf{h}}_p$ and $\hat{\mathbf{h}}_a$ denote the normalized representations of $\mathbf{h}_p$ and $\mathbf{h}_a$ respectively, $\mathbf{W}_r$ is a learnable routing weight matrix, i and j correspond to the subscripts of the feature tokens, $[\cdot;\cdot]$ represents feature concatenation, and $\mathbf{W}_{ij}$ is the feature transformation matrix between tokens i and j.

The final fused features are obtained through a weighted combination:

$$\mathbf{z} = \sum_j \alpha_j \mathbf{s}_j, \tag{6}$$

where α_j represents adaptive weight coefficients calculated through the attention mechanism and $\mathbf{z}$ is the final fused feature vector.

(ii). CAD Command Sequence Decoder

We then decode the fused feature vector $\mathbf{z}$ into a standardized CAD modeling sequence. We first use a multi-head attention mechanism to capture global dependencies:

$$\mathbf{H}^l = \text{MultiHead}(\mathbf{W}_Q^l \cdot \text{PE}(\cdot), \mathbf{W}_K^l \cdot \mathbf{z}, \mathbf{W}_V^l \cdot \mathbf{z}), \tag{7}$$

where $\mathbf{H}^l$ is the output of the l-th attention layer, $\mathbf{W}_Q^l$, $\mathbf{W}_K^l$, and $\mathbf{W}_V^l$ are learnable query, key, and value projection matrices respectively, and $\text{PE}(\cdot)$ represents positional encoding. To better capture the sequential modeling patterns, we further incorporate bidirectional LSTM (Greff et al., 2016) into our architecture:

$$\mathbf{out}_t = [LSTM(\overrightarrow{\mathbf{h}}_t); LSTM(\overleftarrow{\mathbf{h}}_t)], \tag{8}$$

where $\mathbf{out}_t$ is the output at time step t, $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$ denote the forward and backward hidden states respectively, and $[\cdot;\cdot]$ represents concatenation.

For efficient sequence generation, we design the model to predict the entire CAD modeling sequence in parallel:

$$p(\widehat{\mathbf{M}}|\mathbf{z}, \Theta) = \prod_{i=1}^{N_c} p(\mathbf{t}_i, \widehat{\mathbf{p}}_i|\mathbf{z}, \Theta), \tag{9}$$

where $\widehat{\mathbf{M}}$ represents the predicted CAD modeling sequence, Θ denotes the model parameters, N_c is the number of CAD commands in the sequence, $\mathbf{t}_i$ represents the command type at position i, and $\widehat{\mathbf{p}}_i$ represents the predicted parameters for the command at position i.

3.2.3 LLM-Enhanced CAD Command Sequence Generation

The results predicted by TCADGen can assist LLM in generating more accurate CCS. In this paper, we propose an enhanced CCS generation process based on large language models, referred to as CADLLM. This process combines the CCS generated by TCADGen, prediction confidence information, and user-provided description, enabling the precise generation of CCS through a fine-tuned large language model.

During the training phase, the fine-tuned model is trained on 1,000 samples (the choice of dataset size is explained in the experimental section, RQ3) that are consistent with the distribution of the dataset. The task is designed such that the output of TCADGen (predicted CCS and its confidence) serves as the query for supervised fine-tuning (SFT) to generate the correct CCS as the response. By learning the mapping between CCS and confidence, CADLLM corrects the errors generated by TCADGen and produces more accurate CCS.

During the inference phase, CADLLM acts as the inference model. It takes as input the user's parameters, appearance description, CCS generated by TCADGen based on user requirements, and the confidence information and outputs the enhanced CCS data.

4 Experiments

In our experiments, we address three research questions to evaluate the effectiveness of the proposed methodologies. For a detailed description of the dataset and parameters, please refer to Appendix A.2 and A.3.

RQ1: How effective is the proposed LLM-based semi-automated annotation framework in improving the quality of annotations for CAD datasets?

We have proposed a human-machine semi-automated CAD dataset annotation process that utilizes large models for annotation and verification to ensure quality. For the parameter description generation task, both the description generation and describe quality inspection stages utilize the gemma-2-27b-it (Google, 2024) model (the reasons for model selection and parameter description generation evaluation are provided in Appendix A.6). The reflection optimization process uses the gemma-2-27b-it model. For the appearance description generation task, the VLLM used for the description generation stage is the Llama-3.2-11B-Vision-Instruct (Meta, 2024) model, while the point cloud model uses PointLLM (Xu et al., 2024b). The description quality inspection stage also utilizes the gemma-2-27b-it model.

The experimental results demonstrate that this framework significantly improves annotation quality over the Text2CAD baseline (Khan, 2025) across most evaluation metrics.The experimental results are presented in Table 1, with the TCADGen (Text2CAD-dataset) row displaying the corresponding results. For overall performance, command line accuracy improves from 0.804 to 0.890, an increase of about 8.6 percentage points. The framework shows enhanced performance on commands such as Arc, Circle, and Extrude. Notably, the AUC-F1 score for Circle operations increases from 0.731 to 0.771, reflecting improved annotation capability for circular geometric elements. However, performance on complex operations, such as Ex-

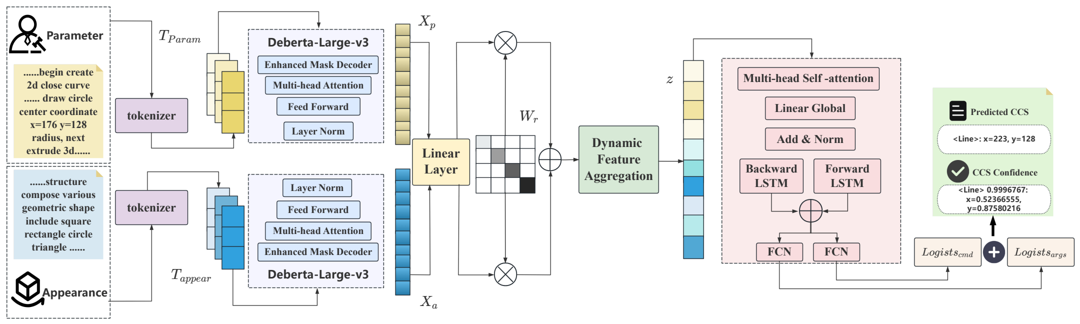


Figure 3: Transformer-based CAD Generator (TCADGen).

Models	**Avg Command**			**Each Command**							
				Line		Arc		Circle		Extrude	
	ACC	F1	AUC	AUC	F1	AUC	F1	AUC	F1	AUC	F1
DeepCAD	0.571	0.606	0.747	0.648	0.797	0.540	0.588	0.587	0.551	0.616	0.615
Text2CAD	0.840	0.722	0.819	0.763	0.904	0.584	0.601	0.751	0.701	0.772	0.705
BERT fine-tuned w/o	0.807	0.731	0.828	0.760	0.937	0.637	0.690	0.757	0.690	0.711	0.622
Dual-channel w/o	0.847	0.769	0.850	0.791	0.945	0.668	0.523	0.802	0.762	0.746	0.684
TCADGen(Text2CAD-dataset)	0.804	0.731	0.822	0.760	0.920	0.613	0.590	0.780	0.731	0.795	0.767
TCADGen	0.890	0.771	0.854	0.808	0.950	0.682	0.642	0.837	0.771	0.781	0.746
TCADGen+CADLLM (Ours)	**0.966**	**0.947**	**0.962**	**0.957**	**0.979**	**0.925**	**0.924**	**0.959**	**0.960**	**0.942**	**0.946**

Table 1: Comparison of different models and datasets for CAD command prediction. The Average Command section shows the overall performance metrics, while the Per Command section shows detailed AUC and F1 scores for a specific command type. Bold numbers indicate the best performance.

trude, shows a slight decrease from 0.767 to 0.746, suggesting areas for further improvement.

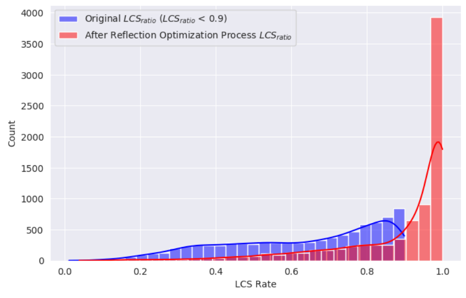


Figure 4: Distribution of $\text{LCS}_{\text{ratio}}$ scores before and after the Reflection Optimization Process.

Additionally, we assess the reflection optimization process for enhancing CCS parameter descriptions with low quality ($\text{LCS}_{\text{ratio}} < 0.9$). This process, which includes error analysis, reflective feedback, and re-evaluation, shows a significant increase in confidence scores ($t = -137.76, p < 0.001$), confirming its effectiveness. Histogram and Kernel Density Estimate (KDE) analyses (see Fig. 4) reveal that after optimization, $\text{LCS}_{\text{ratio}}$ scores cluster near 1.0, indicating improved reliability and consistency.

RQ2: How well does TCADGen perform in translating the human modeling language into standard CAD command sequences?

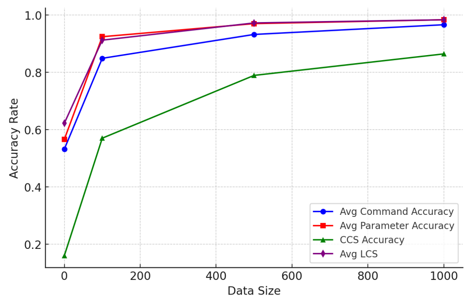


Figure 5: The impact of data size on CADLLM performance. As the data size increases from 0 to 1000 samples.

As shown in Table 1, the experimental results demonstrate that TCADGen significantly outperforms existing methods in generating CAD modeling sequences. Compared to the DeepCAD baseline, TCADGen achieves a 31.8 percentage point improvement in average command accuracy and surpasses DeepCAD in overall performance metrics, with an $F1$ score of 0.771 and AUC of 0.854. Additionally, in comparison to Text2CAD, TCADGen shows a 5 percentage point improvement in average command accuracy and demon-

Input	Model	Acc.	Avg LCS_{ratio}
Parameter Description and Appearance Description	GPT-4o(prompt)	0.617	0.977
	Llama 3.2 3B(prompt)	0.328	0.688
	Llama 3.2 3B(SFT)	0.621	0.883
Transformer-based Prediction of CCS and CCS Confidence	GPT-4o(prompt) + TCADGen	0.670	0.947
	Claude-3.5(prompt) + TCADGen	0.812	0.963
	Llama 3.2 3B(prompt) + TCADGen	0.328	0.683
	CADLLM + DeepCAD	0.677	0.978
	CADLLM + Text2CAD	0.710	0.974
	CADLLM + TCADGen(ours)	**0.864**	**0.983**

Table 2: Performance comparison of different methods for CAD command sequence generation. Our proposed CADLLM + TCADGen achieves the best performance in both accuracy and average LCS_{ratio}.

