

CAD-Judge: Toward Efficient Morphological Grading and Verification for Text-to-CAD Generation

Zheyuan Zhou^{1*}, Jiayi Han^{2*}, Liang Du³, Naiyu Fang⁴, Lemiao Qiu¹, Shuyou Zhang¹

¹ Zhejiang University

² Inspur Genersoft Co. Ltd., Inspur Group Co. Ltd.

³ Tencent Inc.

⁴ Nanyang Technological University

<https://zhouzheyuan.github.io/cad-judge>

Abstract

Computer-Aided Design (CAD) models are widely used across industrial design, simulation, and manufacturing processes. Text-to-CAD systems aim to generate editable, general-purpose CAD models from textual descriptions, significantly reducing the complexity and entry barrier associated with traditional CAD workflows. However, rendering CAD models can be slow, and deploying VLMs to review CAD models can be expensive and may introduce reward hacking that degrades the systems. To address these challenges, we propose CAD-Judge, a novel, verifiable reward system for efficient and effective CAD preference grading and grammatical validation. We adopt the Compiler-as-a-Judge Module (CJM) as a fast, direct reward signal, optimizing model alignment by maximizing generative utility through prospect theory. To further improve the robustness of Text-to-CAD in the testing phase, we introduce a simple yet effective agentic CAD generation approach and adopt the Compiler-as-a-Review Module (CRM), which efficiently verifies the generated CAD models, enabling the system to refine them accordingly. Extensive experiments on challenging CAD datasets demonstrate that our method achieves state-of-the-art performance while maintaining superior efficiency.

Introduction

Computer-Aided Design (CAD) has become an essential tool in modern industrial design and manufacturing, enabling the precise modeling of complex geometries (Robertson and Allen 1993; Cherng et al. 1998; Yamamoto and Nakakoji 2005; Deng, Chen, and Olechowski 2024; Khan et al. 2024a). Traditional workflows involve manually creating 2D sketches using basic primitives such as lines, arcs, and circles, which are then extruded into 3D solid models. While this process is powerful, it demands significant expertise and iterative refinement, often limiting accessibility for non-expert users.

To democratize CAD modeling, recent efforts have focused on text-to-CAD systems—approaches that aim to automate the generation of parametric command sequences directly from natural language descriptions (Khan et al. 2024b; Li et al. 2024; Wang et al. 2025a). Such systems allow

both professionals and hobbyists to rapidly prototype editable CAD models through intuitive text instructions, significantly reducing the time and effort required for manual construction. Current approaches typically rely on end-to-end Transformer-based architectures trained on paired datasets of textual commands and corresponding CAD sequences (Li et al. 2024; Khan et al. 2024b). These models are often trained from scratch, resulting in slow convergence and suboptimal performance. Moreover, limited model capacity restricts their ability to interpret nuanced or complex design instructions. Recent works attempt to address this by integrating pre-trained large language models (LLMs), leveraging their strong language understanding capabilities and general knowledge of CAD principles (Wang et al. 2025a).

However, aligning these models with human preferences remains a major hurdle. As shown in Figure 1, most existing methods depend on pairwise preference data—comparisons between preferred and rejected CAD outputs generated from the same prompt (Wang et al. 2025a). Constructing such data is costly, requiring rendering and ranking multiple generations per prompt via VLMs. Additionally, LLM-generated CAD sequences often fail to conform to strict syntactic rules, leading to high invalidation rates during compilation.

To address these limitations, we introduce **CAD-Judge**, a novel verifiable reward system for efficient and effective CAD preference grading and grammatical validation. Unlike conventional methods that rely on Vision-Language Models (VLMs) and their dependence on rendered images, our approach leverages per-sample binary feedback obtained via a Compiler-as-a-Judge Module (CJM). Specifically, our CJM employs the Chamfer Distance (CD) between generated and ground-truth CAD models as a fast, interpretable reward signal—obviating the need for computationally costly rendering and ranking procedures. Furthermore, to improve the robustness of LLM-generated sequences, we introduce the Compiler-as-a-Review Module (CRM), which efficiently verifies the generated CAD models, allowing the LLMs to modify them. By monitoring compilation outcomes and performing intelligent adjustments based on error diagnostics, this scaling strategy significantly boosts the likelihood of generating valid, executable CAD programs without incur-

*These authors contributed equally.