Method	Model	CD↓	MMD↓	JSD↓
Transformer-based	DeepCAD	169.93	31.91	45.03
	Text2CAD	142.83	28.98	40.23
	CAD Translator	-	2.94	10.92
	TCADGen	120.99	21.36	35.25
LLM-based	CADFusion	45.67	3.49	17.11
	DeepCAD+CADLLM	4.25 (↓ 165.68)	3.13 (↓ 28.78)	8.58 (↓ 36.45)
	Text2CAD+CADLLM	4.31 (↓ 138.52)	3.12 (↓ 25.86)	8.42 (↓ 31.81)
	TCADGen+CADLLM(ours)	**3.12** (↓ 117.87)	**2.78**(↓ 18.58)	**8.38** (↓ 26.87)

Table 3: Quantitative comparison of Chamfer Distance (CD), Minimum Matching Distance (MMD), and Jensen-Shannon Divergence (JSD) metrics, where lower values indicate better performance (↓).

strates better performance for complex modeling commands, such as Line (AUC from 0.648 to 0.808), Arc (AUC from 0.540 to 0.682), and Circle (AUC from 0.587 to 0.837). The ablation studies further validate this effectiveness, showing that removing CAD-specific BERT fine-tuning leads to a 4 percentage points drop in performance, while the dual-channel architecture significantly improves complex command generation, e.g., Extrude ($F1 = 0.7457$), indicating better distribution alignment and semantic consistency with ground truth commands.

RQ3: How effective are LLM in correcting TCADGen-generated CAD command sequences?

We compare two methods of using LLMs to generate CCS: one that directly generates CCS using parameter and appearance descriptions and the other that combines transformer-based predictions with CCS confidence scores. Even with fine-tuning, the direct description method, such as Llama 3.2 3B with prompt or LoRA, performs worse than the Transformer-based approach. For example, Llama achieves 32.8% accuracy (LCS_{ratio} 0.6881), while GPT-4o with predictions and confidence improves to 67.0% accuracy (LCS_{ratio} 0.9477). The optimal method, CADLLM + TCADGen, achieves 86.4% accuracy and LCS_{ratio} of 0.983, demonstrating that combining Transformer predictions with confidence scores is significantly more effective than direct generation from descriptions.

The evaluation results are presented in Table 3 for model generation quality. Among Transformer-based methods, TCADGen achieves the best performance, with a Chamfer Distance (CD) of 120.99, Minimum Matching Distance (MMD) of 21.36, and Jensen-Shannon Divergence (JSD) of 35.25. In comparison, CAD Translator (Li et al., 2024), which employs a one-stage framework, shows inferior performance, with a CD of 142.83. The introduction of CADLLM significantly enhances the performance of Transformer-based models. Specifically, TCADGen+CADLLM (ours) achieves a CD of 4.52 (↓116.47), MMD of 2.78 (↓18.17), and JSD of 8.38 (↓26.57). Meanwhile, CADFusion (Wang et al., 2025), which utilizes a two-stage training process with LLaMA-3-8b-Instruct, demonstrates lower accuracy, with a CD of 45.67. In contrast, our approach, which employs the smaller LLaMA-3.2-3b-Instruct model, outperforms both CADFusion and CAD Translator in all key metrics, further highlighting the effectiveness of CADLLM.

Our experimental results show that increasing the fine-tuning data size (from 0 to 1000 samples) leads to substantial performance improvements in CADLLM (Fig. 5, Table 10). Specifically, CCS accuracy improves from 16.0% to 86.4%, while LCS_{ratio} increases from 0.622 to 0.983. The performance improves only slightly after 500 samples, showing that 1000 samples achieve the best model performance and training cost balance.

5 Conclusion

This paper proposes a CAutoD framework that combines LLMs with domain-specific CAD parametrization to generate CAD models from user descriptions in the detailed design phase. We have demonstrated that, owing to the key innovations such as a semi-automated data annotation pipeline, TCAD-Gen, and LLM-assisted enhancement, the framework has significantly outperformed the existing methods in CAD modeling sequence generation accuracy and efficiency, making it a valuable tool for generating precise CAD models from textual prompts. Future work will focus on scaling the framework and exploring its applications in broader industrial design scenarios.

Limitations

Our framework demonstrates promising results in generating CAD modeling sequences, but several challenges remain. The semi-automatic data annotation process is resource-intensive, requiring a large number of LLM calls and manual verification for quality control. While automation reduces human effort, inconsistencies between multi-view descriptions and point cloud representations still require intervention, limiting scalability for large datasets.

The imbalance in command distributions affects model robustness, as certain operations, such as "Line," appear significantly more often than others like "Arc." This bias in training data leads to better performance on frequent commands while limiting generalization to complex 3D geometries that require underrepresented operations.

While the framework effectively generates and optimizes CAD sequences, it does not explicitly incorporate geometric constraints or structural reasoning, resulting in syntactically correct sequences that may not always align with practical design requirements. Future work could explore integrating geometric priors and constraint-aware learning to improve the reliability and applicability of automated CAD modeling.

Our framework focuses on the detailed design process during the CAD design phase and does not provide adequate support for the conceptual design phase, where parameter descriptions may be incomplete or vague.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 52405253), by the Ministry of Science and Technology of China (Grant No. H20240917), and by the Sichuan Province Science and Technology Innovation Seedling Project (Grant No. MZGC20240134).

References

Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Zaidi, Megan Langwasser, Wei Xu, and Matthew Gombolay. 2024. Generating cad code with vision-language models for 3d designs. *arXiv preprint arXiv:2410.05340*.

Joo Hock Ang, Cindy Goh, and Yun Li. 2016. Smart design for ships in a smart product through-life and industry 4.0 environment. In *2016 IEEE congress on evolutionary computation (CEC)*, pages 5301–5308. IEEE.

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

Robin Trulssen Bye, Ottar Osen, Webjørn Rekdalsbakken, Birger Skogeng Pedersen, and Ibrahim Hameed. 2017. An intelligent winch prototyping tool. In *Proceedings of the 31st European Conference on Modelling and Simulation (ECMS'17)*. ECMS European Council for Modelling and Simulation.

Lihu Chen and Gaël Varoquaux. 2024. What is the role of small models in the llm era: A survey. *arXiv preprint arXiv:2409.06857*.

Xiaoqun Dai and Yan Hong. 2024. Fabric mechanical parameters for 3d cloth simulation in apparel cad: a systematic review. *Computer-Aided Design*, 167:103638.

Yuhao Du, Shunian Chen, Wenbo Zan, Peizhao Li, Mingxuan Wang, Dingjie Song, Bo Li, Yan Hu, and Benyou Wang. 2024. Blenderllm: Training large language models for computer-aided design with self-improvement. *arXiv preprint arXiv:2412.14203*.

Rubin Fan, Fazhi He, Yuxin Liu, Yupeng Song, Linkun Fan, and Xiaohu Yan. 2025. A parametric and feature-based cad dataset to support human-computer interaction for advanced 3d shape learning. *Integrated Computer-Aided Engineering*, 32(1):75–96.

Jiang Fang, Zhicheng Zhang, Jiyan Sun, Jiadong Fu, Haonan He, Yinlong Liu, and Wei Ma. 2024. Improve model robustness in less time than it takes to drink a cup of coffee with plug-and-play plugins. In *Proceedings of the Asian Conference on Computer Vision*, pages 1520–1531.

Google. 2024. Gemma 2 27b. Accessed: February 5, 2025.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2016. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. 2022. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–18.

Sadil Khan. 2025. text2cad-dataset. Accessed: 2025-02-16.

Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. 2020. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 175–191. Springer.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9601–9611.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Xueyang Li, Yu Song, Yunzhong Lou, and Xiangdong Zhou. 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*, pages 8461–8470.

Yun Li. 2022. What should "industry 4.0" differ from previous industrial revolutions? In *2022 27th International Conference on Automation and Computing (ICAC)*, pages 1–1. IEEE.

JianXing Liao, JunYan Xu, SiCheng He, ZeKe Chen, Shui Yu, and Yun Li. 2024. Autoforma: A large language model-based multi-agent for computer-automated design. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1284–1289. IEEE.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Giulio Marchesi, Alvise Camurri Piloni, Vanessa Nicolin, Gianluca Turco, and Roberto Di Lenarda. 2021. Chairside cad/cam materials: current trends of clinical uses. *Biology*, 10(11):1170.

Meta. 2024. Llama 3.2 11b vision instruct. Accessed: February 5, 2025.

Mensah Kwabena Patrick, Adebayo Felix Adekoya, Ayidzoe Abra Mighty, and Baagyire Y Edward. 2022. Capsule networks–a survey. *Journal of King Saud University-computer and information sciences*, 34(1):1295–1310.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. 2024. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*.

Martin Rapp, Hussam Amrouch, Yibo Lin, Bei Yu, David Z Pan, Marilyn Wolf, and Jörg Henkel. 2021. Mlcad: A survey of research in machine learning for cad keynote paper. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(10):3162–3181.

Hanh Thi Hong Tran, Nishan Chatterjee, Senja Pollak, and Antoine Doucet. 2024. Deberta beats behemoths: A comparative analysis of fine-tuning, prompting, and peft approaches on legallensner. In *Proceedings of the Natural Legal Language Processing Workshop 2024*, pages 371–380.

Kashi Venkatesh Vishwanath, Diwaker Gupta, Amin Vahdat, and Ken Yocum. 2009. Modelnet: Towards a datacenter emulation environment. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, pages 81–82. IEEE.

Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. 2025. Text-to-cad generation through infusing visual feedback in large language models. *arXiv preprint arXiv:2501.19054*.

Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2021. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4):1–24.

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

Sifan Wu, Amir Hosein Khasahmadi, Mor Katz, Pradeep Kumar Jayaraman, Yewen Pu, Karl Willis, and Bang Liu. 2024. Cadvlm: Bridging language and vision in the generation of parametric cad sketches. In *European Conference on Computer Vision*, pages 368–384. Springer.

Canwen Xu, Yichong Xu, Shuohang Wang, Yang Liu, Chenguang Zhu, and Julian McAuley. 2023. Small models are valuable plug-ins for large language models. *arXiv preprint arXiv:2305.08848*.

Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. 2024a. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm. *arXiv preprint arXiv:2411.04954*.

Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. 2024b. Pointllm: Empowering large language models to understand point clouds. In *ECCV*.

Mohsen Yavartanoo, Sangmin Hong, Reyhaneh Neshatavar, and Kyoung Mu Lee. 2024. Text2cad: Text to 3d cad generation via technical drawings. *arXiv preprint arXiv:2411.06206*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Appendix

A.1 CAD Command Sequence

In the CAD modeling task proposed in this paper, we adopt a command sequence representation based on sketch-extrusion, and the generated sequence is referred to as the CAD Command Sequence (CCS). The CCS example is shown in Fig. 6, and the detailed CCS parameters are presented in Table 4.

The sequence consists of two-dimensional sketch commands and three-dimensional extrusion commands. Specifically, the two-dimensional sketch section begins with the tag <SOL>, representing the start of a closed contour. Each contour can be constructed using three basic curve commands:

- The **Line** command determines the direction and length of the straight line by specifying the endpoint coordinates (x, y);
- The **Arc** command requires the endpoint coordinates (x, y), sweep angle α, and direction flag f to define the shape and direction of the arc;
- The **Circle** command creates a complete circular contour by specifying the center coordinates (x, y) and radius r.

Once the two-dimensional sketch is completed, the **Extrude** command E transforms it into a three-dimensional solid. This command includes several key parameters:

- The spatial orientation of the sketch plane is determined by the Euler angles (θ, ϕ, γ);
- The translation vector (p_x, p_y, p_z) specifies the position of the origin of the sketch plane;
- The scaling factor s controls the size of the contour;
- The bidirectional extrusion distances (e_1, e_2) determine the thickness of the solid;
- The Boolean operation type b is used to specify the interaction with existing geometry, which can be a new entity, union, subtraction, or intersection operation;
- The extrusion type u specifies the extrusion method, including unidirectional, symmetric, or bidirectional extrusion.

The CAD sequence can be modeled by creating two-dimensional sketches and repeatedly applying extrusion commands.

A.2 Dataset

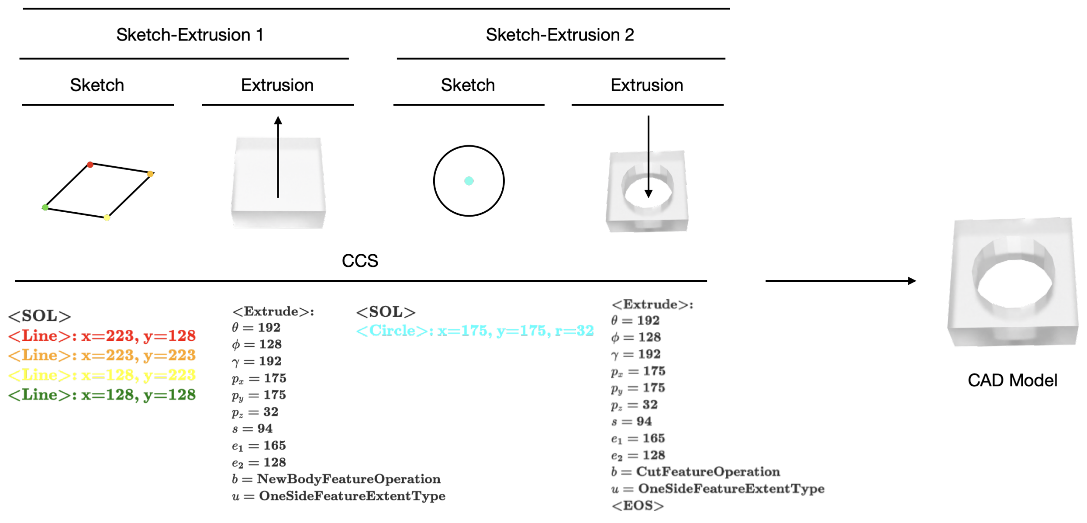
The DeepCAD dataset (Wu et al., 2021), derived from the ABC dataset (Koch et al., 2019), is a large-scale collection of 178,000 parametric CAD models designed for CAD model generation and reconstruction. Unlike other datasets, such as Fusion 360 (Willis et al., 2021) with only 8,000 instances, DeepCAD includes sketch-extrusion sequences, enabling models to learn procedural modeling behaviors rather than static geometry. Its extensive cross-industry coverage ensures strong generalization to external datasets, reducing concerns about overfitting. For this study, DeepCAD was preprocessed to ensure consistency, remove incomplete instances, and improve training efficiency. The corresponding STL file was generated for each sample based on the original model information provided in the dataset's JSON files. Simultaneously, the CCS descriptions were extracted from the JSON files. The generated STL files were then sampled from nine different viewpoints to produce multi-view images. Additionally, point sampling was performed files, generating point cloud files containing 8,000 points. The subsequent training and validation phases included only those samples for which the STL files, multi-view images, and point cloud files were successfully generated. The final dataset used in this experiment consisted of 155,503 training samples and 5,647 test samples.

A.3 Parameter Setting

We use two main parts in our system: a TCADGen model and a fine-tuned LLM(CADLLM).

For TCADGen, we build a simple Transformer decoder. It has 4 layers and 8 attention heads in each layer. The model's base size is $d_{model} = 256$, and it can handle sequences up to 300 tokens. The argument size is 256, and the feedforward layers are 512 units wide. We use a dropout of 0.1 throughout the model to avoid overfitting. We trained the model using a single A6000 GPU for 24 hours. The inference of 5,647 data samples takes 240 seconds.

For the LLM part, Our system uses two main parts and fine-tune it using LoRA, leveraging the LlamaFactory framework (Zheng et al., 2024) for fine-tuning. We set the basic batch size to 1 and use gradient accumulation for 8 steps. This gives



```
<SOL>
<Line>: x=223, y=128
<Line>: x=223, y=223
<Line>: x=128, y=223
<Line>: x=128, y=128
<Extrude>:
θ = 192
ϕ = 128
γ = 192
p_x = 175
p_y = 175
p_z = 32
s = 94
e_1 = 165
e_2 = 128
b = NewBodyFeatureOperation
u = OneSideFeatureExtentType
<SOL>
<Circle>: x=175, y=175, r=32
<Extrude>:
θ = 192
ϕ = 128
γ = 192
p_x = 175
p_y = 175
p_z = 32
s = 94
e_1 = 165
e_2 = 128
b = CutFeatureOperation
u = OneSideFeatureExtentType
<EOS>
```

Figure 6: Illustration of the CAD Command Sequence (CCS) representation. The process shows two sketch-extrusion operations: (1) creating a base by sketching a rectangle using Line commands with specified coordinates, followed by an extrusion operation to form a 3D block; (2) adding a circular hole by sketching a circle and applying a cut extrusion operation.

us an effective batch size of 8. We use a learning rate of $1e^{-4}$ with cosine decay and 10% warmup steps. We train the model for 3 epochs with BF16 precision. This helps us save memory and train faster. The model can take inputs up to 4096 tokens long. We check the model's performance every 500 step using 20% of our data as a validation set. We save the model version that performs best. We trained and performed inference using a single A6000 GPU. The training duration was 2 hours. For inference, we utilized the vllm framework (Kwon et al., 2023) with a concurrency of 2. Inference of 5,647 test samples took 1.5 hours

A.4 Application of the Framework for Motorcycle Frame Model Generation

In this work, we propose a framework aimed at detailed CAD design, where both the parameters and the full specifications of the model are predefined. The framework generates individual components of the motorcycle frame, which are then manually assembled to produce the final CAD model. Specifically, the proposed framework facilitates the rapid generation of parts that conform to the specified requirements, based on textual descriptions of both appearance and parameters. The generated models are presented in Fig. 7.

A.5 Examples of annotation processes

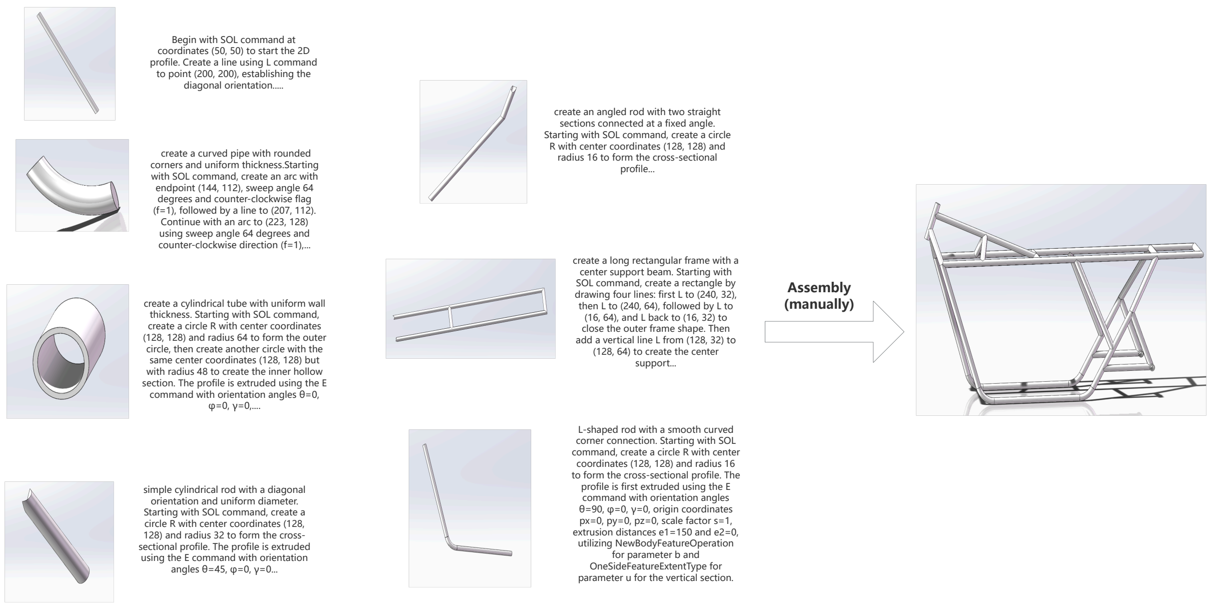
A.5.1 Parameter Description Labeling Process

Figure 8 shows the Parameter Description Labeling Process.Table 5 shows the parameter descriptions generated by the annotation process proposed in this paper, where $\text{LCS}_{\text{ratio}}$ refers to the comparison between the CCS generated by the large language model in the description quality control stage and the ground truth. When the $\text{LCS}_{\text{ratio}}$ of a description (e.g., No.00159955) is greater than 0.9, it indicates that the parameter description is successfully synthesized and can be adopted. If the $\text{LCS}_{\text{ratio}}$ is below 0.9 (e.g., No.00262583), the description enters the subsequent reflection and optimization process.

Reflection Optimization Process The original CCS and the optimized descriptions are shown in Table 6. The reflection optimization process is detailed in Table 7, where, shows the original CCS and optimized descriptions after two rounds of optimization, the description of the generated parameters achieves an $\text{LCS}_{\text{ratio}}$ of 1.

A.5.2 Appearance Description Annotation

The examples of appearance description generation are shown in Table 8. For each data sample, in the description generation stage, both multi-view descriptions (generated by the VLLM) and point cloud descriptions (generated by the point-cloud-



Begin with SOL command at coordinates (50, 50) to start the 2D profile. Create a line using L command to point (200, 200), establishing the diagonal orientation.....

create a curved pipe with rounded corners and uniform thickness.Starting with SOL command, create an arc with endpoint (144, 112), sweep angle 64 degrees and counter-clockwise flag (f=1), followed by a line to (207, 112). Continue with an arc to (223, 128) using sweep angle 64 degrees and counter-clockwise direction (f=1),...

create a cylindrical tube with uniform wall thickness. Starting with SOL command, create a circle R with center coordinates (128, 128) and radius 64 to form the outer circle, then create another circle with the same center coordinates (128, 128) but with radius 48 to create the inner hollow section. The profile is extruded using the E command with orientation angles θ=0, φ=0, γ=0,....

simple cylindrical rod with a diagonal orientation and uniform diameter. Starting with SOL command, create a circle R with center coordinates (128, 128) and radius 32 to form the cross-sectional profile. The profile is extruded using the E command with orientation angles θ=45, φ=0, γ=0...

create an angled rod with two straight sections connected at a fixed angle. Starting with SOL command, create a circle R with center coordinates (128, 128) and radius 16 to form the cross-sectional profile...

create a long rectangular frame with a center support beam. Starting with SOL command, create a rectangle by drawing four lines: first L to (240, 32), then L to (240, 64), followed by L to (16, 64), and L back to (16, 32) to close the outer frame shape. Then add a vertical line L from (128, 32) to (128, 64) to create the center support...

L-shaped rod with a smooth curved corner connection. Starting with SOL command, create a circle R with center coordinates (128, 128) and radius 16 to form the cross-sectional profile. The profile is first extruded using the E command with orientation angles θ=90, φ=0, γ=0, origin coordinates px=0, py=0, pz=0, scale factor s=1, extrusion distances e1=150 and e2=0, utilizing NewBodyFeatureOperation for parameter b and OneSideFeatureExtentType for parameter u for the vertical section.

Figure 7: The motorcycle frame design process: Parts generated using our framework are manually assembled to create the final frame. The figure shows the individual components and their integration into the complete design.

based large language model) are produced simultaneously. The two descriptions are verified using a large language model in the description quality control stage. For example, in sample No. 00247561, the multi-view and point cloud descriptions were inconsistent, and the model's output was flagged as False. This sample was then sent for manual annotation. In contrast, for sample No. 00041425, the multi-view and point cloud descriptions were consistent, and the model's output was True, allowing for generating a unified description based on both representations.

A.5.3 LLMCAD Enhancement

Figure 9 shows the final CCS generated by the CADLLM based on the TCADGen CCS output and confidence output. The red and green areas highlight the modified sections, which correspond to the parts where the confidence prediction was lower. This demonstrates that the CAD LLM task involves generating a new, correct CCS based on the existing parameter descriptions, considering the confidence levels and the TCADGen output. In this example, the CCS output from the CAD LLM exactly matches the ground truth.

A.6 The Ability of LLMs to Generate Parameter Descriptions

To assess the ability of LLMs in generating CAD modeling descriptions, we introduce a two-phase evaluation framework consisting of forward generation and backward validation. We evaluate a test set of 1000 samples that reflect the natural distribution of our dataset. LLMs generate CCS descriptions based on given prompts in the forward generation phase. The backward validation phase employs different LLMs to verify these generated descriptions.

We define four evaluation criteria based on $\text{LCS}_{\text{ratio}}$:

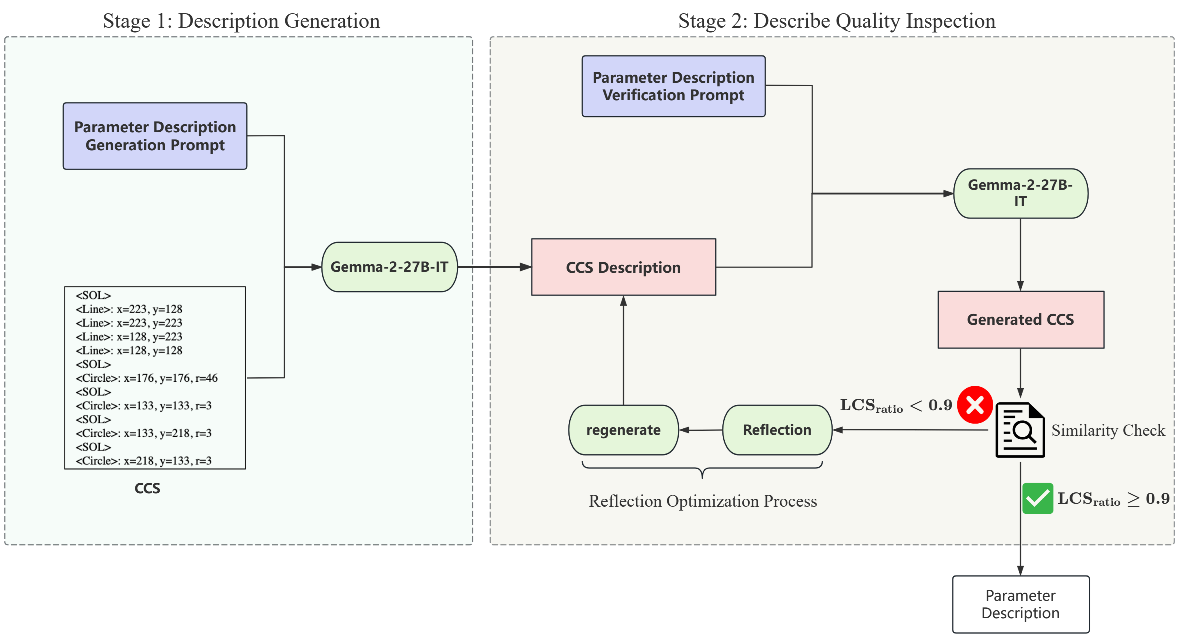
1. Exact matches: $\text{LCS}_{\text{ratio}} = 1$
2. High-quality matches: $\text{LCS}_{\text{ratio}} \geq 0.98$
3. Acceptable matches: $\text{LCS}_{\text{ratio}} \geq 0.9$
4. Mean $\text{LCS}_{\text{ratio}}$ across all test samples

The proportions for each criterion are computed as follows:

$$\text{Proportion}_{\text{criterion}} = \frac{\text{count}(\text{LCS}_{\text{ratio}} \geq \text{criterion})}{N}, \tag{10}$$

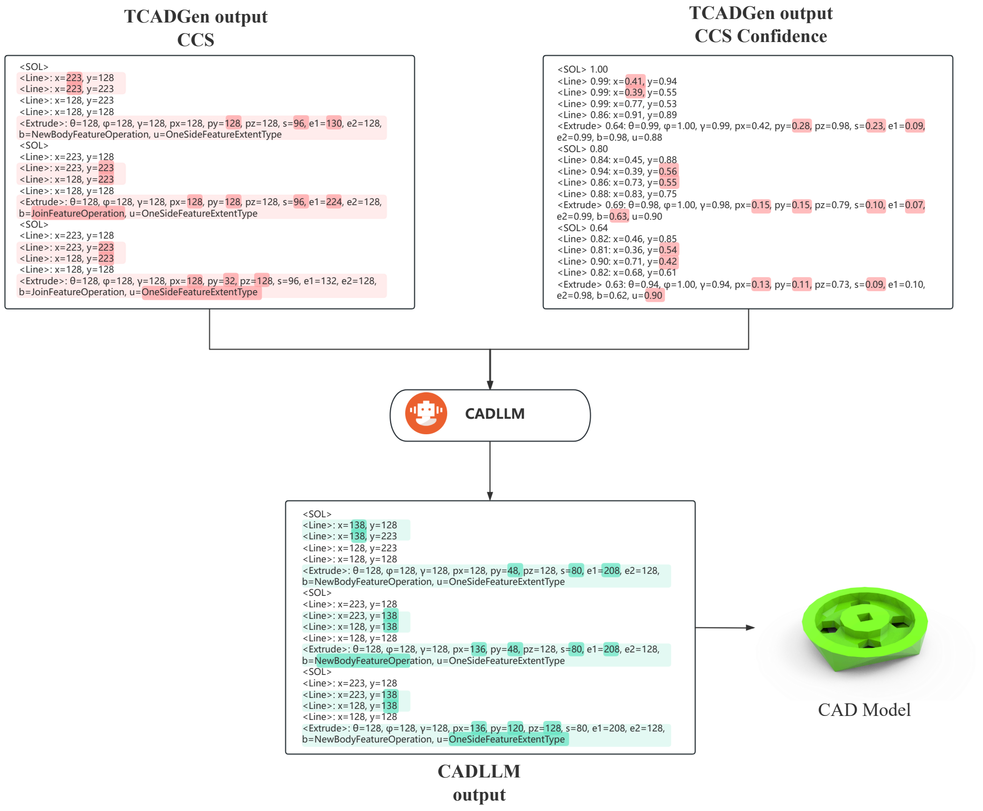
where N is the total number of samples, and $\text{LCS}^{(i)}_{\text{ratio}}$ is the $\text{LCS}_{\text{ratio}}$ for the i-th sample.The average $\text{LCS}_{\text{ratio}}$ is calculated as:

$$\text{Average LCS}_{\text{ratio}} = \frac{1}{N}\sum_{i=1}^{N} \text{LCS}^{(i)}_{\text{ratio}}, \tag{11}$$