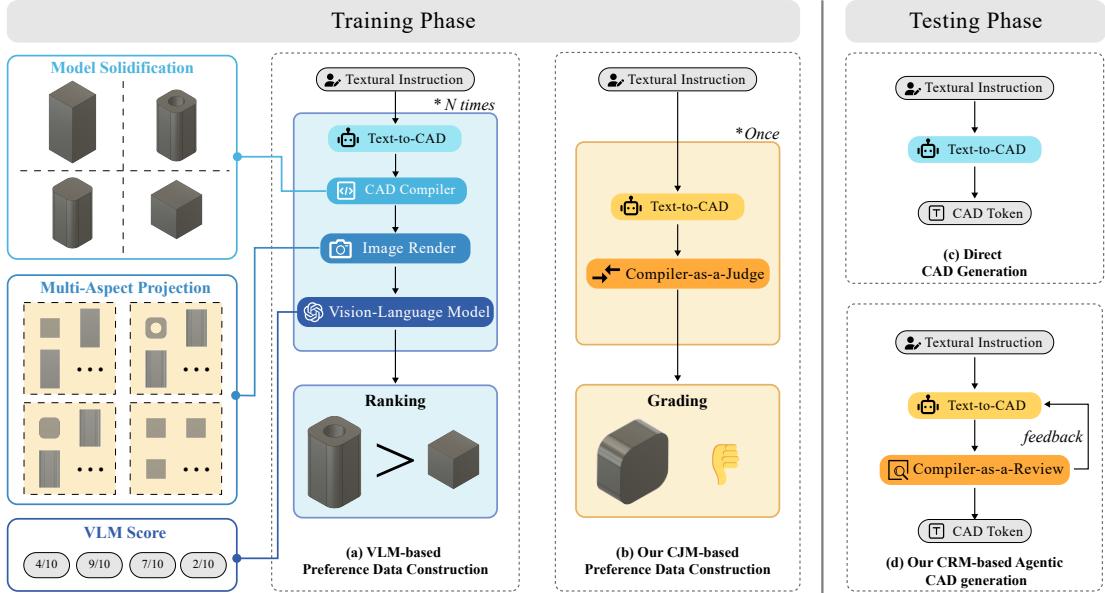


Figure 1: Comparison between existing preference data construction pipelines and ours. (a) Existing Vision-Language Model (VLM)-based methods rely on costly multi-sample rendering and projection for score grading. (b) Our method leverages per-sample preferences derived from a native quantifiable CAD compiler, enabling efficient and scalable data collection. The “>” symbol denotes pairwise preferences, where one item is selected as superior from a ranked list, while the thumbs-down icon indicates independent binary negative preferences without explicit pairing. (c) Existing direct CAD sequence generation methods output the final CAD sequence directly through LLMs. (d) Our approach incorporates feedback on erroneous samples, enabling the model to correct obvious mistakes and iteratively generate the final CAD sequence.

ring substantial computational overhead.

Our key contributions are summarized as follows:

- An efficient Compiler-as-a-Judge Module (CJM) for constructing per-sample binary preference data using CJM-derived rewards.
- A paradigm grounded in prospect theory that directly optimizes the alignment of generated CAD command sequences by maximizing generative utility.
- A Compiler-as-a-Review Module (CRM) that efficiently reviews the LLM-generated CAD sequences during test time for robust CAD generation.
- Extensive experimental evaluations on various datasets demonstrating that CAD-Judge achieves state-of-the-art performance on text-to-CAD tasks while maintaining high computational efficiency.

Related Workers

CAD Generation. CAD, depending on its representation methods, can be categorized into constructive solid geometry (CSG) (Du et al. 2018; Kania, Zieba, and Kajdanowicz 2020; Yu et al. 2022, 2023), boundary representation (B-rep) (Jayaraman et al. 2022; Wang, Zheng, and Zhou 2022; Xu et al. 2024), and sketch-and-extrude modeling (SEM) (Willis et al. 2021a; Wu, Xiao, and Zheng 2021). Our research primarily focuses on SEM, which records the drawing history of a CAD model, starting from curves, progressing to 2D sketches, and ultimately culminating in the extruding

tion to form a 3D model. DeepCAD (Wu, Xiao, and Zheng 2021) was the pioneer in proposing a sketch-extrusion construction sequence representation for CAD models. In a preliminary experiment, it predicted the CAD history from latent vectors or point clouds. CAD Translator (Li et al. 2024) utilize an encoder-decoder architecture. By aligning the encoded CAD Embedding with the Text Embedding, the model acquires the ability to generate CAD sequences from text. After the fusion process, the embeddings are fed into the decoder to recover 3D parametric CAD sequences. Text2CAD (Khan et al. 2024b) trains an end-to-end transformer-based autoregressive network to generate parametric CAD models from input texts. (Yavartanoo et al. 2024) employs images as an intermediate medium and realizes the generation in multiple steps. Specifically, the text is transformed into an isometric image that represents the described features. Subsequently, this image is mapped into orthographic technical drawings, which serve as the basis for generating the 3D CAD model. CAD-GPT (Wang et al. 2025b) introduces a Multimodal Large Language Model (MLLM) that can precisely synthesize CAD modeling sequences from either a single image or a textual description. The CAD-MLLM has the capability to generate parametric CAD models based on multimodal inputs, such as text, images, point clouds, or a combination of these modalities. CAD-Coder (Guan et al. 2025) fine-tunes LLMs to generate CadQuery scripts for building CAD models by incorporating GRPO and CoT techniques. CADmium (Govindarajan et al.

2025) introduces a novel annotation pipeline that leverages GPT-4.1 to process up to 10 multi-view images of 3D objects along with their construction sequences. CADFusion (Wang et al. 2025a) leverages 2D renderings generated by generative models and utilizes the image knowledge from a Vision-Language Model (VLM) as a reward signal to construct a DPO dataset, improving the performance of the model after SFT through an iterative training procedure. In contrast, our method does not rely on DPO, and therefore does not require pairwise preference annotations. By only requiring binary preferences, our approach enables the rapid and efficient construction of high-quality alignment data without the need for LVMs.

Large Language Models (LLMs). LLMs have achieved remarkable success in recent years, primarily through the pretraining process on extensive general-purpose datasets. To adapt LLMs to specific tasks, supervised fine-tuning (SFT) has emerged as a powerful approach. By training LLMs on task-specific labeled data, SFT can significantly enhance the models' performance on targeted tasks (Chen et al. 2024; Yue et al. 2024; Han et al. 2025; Liu et al. 2025a). Reinforcement Learning (RL) is another crucial technique to align LLM outputs with human preferences. PPO (Schulman et al. 2017) aims to iteratively update the model's policy to maximize cumulative rewards based on human feedback or predefined evaluation metrics. DPO (Rafailov et al. 2023) simplifies the RL process by directly optimizing the model's output based on pairwise human preferences, reducing computational complexity while maintaining effectiveness. Recently, KTO (Ethayarajh et al. 2024) directly maximizes the utility of generations instead of maximizing the log-likelihood of preferences, which only requires a binary signal of whether an output is desirable or undesirable for an input. Many recent works propose using the rule-based approach to identify the samples to be chosen or rejected to avoid reward hacking. For example, DeepSeek-R1 proposes GPRO (DeepSeek-AI et al. 2025), which leverages rule-based reward, could significantly enhance the reasoning capacity of LLMs. Visual-RFT (Liu et al. 2025b) also proposes to utilize rule-based metrics (e.g., F1-score, IoU) to sample preferred reasoning trajectories to enhance multi-model perception tasks.

Methodology

Overview

Let a textual description be denoted as x , and a CAD parametric sequence as y . The text-to-CAD task can be formulated as first encoding the description text x into a CAD parametric representation y using a function $f(\cdot)$. The generated CAD parametric sequence y can then be rendered into a 3D solid object via a CAD compiler $c(\cdot)$, such as `pythonOCC` (Paviot 2022). The overall objective is for the rendered 3D model $m = c(f(x))$, to closely match the user's intended 3D object.