Stage 1: Description Generation
Stage 2: Describe Quality Inspection
Parameter Description Generation Prompt
Gemma-2-27B-IT
CCS Description
Parameter Description Verification Prompt
Gemma-2-27B-IT
Generated CCS
Similarity Check
$\mathrm{LCS_{ratio}} < 0.9$
Reflection
regenerate
Reflection Optimization Process
$\mathrm{LCS_{ratio}} \geq 0.9$
Parameter Description
CCS

```
<SOL>
<Line>: x=223, y=128
<Line>: x=223, y=223
<Line>: x=128, y=223
<Line>: x=128, y=128
<SOL>
<Circle>: x=176, y=176, r=46
<SOL>
<Circle>: x=133, y=133, r=3
<SOL>
<Circle>: x=133, y=218, r=3
<SOL>
<Circle>: x=218, y=133, r=3
```

Figure 8: Parameter Description Annotation Process Flow.



```
<SOL>
<Line>: x=223, y=128
<Line>: x=223, y=223
<Line>: x=128, y=223
<Line>: x=128, y=128
<Extrude>: θ=128, φ=128, γ=128, px=128, py=128, pz=128, s=96, e1=130, e2=128,
b=NewBodyFeatureOperation, u=OneSideFeatureExtentType
<SOL>
<Line>: x=223, y=128
<Line>: x=223, y=223
<Line>: x=128, y=223
<Line>: x=128, y=128
<Extrude>: θ=128, φ=128, γ=128, px=128, py=128, pz=128, s=96, e1=224, e2=128,
b=JoinFeatureOperation, u=OneSideFeatureExtentType
<SOL>
<Line>: x=223, y=128
<Line>: x=223, y=223
<Line>: x=128, y=223
<Line>: x=128, y=128
<Extrude>: θ=128, φ=128, γ=128, px=128, py=32, pz=128, s=96, e1=132, e2=128,
b=JoinFeatureOperation, u=OneSideFeatureExtentType
```

```
<SOL> 1.00
<Line> 0.99: x=0.41, y=0.94
<Line> 0.99: x=0.39, y=0.55
<Line> 0.99: x=0.77, y=0.53
<Line> 0.86: x=0.91, y=0.89
<Extrude> 0.64: θ=0.99, φ=1.00, γ=0.99, px=0.42, py=0.28, pz=0.98, s=0.23, e1=0.09,
e2=0.99, b=0.98, u=0.88
<SOL> 0.80
<Line> 0.84: x=0.45, y=0.88
<Line> 0.94: x=0.39, y=0.56
<Line> 0.86: x=0.73, y=0.55
<Line> 0.88: x=0.83, y=0.75
<Extrude> 0.69: θ=0.98, φ=1.00, γ=0.98, px=0.15, py=0.15, pz=0.79, s=0.10, e1=0.07,
e2=0.99, b=0.63, u=0.90
<SOL> 0.64
<Line> 0.82: x=0.46, y=0.85
<Line> 0.81: x=0.36, y=0.54
<Line> 0.90: x=0.71, y=0.42
<Line> 0.82: x=0.68, y=0.61
<Extrude> 0.63: θ=0.94, φ=1.00, γ=0.94, px=0.13, py=0.11, pz=0.73, s=0.09, e1=0.10,
e2=0.98, b=0.62, u=0.90
```

```
<SOL>
<Line>: x=138, y=128
<Line>: x=138, y=223
<Line>: x=128, y=223
<Line>: x=128, y=128
<Extrude>: θ=128, φ=128, γ=128, px=128, py=48, pz=128, s=80, e1=208, e2=128,
b=NewBodyFeatureOperation, u=OneSideFeatureExtentType
<SOL>
<Line>: x=223, y=128
<Line>: x=223, y=138
<Line>: x=128, y=138
<Line>: x=128, y=128
<Extrude>: θ=128, φ=128, γ=128, px=136, py=48, pz=128, s=80, e1=208, e2=128,
b=NewBodyFeatureOperation, u=OneSideFeatureExtentType
<SOL>
<Line>: x=223, y=128
<Line>: x=223, y=138
<Line>: x=128, y=138
<Line>: x=128, y=128
<Extrude>: θ=128, φ=128, γ=128, px=136, py=120, pz=128, s=80, e1=208, e2=128,
b=NewBodyFeatureOperation, u=OneSideFeatureExtentType
```

Figure 9: Final CCS generated by CAD LLM using TCADGen CCS output and confidence, with red and green areas indicating modified sections.

We present the raw evaluation data in Table 11. Table 12 summarizes the results for the Best Reverse Validation Model, where the model with the highest proportion of LCS Greater than 0.9 for each forward generation model is selected as the Best Reverse Validation Model. The results show strong performance from both closed-source and open-source models. In forward generation, Gemini-1.5-Pro and GPT-4o achieve LCSratio values above 93.9%. Among open-source models, Meta-Llama-3.1-70B-Instruct-Turbo and Meta-Llama-3.1-405B-Instruct-Turbo perform well. Notably, Gemini-2-27B-IT shows competitive results despite its smaller size. For backward validation, we define the Best Reverse Model as the one that most frequently achieves LCSratio > 0.9. Gemini-2-27B-IT is the most reliable model for this validation task, demonstrating its effectiveness in accurately verifying CCS descriptions.

A.7 Prompts Used in Experiments

Table 13 presents all the prompt templates used in this experiment.

Command	**Parameter**	**Value Range**	**Description**
`<SOL>`	$\emptyset$	-	Contour Start Marker
Line	x	$[0, 255]$	End point x coordinate of the line segment
	y	$[0, 255]$	End point y coordinate of the line segment
Arc	x	$[0, 255]$	End point x coordinate of the arc
	y	$[0, 255]$	End point y coordinate of the arc
	α	$[0, 255]$	Sweep angle of the arc
	f	$\{0, 1\}$	Counterclockwise direction flag
Circle	x	$[0, 255]$	Center x coordinate of the circle
	y	$[0, 255]$	Center y coordinate of the circle
	r	$[0, 255]$	Radius of the circle
Extrude	θ	$[0, 255]$	Rotation angle around the x-axis of the sketch plane
	ϕ	$[0, 255]$	Rotation angle around the y-axis of the sketch plane
	γ	$[0, 255]$	Rotation angle around the z-axis of the sketch plane
	p_x	$[0, 255]$	x-coordinate of the sketch plane origin
	p_y	$[0, 255]$	y-coordinate of the sketch plane origin
	p_z	$[0, 255]$	z-coordinate of the sketch plane origin
	s	$[0, 255]$	Scaling factor for the sketch contour
	e_1	$[0, 255]$	Positive direction extrusion distance
	e_2	$[0, 255]$	Negative direction extrusion distance
	b	$\{7, 8, 9, 10\}$	Boolean operation type:
			7: NewBodyFeatureOperation
			8: JoinFeatureOperation
			9: CutFeatureOperation
			10: IntersectFeatureOperation
	u	$\{1, 2, 3\}$	Extrusion type:
			1: OneSideFeatureExtentType
			2: SymmetricFeatureExtentType
			3: TwoSidesFeatureExtentType
`<EOS>`	$\emptyset$	-	Sequence End Marker

Table 4: CAD Command Sequence (CCS) Parameters.

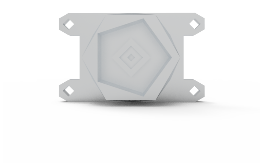
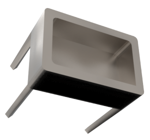
ID	Generated Parameter Description	$\text{LCS}_{\text{ratio}}$	Model Visualization
00159955	The sequence begins with creating two concentric circles to define the initial 2D closed curve. The first circle is centered at coordinates (176, 128) with a radius of 47, and the second circle, also centered at the same point, has a smaller radius of 40. This closed curve is then extruded with the extrusion parameters θ=128, φ=128, and γ=128, indicating the orientation of the sketch plane. The sketch plane's origin is set at (70, 128, 128), and a scale factor of 115 is applied to adjust the profile size. The extrusion distances are set to e1=142 and e2=128, specifying how far the shape extends in the extrusion direction. The operation type for the extrusion is "NewBodyFeatureOperation," meaning a new body is created as a result. The extrusion direction is one-sided, indicated by the parameter u=OneSideFeatureExtentType, ensuring the shape is extruded in one direction. Next, a new 2D closed curve is created with a combination of lines and arcs. Starting from the point (140, 128), a line is drawn to (195, 128), followed by an arc that sweeps counter-clockwise from (195, 128) to (195, 223) with a sweep angle α=128. A series of lines are then drawn to complete the curve, and two circles are added at the coordinates (128, 140) and (128, 211), each with a radius of 6. These circles are also extruded using the parameters θ=128, φ=128, and γ=128, with the origin set at (46, 70, 128) and a scale factor of 115. The extrusion distances are e1=142 and e2=128, with a "JoinFeatureOperation" applied to merge this new feature with the existing body. The extrusion is performed in a one-sided direction, ensuring the shape extends only in one direction...	0.99	
00262583	The modeling process begins with the creation of two circles, both centered at (176, 128) with a radius of 48. The first circle is extruded into a 3D feature with an extrusion direction defined by the parameters θ=128, φ=128, and γ=128, and the origin placed at (56, 170, 128). The extrusion uses a scale factor of 12 and extends by 176 units in the first phase and 128 units in the second, creating a new body i, extendingion. The second circle is then extruded using similar parameters, but the origin shifts to (188, 86, 128), and a "JoinFeatureOperation" is applied to merge the extrusion with the existing feature. Next, a more complex 2D closed curve is created using arcs and lines. The curve is defined with specific arc directions and sweep angles, forming a polygon that closes at the starting point. A circle with a radius of 4 is defined at (132, 128) and extruded using the same extrusion parameters, extending the shape into 3D...	0.74	

Table 5: Examples of parameter description generation.