Our CAD-Judge introduces a two-stage training paradigm that integrates supervision with performance feedback for text-to-CAD generation. In the first stage, a pre-trained LLM is fine-tuned using text-CAD sequence pairs as supervised

signals, enabling the model to learn the structural patterns and compositional logic of CAD sequences while adapting its general language capabilities to the domain-specific syntax of CAD instructions. The second stage incorporates compiler-mediated feedback, where the LLM is incentivized to generate executable and geometrically accurate sequences by rewarding successful compilations and penalizing errors or deviations from ground-truth models. This dual-objective approach bridges symbolic reasoning with functional execution constraints, enhancing both syntactic correctness and geometric fidelity.

Agentic CAD Generation

Despite their strong language understanding capabilities, LLMs often produce outputs that exhibit syntactic or structural inconsistencies when applied to domain-specific tasks such as text-to-CAD generation. Prior work has shown that LLM-generated CAD sequences suffer from a high invalidation rate due to deviations from the expected parametric command structure (Li et al. 2024; Wang et al. 2025a). These irregularities prevent successful compilation into executable CAD models, limiting the practical usability of such systems. To mitigate this issue, we propose a simple yet effective agentic CAD generation approach. There are two nodes in this agent: the generation node and the review node. The generation node is implemented with a well-trained LLM (specifically, the model with our two-stage fine-tuning with verifiable CAD reward), and the review node is implemented with CRM. During inference, the generation node first generates the CAD sequence according to the input prompt. The review node verifies the validation of the generated sequence by CRM, and expands the review information to the input prompt for further refinement. We iterate this generate-review phase until the sequence is valid or exceeds the iteration limit.

Compiler-as-a-Judge Module (CJM) for Verifiable CAD Reward

The core objective of Text-to-CAD task is to ensure that 3D CAD models accurately align with the design intent described in text while strictly adhering to the rules and requirements defined by human experts. In practical implementation, however, rendering multi-view images of CAD models for validation purposes tends to be computationally expensive and slow, significantly impeding workflow efficiency. To address these limitations and facilitate effective learning from design failures, we propose the CJM by leveraging local compilers instead of relying on remote vision-language models (VLMs), which substantially expedite the validation process.

Specifically, as illustrated in Figure 2, both the predicted sequence \hat{y} and ground-truth CAD sequence y , represented as sketch-and-extrude (SE) parametric sequences, are first reconstructed into solid models. These solids are then converted to boundary representation (BRep) models, subsequently tessellated into meshes, and finally sampled uniformly to generate point clouds. The Chamfer Distance (CD) is computed between the predicted point cloud \hat{P} (derived

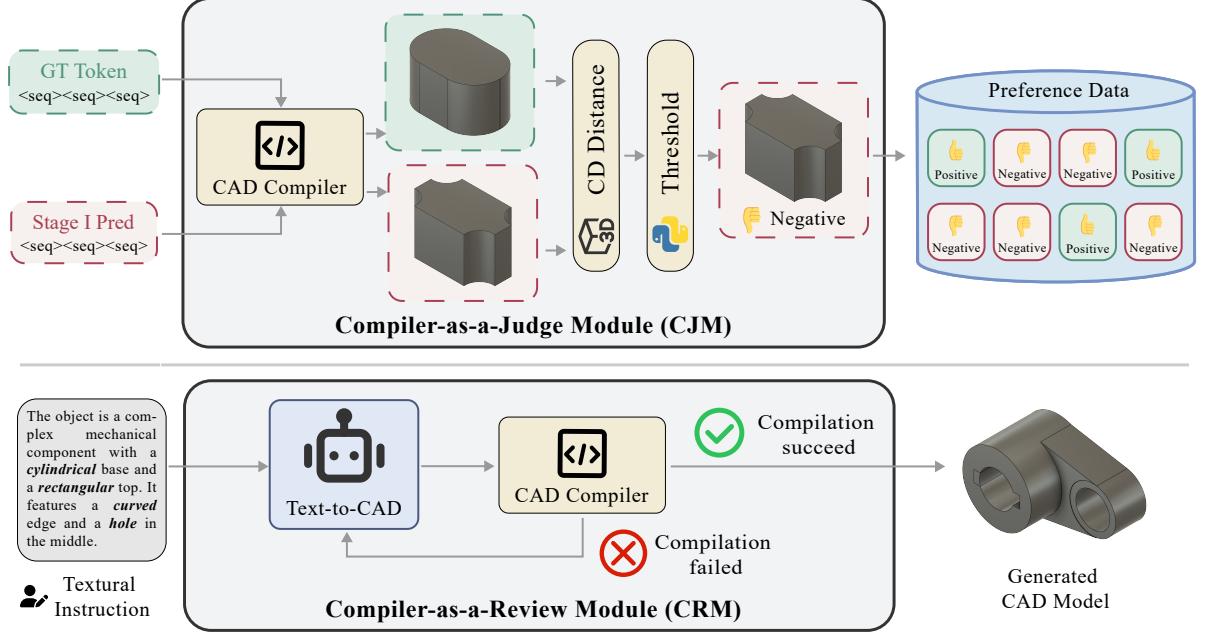


Figure 2: Overview of our CAD-Judge framework. Our **Compiler-as-a-Judge Module (CJM)** functions as a rule-based reward signal for grading after the Stage I SFT training. Predictions exceeding the Chamfer Distance (CD) threshold are treated as negative samples, while those with CD below the threshold undergo are selected with as positive samples. During inference, our **Compiler-as-a-Review Module (CRM)** validates whether the generated CAD sequence can be successfully compiled. If compilation fails, the model resamples until a valid sequence is produced, thereby significantly reducing the invalid generation ratio.

from \hat{y}) and the ground-truth point cloud P (derived from y) as:

$$CD(\hat{P}, P) = \frac{1}{|\hat{P}|} \sum_{\hat{p} \in \hat{P}} \min_{p \in P} \|\hat{p} - p\|_2^2 + \frac{1}{|P|} \sum_{p \in P} \min_{\hat{p} \in \hat{P}} \|p - \hat{p}\|_2^2, \quad (1)$$

where $|\hat{P}|$ and $|P|$ denote the number of points in \hat{P} and P respectively, and $\|\cdot\|_2$ represents the Euclidean distance. For samples that compile successfully and have CD below the threshold, we construct training tuples: with probability α , we form $(x, \hat{y}, \text{true})$ to represent desired samples. Samples failing to compile at any stage or with CD above a predefined threshold are marked as rejected that are with probability $1 - \alpha$, we form (x, y, false) to represent undesired samples.

By incorporating predicted sequences rejected in the first stage as ‘‘rejection samples’’, LLMs leverage such preference-derived data to rectify errors and reduce inaccuracies. Our approach constructs preference data by harnessing preference-based reward signals that are directly quantifiable through explicit rules from a CAD compiler, thereby alleviating the costs associated with the traditional rendering and ranking pipeline with VLM. Furthermore, leveraging local compilation tools and focusing on binary preferences enables efficient and scalable alignment.

Compiler-as-a-Review Module (CRM) for Agentic CAD Generation

Different from the CJM, the ground truth sequence is not accessible during the test phase. Therefore, CRM simply parses the sequence into the CAD model. The error information of the parsing would be utilized as the review of the sequence for self-refinement.

Specifically, after the model generates a sequence \hat{y} from an input textual description, the output is immediately fed into a CAD compiler for validation. This validation process may identify various errors that cause compilation failures, including invalid sequence formatting (e.g., missing `<end>` terminators), geometric inconsistencies in sketch construction such as unclosable loops, incorrect extrusion parameters, and violations of boolean operation rules. Upon detecting an invalid sequence, the model performs re-sampling, combining the original input prompt x with the error information derived from \hat{y} , leveraging its stochastic decoding mechanism to generate a new candidate sequence. The high efficiency of the local CAD compiler and compact tokenization of CAD commands ensure each validation step introduces negligible computational overhead, enabling our method to maintain real-time performance.