Original CCS	Generated Parameter Description	$\text{LCS}_{\text{ratio}}$
<SOL> <Arc>: x=144, y=112, α=64, f=1 <Line>: x=207, y=112 <Arc>: x=223, y=128, α=64, f=1 <Line>: x=223, y=204 <Arc>: x=207, y=220, α=64, f=1 <Line>: x=144, y=220 <Arc>: x=128, y=204, α=64, f=1 <Line>: x=128, y=128 <SOL> <Arc>: x=144, y=114, α=64, f=1 <Line>: x=207, y=114 <Arc>: x=220, y=128, α=64, f=1 <Line>: x=220, y=190 <Arc>: x=207, y=204, α=64, f=1 <Line>: x=144, y=204 <Arc>: x=131, y=190, α=64, f=1 <Line>: x=131, y=128 <Extrude>: θ=192, φ=64, γ=192, px=105, py=121, pz=40, s=46, e1=148, e2=128, b=NewBodyFeatureOperation, u=OneSideFeatureExtentType	First, draw an arc with an end-point at x=144, y=112, a sweep angle of 64 degrees, and a counter-clockwise direction. Then, draw a line to x=207, y=112, followed by an arc to x=223, y=128 with a sweep angle of 64 degrees and a counter-clockwise direction. Next, draw a line to x=223, y=204, an arc to x=207, y=220 with a sweep angle of 64 degrees and a counter-clockwise direction, a line to x=144, y=220, and an arc to x=128, y=204 with a sweep angle of 64 degrees and a counter-clockwise direction. Finally, draw a line to x=128, y=128. Repeat the same sequence starting with an arc with an end-point at x=144, y=114, a sweep angle of 64 degrees, and a counter-clockwise direction. Then, extrude the 2D sketch by 148 units in the direction specified by θ=192, φ=64, γ=192, originating at px=105, py=121, pz=40, and scaling the profile by a factor of 46. The extrusion will extend for a distance of 128 units in the direction given by u=OneSideFeatureExtentType and will be treated as a NewBodyFeatureOperation.	0.675

Table 6: The original CCS and the first generated descriptions.

Round	Reflection Output	Optimization Output	$\text{LCS}_{\text{ratio}}$
1	**Generated vs. Real CCS Differences:** 1. Arc End Points: Generated CCS ends at $x = 223, y = 130$; real CCS at $x = 220, y = 128$. 2. Line End Points: Generated CCS ends at $x = 207, y = 114$; real CCS at $x = 207, y = 112$. 3. Extra Commands: Real CCS includes a line to $x = 220, y = 190$; generated CCS specifies $x = 223, y = 204$. 4. Arc Variation: Generated CCS ends an arc at $x = 207, y = 222$; real CCS at $x = 207, y = 204$. 5. Missing Variables: Generated CCS omits e_2 (128 in real CCS). **Problems in Parametric Description:** 1. Ambiguous Arcs: Incomplete parameters cause endpoint inconsistencies. 2. Unclear Pathing: Lacks instructions for connecting arcs and lines or closing loops. 3. Rotation Inconsistencies: Arc rotation directions are unclear. 4. Missing Line Dimensions: No explicit dimensions for connecting lines. 5. Undefined Parameters: e_2 is missing from the description. **Improvement Suggestions:** 1. Define arcs explicitly with endpoints, radii, and centers. 2. Clarify pathing, including connection rules and loop closures. 3. Standardize rotation directions. 4. Specify line dimensions clearly. 5. Define all variables, including e_2. 6. Add visual references to support the description.	First, draw an arc with an end-point at $x = 144, y = 112$, a sweep angle of 64 degrees, and a counter-clockwise direction. Then, draw a line to $x = 207, y = 112$, followed by an arc to $x = 220, y = 128$ with a sweep angle of 64 degrees and a counter-clockwise direction. Next, draw a line to $x = 220, y = 190$, an arc to $x = 207, y = 204$ with a sweep angle of 64 degrees and a counter-clockwise direction, a line to $x = 144, y = 204$, and an arc to $x = 131, y = 190$ with a sweep angle of 64 degrees and a counter-clockwise direction. Finally, draw a line to $x = 131, y = 128$. Repeat the same sequence starting with an arc with an end-point at $x = 144, y = 114$, a sweep angle of 64 degrees, and a counter-clockwise direction. Then, draw a line to $x = 207, y = 114$, followed by an arc to $x = 220, y = 128$ with a sweep angle of 64 degrees and a counter-clockwise direction. Next, draw a line to $x = 220, y = 190$, an arc to $x = 207, y = 204$ with a sweep angle of 64 degrees and a counter-clockwise direction, a line to $x = 144, y = 204$, and an arc to $x = 131, y = 190$ with a sweep angle of 64 degrees and a counter-clockwise direction. Finally, draw a line to $x = 131, y = 128$. Then, extrude the 2D sketch by 148 units in the direction specified by $\theta = 192$, $\phi = 64$, $\gamma = 192$, originating at $px = 105$, $py = 121$, $pz = 40$, and scaling the profile by a factor of 46. The extrusion will extend for a distance of 128 units in the direction given by u=OneSideFeatureExtentType and be treated as a NewBodyFeatureOperation.	0.737
2	**Generated vs. Real CCS Differences:** 1. Arc Direction: Generated CCS uses $f = -1$ (clockwise), real CCS uses $f = 1$ (counter-clockwise). 2. Arc Endpoint Discrepancy: Generated CCS ends at $x = 220, y = 128$, real CCS at $x = 223, y = 128$. 3. Line Mismatch: Generated CCS connects to $x = 220, y = 190$, real CCS to $x = 223, y = 204$. 4. Arc Endpoint Discrepancy: Later arcs end at $x = 207, y = 220$, instead of $x = 207, y = 204$. 5. Fewer Commands: Generated CCS omits lines and arcs. 6. Extrusion Parameters: Generated CCS uses $e_1 = 0$ and omits e_2; real CCS uses $e_1 = 148$ and $e_2 = 128$. **Problems in CAD Parametric Description:** 1. Unclear Arc Directions: f not explicitly defined. 2. Ambiguous Arc Coordinates: Lack of precise endpoint relationships. 3. Endpoint Proximity Issues: Close arc endpoints cause confusion. 4. Incomplete Extrusion Definitions: e_1 and e_2 inadequately described. **Suggestions for Improvement:** 1. Define arc directions explicitly. 2. Derive arc endpoints explicitly from previous commands. 3. Implement validation checks for consistency. 4. Provide clear extrusion parameter definitions. 5. Ensure sequential coherence in parametric commands.	First, draw an arc with an end-point at $x = 144, y = 112$, a sweep angle of 64 degrees, and a counter-clockwise direction ($f = 1$). Then, draw a line to $x = 207, y = 112$. Next, draw an arc to $x = 223, y = 128$ with a sweep angle of 64 degrees and a counter-clockwise direction ($f = 1$). Draw a line to $x = 223, y = 204$. Then, draw an arc to $x = 207, y = 220$ with a sweep angle of 64 degrees and a counter-clockwise direction ($f = 1$). After that, draw a line to $x = 144, y = 220$. Then, draw an arc to $x = 128, y = 204$ with a sweep angle of 64 degrees and a counter-clockwise direction ($f = 1$). Finally, draw a line to $x = 128, y = 128$. Repeat the same sequence starting with an arc with an end-point at $x = 144, y = 114$, a sweep angle of 64 degrees, and a counter-clockwise direction ($f = 1$). Then, draw a line to $x = 207, y = 114$. Next, draw an arc to $x = 220, y = 128$ with a sweep angle of 64 degrees and a counter-clockwise direction ($f = 1$). After that, draw a line to $x = 220, y = 190$. Then, draw an arc to $x = 207, y = 204$ with a sweep angle of 64 degrees and a counter-clockwise direction ($f = 1$). Draw a line to $x = 144, y = 204$. Next, draw an arc to $x = 131, y = 190$ with a sweep angle of 64 degrees and a counter-clockwise direction ($f = 1$). Finally, draw a line to $x = 131, y = 128$. Then, extrude the 2D sketch by 148 units in the direction specified by $\theta = 192$, $\phi = 64$, $\gamma = 192$, originating at $px = 105$, $py = 121$, $pz = 40$, and scaling the profile by a factor of 46. The extrusion will extend for a distance of 128 units in the direction given by u=OneSideFeatureExtentType and will be treated as a NewBodyFeatureOperation with parameters $e_1 = 148$ and $e_2 = 128$.	1.000

Table 7: Multiple-round Reflection and Optimization Results for CCS Generation.

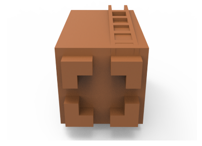
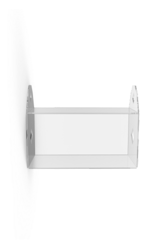
Model Visualization	**Point Cloud Description**	**Multi-View Description**	**LLM Determination**	**Comprehensive Description**
	The point cloud represents a 3D model of an engine, exhibiting intricate details. It includes components such as cylinders, connectors, and automotive parts, though the specific engine type is not identified. The model is likely intended for educational or demonstration purposes in mechanics or engineering.	The object is cube-shaped with a ladder on its top surface. It consists of a large cube, four smaller L-shaped protrusions on one side, and a ladder structure. The geometric form is simple, with clearly defined edges and shapes, which could be a building block or part of a larger assembly.	FALSE (Manually Annotated)	This 3D object has a cubic form with a ladder attached to its upper surface. It consists of a main cube, four smaller L-shaped protrusions, and a ladder-like structure.
	The 3D point cloud represents a square box with a closed and smooth surface, suggesting it could be a container or storage object. Its shape implies various uses, such as storing small items or being a building block. The lack of overall context makes its practical use unclear, but it may be intended for architectural modeling, digital art, or advanced mathematical modeling.	The 3D model appears as a cube with a cylindrical protrusion on one side, featuring a curved top and a flat bottom. The model has a cube base, a cylindrical body, and a curved top. The smooth surface and tapering cylindrical body suggest the model could serve as a container or vessel.	TRUE (Large Model Annotation)	The 3D model has a square box (cube) as the base. One side of the cube has a cylindrical protrusion with a curved top and flat bottom. The cylindrical body tapers subtly towards the curved top.

Table 8: Point Cloud and Multi-View Descriptions with Model Determinations and Comprehensive Analysis.

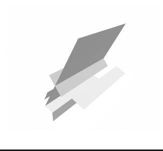
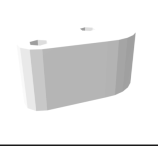
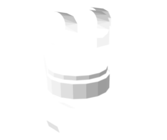
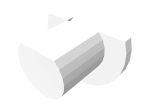
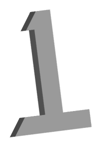
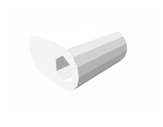
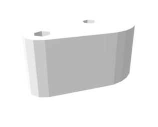
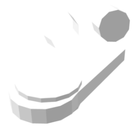
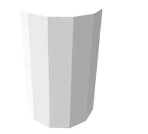
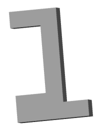
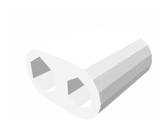
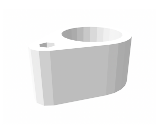
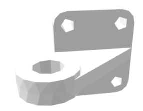
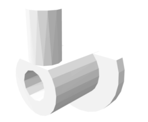
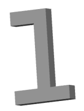
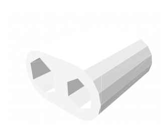
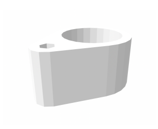
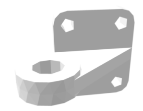
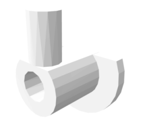
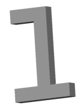
Models	Generated Results				
	00001213	00004252	00005341	00012350	00016271
DeepCAD					
Text2CAD					
TCADGen+CADLLM					
Ground Truth					

Table 9: CAD model effects generated by different methods.

Data	**Command Prediction**			**Param**	**Command-Type Performance**				**CCS**	
Size	ACC	F1	AUC	**ACC**	Line	Arc	Circle	Extrude	Accuracy	Avg LCS$_{\text{ratio}}$
0	0.532	0.769	0.851	0.566	0.939	0.564	0.756	0.675	0.160	0.622
100	0.849	0.818	0.871	0.924	0.928	0.678	0.814	0.788	0.567	0.912
500	0.932	0.907	0.934	0.970	0.931	0.855	0.907	0.900	0.789	0.972
1000	**0.966**	**0.947**	**0.962**	**0.983**	**0.979**	**0.924**	**0.960**	**0.946**	**0.864**	**0.983**

Table 10: Performance analysis across different training data sizes.

Reverse Verification Models / Forward Generation Models	Similarity	GPT-4o-mini	Gemini-1.5-Flash	Google_Gemma-2-27B-it	Google_Gemma-2-9B-it	Meta-Llama-3.1-70B	Meta-Llama-3.1-405B	Qwen2.5-72B
claude-3-5-haiku-20241022	1	0.2	0.16	0.36	0.08	0.24	0.24	0.24
	0.98	0.44	0.36	0.52	0.36	0.52	0.56	0.52
	0.9	0.68	0.64	0.80	0.56	0.84	0.76	0.80
	Average	0.905	0.866	0.932	0.829	0.927	0.938	0.929
claude-3-5-sonnet-20241022	1	0.24	0.24	0.32	0.12	0.32	0.24	0.24
	0.98	0.52	0.76	0.84	0.48	0.68	0.80	0.80
	0.9	0.76	0.84	0.88	0.68	0.80	0.84	0.88
	Average	0.896	0.939	0.942	0.841	0.937	0.932	0.943
ERNIE-4.0-8K-Latest	1	0.385	0.346	0.346	0.038	0.346	0.385	0.346
	0.98	0.577	0.615	0.538	0.346	0.577	0.615	0.615
	0.9	0.769	0.846	0.731	0.731	0.846	0.808	0.846
	Average	0.9	0.918	0.915	0.877	0.929	0.935	0.933
ERNIE-Speed-8K	1	0.141	0.141	0.051	0.0	0.141	0.15	0.141
	0.98	0.224	0.229	0.103	0.206	0.235	0.245	0.235
	0.9	0.418	0.424	0.462	0.412	0.412	0.449	0.418
	Average	0.661	0.669	0.672	0.632	0.67	0.684	0.669
gemini-1.5-flash	1	0.373	0.363	0.43	0.13	0.397	0.404	0.417
	0.98	0.573	0.633	0.717	0.53	0.627	0.68	0.65
	0.9	0.817	0.807	0.883	0.763	0.843	0.859	0.853
	Average	0.929	0.937	0.95	0.905	0.939	0.945	0.944
gemini-1.5-pro	1	0.333	0.45	0.487	0.067	0.453	0.429	0.4
	0.98	0.56	0.743	0.807	0.472	0.71	0.751	0.687
	0.9	0.867	0.883	0.947	0.829	0.887	0.87	0.92
	Average	0.942	0.956	0.973	0.928	0.955	0.953	0.964
gemma-2-27b-it	1	0.217	0.19	0.253	0.027	0.27	0.282	0.2
	0.98	0.413	0.463	0.563	0.387	0.487	0.533	0.43
	0.9	0.787	0.787	0.877	0.78	0.82	0.861	0.823
	Average	0.912	0.923	0.947	0.9	0.929	0.948	0.932
gemma-2-9b-it	1	0.397	0.387	0.407	0.053	0.4	0.41	0.415
	0.98	0.647	0.667	0.677	0.62	0.687	0.68	0.682
	0.9	0.78	0.8	0.797	0.773	0.787	0.782	0.793
	Average	0.902	0.91	0.914	0.898	0.908	0.911	0.909
gpt-4o	1	0.435	0.421	0.462	0.1	0.472	0.43	0.442
	0.98	0.736	0.773	0.773	0.589	0.783	0.795	0.757
	0.9	0.87	0.916	0.9	0.86	0.9	0.93	0.899
	Average	0.951	0.964	0.962	0.939	0.963	0.971	0.957
gpt-4o-mini	1	0.453	0.403	0.44	0.123	0.463	0.452	0.477
	0.98	0.677	0.697	0.71	0.667	0.723	0.756	0.707
	0.9	0.813	0.837	0.857	0.82	0.867	0.87	0.86
	Average	0.933	0.946	0.949	0.932	0.944	0.954	0.946
Llama-3.1-405B-Instruct	1	0.457	0.496	0.558	0.068	0.568	0.517	0.565
	0.98	0.662	0.766	0.856	0.522	0.827	0.779	0.809
	0.9	0.856	0.874	0.939	0.802	0.928	0.858	0.921
	Average	0.943	0.967	0.976	0.923	0.968	0.959	0.971
Llama-3.1-70B-Instruct	1	0.534	0.605	0.611	0.071	0.639	0.605	0.642
	0.98	0.726	0.838	0.841	0.541	0.845	0.855	0.831
	0.9	0.902	0.943	0.946	0.841	0.912	0.924	0.936
	Average	0.963	0.979	0.981	0.941	0.972	0.969	0.978
Qwen2.5-72B-Instruct-Turbo	1	0.25	0.31	0.323	0.017	0.363	0.283	0.29
	0.98	0.45	0.59	0.613	0.33	0.637	0.595	0.573
	0.9	0.78	0.823	0.817	0.687	0.85	0.848	0.81
	Average	0.912	0.936	0.942	0.883	0.947	0.946	0.935

Table 11: Proportions of results generated by the forward generation model with $\mathrm{LCS}_{\mathrm{ratio}}$ values of 1, 0.98, and 0.9, as well as the average $\mathrm{LCS}_{\mathrm{ratio}}$ for the backward validation model, which verifies the generated CCS descriptions.

Forward Model	**Best Reverse Model**	**Proportion of LCS Greater than 0.9**
gemini-1.5-pro	gemma-2-27b-it	0.947
Llama-3.1-70B-Instruct-Turbo	gemma-2-27b-it	0.946
Llama-3.1-405B-Instruct-Turbo	gemma-2-27b-it	0.939
gpt-4o	Llama-3.1-405B-Instruct	0.930
ERNIE-3.5-8K	gemini-1.5-flash	0.906
gemini-1.5-flash	gemma-2-27b-it	0.883
claude-3-5-sonnet-20241022	gemma-2-27b-it	0.880
gemma-2-27b-it	gemma-2-27b-it	0.877
gpt-4o-mini	Llama-3.1-405B-Instruct	0.870
Qwen2.5-72B-Instruct-Turbo	Llama-3.1-70B-Instruct	0.850
ERNIE-4.0-8K-Latest	gemini-1.5-flash	0.846
claude-3-5-haiku-20241022	Llama-3.1-70B-Instruct	0.840
ERNIE-4.0-Turbo-8K	gemma-2-27b-it	0.827
gemma-2-9b-it	gemini-1.5-flash	0.800
ERNIE-Speed-8K	gemma-2-27b-it	0.462

Table 12: Proportion of LCS Greater than 0.9 for Different Forward and Reverse Models in the Two-Phase Evaluation Framework.

Table 13: Prompts Used in Experiments

Process Name	**Prompt**
Multi-view analysis	You are an experienced CAD engineer tasked with providing natural language descriptions of design objects based on images. Your goal is to create clear, specific descriptions for junior designers to understand and model, focusing on shape features, structural elements, and spatial relationships. You will be given an image. Your task is to analyze this description and generate four concise sentences that describe: 1. What the 3D model looks like 2. What it is composed of 3. Its appearance 4. What it can do or its purpose Follow these guidelines when generating your description: • Focus on the most important and distinctive features of the object • Use clear and specific language to describe shapes, structures, and spatial relationships • Avoid using phrases like "this multi-view image shows" or "distinct views" or explaining the image itself • Do not include any explanations or additional commentary • Do not describe the color, metallic sheen, or use words like "blue", "shadow", "transparent", "metal", "plastic", "image", "black", "grey", "CAD model", "abstract", "orange", "purple", "golden", "smooth", or "green" Present your four sentences in the following format: • Sentence describing what the 3D model looks like • Sentence describing what it is composed of • Sentence describing its appearance • Sentence describing what it can do or its purpose Ensure that each sentence is concise, clear, and adheres to the abovementioned guidelines.