Two Stage Fine-tuning for CAD Generation

Supervised Fine-Tuning for CAD Parametric Sequence Generation. We use a pre-trained LLM as the foundation of

our method, aiming to leverage its accurate understanding of various language prompts and its knowledge of the 3D CAD design process (Makatura et al. 2023). We use the textual description introduced in Text2CAD (Khan et al. 2024b), and concatenate it with a fixed instruction to form the complete text input as x to the LLM. We use the previously mentioned SE construction hierarchies as our textural CAD sequence y , which not only reduces the output difficulty and length for the LLM but also makes full use of the LLM’s existing structured output capabilities.

Our method fine-tunes the pre-trained LLM by minimizing the discrepancy between the generated parametric sequence \hat{y} and the ground-truth parametric sequence y using the cross-entropy loss function, denoted as \mathcal{L}_{SFT} :

$$\mathcal{L}_{\text{sft}} = -\mathbb{E}_{(x,y) \sim D_{SFT}} \left[\frac{1}{T} \sum_{t=1}^T \log p(\hat{y}_t = y_t | x) \right], \quad (2)$$

where T is the length of the sequence, and $p(\cdot)$ is the predicted probability of the t -th token. By minimizing this loss function, we can make the generated parametric sequence closer to the ground-truth parametric sequence, thus improving the accuracy and reliability of the model in generating CAD parametric sequences.

Preference Alignment with Binary Preference.

Text-to-CAD generation can be inherently framed as an absolute preference task rather than a relative comparison task. These relative approaches often struggle to rank flawed outputs—particularly when such rankings cannot be directly derived from 3D models. Additionally, from a decision-theoretic perspective (Tversky and Kahneman 1992), the expected utility hypothesis posits that rational agents evaluate actions based on their expected outcomes. In this context, the binary nature of rewards also enables models to learn from negative and positive samples with differentiated weights more effectively.

To align large language models with preferences, the framework introduces an implied reward defined as:

$$r_\theta(x, y) = \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}, \quad (3)$$

with π_θ being the trainable model with parameters θ , and π_{ref} serving as the reference model. A reference point z_0 is defined as the expected Kullback-Leibler divergence between the current model and reference model distribution:

$$z_0 = \mathbb{E}_x [\text{KL}(\pi_\theta(\hat{y}|x) || \pi_{\text{ref}}(\hat{y}|x))], \quad (4)$$

serving as a divergence penalty to constrain distribution shift.

The value function is constructed as:

$$v(x, y) = \begin{cases} \lambda_D \sigma(\beta(r_\theta(x, y) - z_0)) & \text{if } y \sim y_{\text{desirable}} | x, \\ \lambda_U \sigma(\beta(z_0 - r_\theta(x, y))) & \text{if } y \sim y_{\text{undesirable}} | x, \end{cases} \quad (5)$$

where λ_D and λ_U are weights assigned to desirable and undesirable samples respectively, $\sigma(\cdot)$ denotes the sigmoid function approximating the utility function, and β is a scaling factor. The KTO framework formulates the overall optimization objective as:

$$L_{\text{KTO}} = \mathbb{E}_{x, y \sim D_{\text{KTO}}} [v(x, y)], \quad (6)$$

where $v(x, y)$ represents the value function.

Experiment

Setups

Datasets. We utilize the widely adopted DeepCAD dataset (Wu, Xiao, and Zheng 2021) as a source of CAD parametric sequences, with text prompts incorporated from Text2CAD (Khan et al. 2024b). The DeepCAD dataset is randomly partitioned into training, validation, and test sets at a ratio of 90%–5%–5%, resulting in a total of 137k samples aligned with the previous method. The CADPrompt (Alrashedy et al. 2025) comprises expert-written natural language instructions for 200 samples from DeepCAD dataset, serving as a crucial test for out-of-distribution generalization. Additionally, we incorporate the more challenging Fusion360 Reconstruction dataset (Willis et al. 2021b), paired with corresponding text descriptions derived from the CADmium dataset (Govindarajan et al. 2025). The Fusion360 dataset provides an official 80:20 split, consisting of 6,900 and 1,725 designs for training and testing, respectively. All sketches and final CAD models are normalized to ensure consistency across samples. The parametric sequences are represented using 8-bit numerical precision, following prior work (Wu, Xiao, and Zheng 2021; Khan et al. 2024b; Wang et al. 2025a), to facilitate stable and efficient training.