Continued on next page

Table 13 – continued from previous page

Process Name	Prompt
Point cloud analysis	You will be analyzing a 3D point cloud to describe its geometric structure. Your task is to generate four precise sentences, each focusing on a different aspect of the object represented by the point cloud: 1. Overall Shape 2. Components 3. Structural Details 4. Function For each aspect, follow these specific guidelines: • **Overall Shape:** Describe the basic geometric form and approximate dimensions. Focus on the general shape and proportions. • **Components:** List the main structural elements and their spatial relationships. Identify distinct parts and how they connect or relate to each other. • **Structural Details:** Describe key geometric features, patterns, and surface characteristics. Focus on form rather than color or texture. • **Function:** State the most likely practical purpose based purely on the form and structure. Make an educated guess about the object's intended use. Important guidelines: • Be specific and detailed in your descriptions. • Avoid any mention of colors, materials, or textures that cannot be definitively determined from point cloud data alone. • Focus solely on geometry, form, and function. • Use precise language and, where appropriate, include approximate measurements or proportions. Keep descriptions focused on geometry, form, and function. Avoid any references to color, material, or texture that cannot be definitively determined from point cloud data alone. Write your analysis in four separate sentences, one for each aspect. Do not label or number the sentences. Ensure that each sentence flows naturally into the next.

Continued on next page

Table 13 – continued from previous page

Process Name	Prompt
Appearance description generation and verification	You are a parameter in analyzing and describing 3D models based on point cloud data and multiple perspective descriptions. Your task is to analyze the compatibility between two descriptions and provide a synthesized description when they share fundamental geometric characteristics. You will be given two input descriptions: • `{POINT_CLOUD_DESCRIPTION}` • `{MULTIPLE_PERSPECTIVES_DESCRIPTION}` Analyze these descriptions according to the following criteria: 1. **Core Analysis Approach:** • Extract basic geometric forms (shapes, volumes, structures) • Look for ANY potential geometric compatibility • Focus on finding similarities rather than differences • Consider different ways of describing the same geometric concept • Default to finding compatibility unless clearly contradictory 2. **Ignore Non-Essential Elements:** • Color and material properties • Surface textures and finishes • Intended purpose or function • Aesthetic qualities • Subjective interpretations

Continued on next page

Table 13 – continued from previous page

Process Name	Prompt
Parameter description generation	You are a parameter in CAD command descriptions. Your task is to describe a provided CAD command sequence based on specific requirements. Follow these instructions carefully: First, familiarize yourself with these key terms and commands: • **⟨SOL⟩:** Denotes the start of a 2D closed curve. All commands prior to the Extrude command belong to this 2D curve. • **L (Line):** x, y—Coordinates of the line's end-point, defining direction and length in the plane. • **A (Arc):** x, y—Coordinates of the arc's end-point; α—Sweep angle, indicating arc curvature; f—Counter-clockwise flag, specifying arc direction. • **R (Circle):** x, y—Center coordinates of the circle; r—Radius, specifying circle size. • **E (Extrude):** θ, ϕ, γ—Orientation of the sketch plane, defining its rotation and direction; px, py, pz—Origin of the sketch plane in 3D space; s—Scale factor, adjusting profile size; e1, e2—Extrude distances; b—Boolean operation type; u—Extrude direction. Now, describe this CAD command sequence following these guidelines: 1. Only output command descriptions—no additional content or explanations. 2. Include each command, its parameters, and reflect the order of execution in your description. 3. Describe how the Extrude command and its parameters transform the 2D curve into a 3D model. 4. For the Extrude command, fully output the parameters b and u without changing them. Parameter b can be "NewBodyFeatureOperation", "JoinFeatureOperation", "CutFeatureOperation", or "IntersectFeatureOperation". Parameter u can be "OneSideFeatureExtentType", "SymmetricFeatureExtentType", or "TwoSidesFThe parameterentType". 5. Do not change any parameter values in your description. 6. Form a single, cohesive paragraph of CAD modeling guidance. Describe the sequence in natural language, including every parameter without omission. Here is the CAD command sequence you need to describe: • `{CAD_COMMAND_SEQUENCE}` Your output should be a flowing, descriptive paragraph that guides the reader through the CAD modeling process, detailing each step and parameter in the order they appear in the command sequence. Do not use bullet points or numbered lists. Ensure your description is comprehensive, covering all aspects of the provided sequence.

Continued on next page

Table 13 – continued from previous page

Process Name	**Prompt**
Parameter description verification	You are tasked with converting a CAD command description into a precise CAD operation sequence. Follow these instructions carefully: 1. **Read the description carefully** and identify the CAD operations mentioned. These may include creating lines, arcs, circles, and performing extrude operations. 2. **Convert each identified operation into the corresponding CAD command format** as follows: • To start a new sketch: – `<SOL>` • For a line: – `<Line>`: x=<xValue>, y=<yValue> • For an arc: – `<Arc>`: x=<xValue>, y=<yValue>, α=<alphaValue>, f=<fValue> • For a circle: – `<Circle>`: x=<xValue>, y=<yValue>, r=<radiusValue> • For an extrude operation: – `<Extrude>`: θ=<thetaValue>, φ=<phiValue>, γ=<gammaValue>, px=<pxValue>, py=<pyValue>, pz=<pzValue>, s=<sValue>, e1=<e1Value>, e2=<e2Value>, b=<extrudeOperation>, u=<extentType> 3. **Arrange the converted operations in the sequence they appear in the description**. Start each new sketch with `<SOL>`. 4. After converting all operations, end the CAD operation sequence with: • `<EOS>` 5. Ensure that you only output the CAD operation sequence without any additional explanations or comments. 6. If any information is missing or unclear in the description, use reasonable default values or omit the parameter. 7. Remember: • There should be only one `<SOL>` tag per sketch. • `<EOS>` marks the end of the entire CAD operation sequence. • Do not include any text or explanations outside of the specified command formats. 8. You will be given a CAD command description in the following format: • `{CAD_COMMAND_DESCRIPTION}` Output the resulting CAD operation sequence exactly as specified, with no additional commentary.

Continued on next page

Table 13 – continued from previous page

Process Name	Prompt
Reflection on parameter description issues	You are a CAD parametric description analysis parameter. Your task is to analyze the differences between a generated CAD Command Sequence (CCS) and a real CCS, identify problems in the CAD parametric description, and provide suggestions for improvement. 1. CAD parametric description: • `{CAD_DESCRIPTION}` 2. Generated CCS based on the CAD description: • `{GENERATED_CCS}` **1. Differences between generated and real CCS:** • Missing commands in the generated CCS: [List any specific commands present in the real CCS but missing in the generated one.] • Extra commands in the generated CCS: [List any commands that appear in the generated CCS but are not in the real CCS.] • Differences in command parameters or order: [List and describe any differences in parameters or the order of commands.] **2. Problems in the CAD parametric description:** • Missing information: [Discuss any information that is not clearly stated in the CAD parametric description, which may have led to missing or t commands.] • Incorrect specifications: [Identify any specifications that are inaccurate or potentially lead to errors in the generated CCS.] • inaccurate specifications [Highlight any parts of the CAD description that are ambiguous and could be interpreted in multiple ways. **3. Improvement suggestions:** • Clarify any ambiguous descriptions by providing more specific and detailed parameters. • Ensure that all necessary information for each command is provided to avoid missing commands or incomplete sequences. • Double-check the order of commands in the CAD description to match the expected sequence for correct execution. • Review the parameter specifications to ensure they are accurate and precise, avoiding potential discrepancies.

Continued on next page

Table 13 – continued from previous page

Process Name	**Prompt**
Parameter Description Regeneration	You are a CAD parametric description modification parameter. Your task is to analyze and modify a given CAD parametric description to make it more accurate, complete, and unambiguous based on a real CCS sequence and a provided opinion. Follow these steps: 1. **Review the provided CAD parametric description:** • `{CAD_DESCRIPTION}` 2. **Examine the real CCS sequence:** • `{REAL_CCS}` 3. **Consider the provided opinion:** • `{OPINION}` 4. **Familiarize yourself with these key terms and commands in the CCS:** • `⟨SOL⟩`: Start of a 2D closed curve • `L (Line)`: x, y - Coordinates of the line's end-point • `A (Arc)`: x, y - Coordinates of the arc's end-point; α - Sweep angle; f - Counter-clockwise flag • `R (Circle)`: x, y - Center coordinates; r - Radius • `E (Extrude)`: θ, ϕ, γ - Orientation of the sketch plane; p_x, p_y, p_z - Origin of the sketch plane; s - Scale factor; e_1, e_2 - Extrude distances; b - Boolean operation type; u - Extrude direction 5. **Pay special attention to the Extrude command parameters:** • `Parameter` b: "NewBodyFeatureOperation", "JoinFeatureOperation", "CutFeatureOperation", or "IntersectFeatureOperation" • `Parameter` u: "OneSideFeatureExtentType", "SymmetricFeatureExtentType", or "TwoSidesFeatureExtentType" 6. **Analyze the differences between the CAD description and the real CCS.** 7. **Based on your analysis, modify the CAD parametric description to address any issues found.** 8. **Output CAD parametric description ONLY!!!**

Continued on next page

Table 13 – continued from previous page

Process Name	Prompt
LLMCAD Enhanced CCS	You are a CAD sequence generation parameter. Your task is to generate a correct CAD Command Sequence (CCS) based on a CAD description, existing CCS, and confidence levels. Follow these instructions carefully: Now, consider the following CAD description: `{CAD_DESCRIPTION}` Here is the existing CCS and its associated confidence levels: `{EXISTING_CCS}` `{CONFIDENCE}` Analyze the confidence levels provided. Pay special attention to parameters with confidence levels below 0.98, as these should be the focus of your modifications. Based on the CAD description and the confidence levels, generate a new CCS. Modify the existing CCS to align with the description, focusing on adjusting parameters with low confidence levels. Ensure that the new CCS accurately represents the described CAD operations. Output the new CCS without any explanations or additional comments.