Implementation Details. Our experiments are conducted using PyTorch on a single NVIDIA RTX 4090 GPU. We apply Low-Rank Adaptation (LoRA) (Hu et al. 2022) to enable efficient fine-tuning while maintaining performance. During SFT, the model is trained for 3 epochs with a batch size of 1, gradient accumulation over 8 steps, a learning rate of 1.0×10^{-4} , and a cosine scheduler with 10% warm-up. In the second phase, outputs generated on the validation set are compared with ground-truth sequences to construct a binary preference dataset, with training performed at a reduced learning rate of 5.0×10^{-6} for stability.

Metrics. Our evaluation focuses on the fidelity and alignment of generated CAD models with respect to input textual instructions. Following Text2CAD (Khan et al. 2024b), we report three metrics: F1 score, Chamfer Distance (CD), and Invalidity Ratio (IR). The F1 score assesses correspondence between generated and ground-truth parametric sequences, computed separately for geometric primitives (lines, arcs, circles) and extrusion operations, then averaged. CD measures geometric similarity between point clouds of generated and reference 3D models, reflecting shape quality. Invalidity Ratio quantifies the fraction of syntactically invalid or non-executable outputs, indicating generation reliability.

Main Results

Quantitative Evaluation. The results in Table 1 demonstrate the effectiveness of our proposed method, CAD-Judge, in generating CAD command sequences across different text prompts, compared to state-of-the-art approaches.

DeepCAD shows basic semantic understanding by identifying primitives and extrusions but lacks precise parameterization, resulting in poor geometric alignment (Huang et al. 2023). Text2CAD improves both semantic and geometric accuracy, reflecting better integration of language and shape understanding. CADFusion excels in primitive classification

Table 1: Quantitative test results on the DeepCAD dataset. The results include F1 scores for primitives and extrusions as well as mean and median Chamfer Distance (CD) and Invalidity Ratio (IR). CD is multiplied by 10^3 .

Prompt	Model	F1↑				CD↓		IR↓
		Line	Arc	Circle	Extrusion	Median	Mean	
L3	DeepCAD (Wu, Xiao, and Zheng 2021)	0.77	0.20	0.65	0.89	32.82	97.93	10.00
	Text2CAD (Khan et al. 2024b)	0.81	0.35	0.74	0.93	0.37	29.27	2.38
	CADFusion (Wang et al. 2025a)	0.79	0.43	0.69	0.92	-	30.23	-
	CAD-Coder (Guan et al. 2025)	-	-	-	-	0.17	6.54	1.45
	CAD-Judge (Ours)	0.99	0.96	0.99	1.00	0.15	4.66	1.38
L2	Text2CAD (Khan et al. 2024b)	0.73	0.07	0.66	0.93	74.1	150.01	1.84
	CADFusion (Wang et al. 2025a)	0.67	0.05	0.48	0.94	-	146.15	-
	CAD-Judge (Ours)	0.89	0.49	0.94	1.00	0.62	52.55	1.54

Table 2: Quantitative generalization results on the CADPrompt dataset and Fusion360 dataset.

Dataset	Model	F1↑				CD↓		IR↓
		Line	Arc	Circle	Extrusion	Median	Mean	
CADPrompt	Text2CAD (Khan et al. 2024b)	0.67	0.00	0.31	0.84	127.70	-	1.57
	CADMium (Govindarajan et al. 2025)	0.69	0.22	0.61	0.89	116.75	-	6.28
	CAD-Judge (Ours)	0.70	0.15	0.65	0.90	42.69	152.40	2.55
Fusion360	Text2CAD (Khan et al. 2024b)	0.51	0.14	0.60	0.87	157.06	239.51	3.57
	CADMium (Govindarajan et al. 2025)	0.84	0.71	0.90	0.98	76.05	-	12.58
	CAD-Judge (Ours)	0.81	0.61	0.91	0.99	72.83	177.45	8.16

through pre-trained language models but struggles with accurate parameter control, revealing a gap between recognition and execution. CAD-Coder improves model reasoning by introducing a CoT planning process, demonstrating strong performance on the CD metric. CAD-Judge achieves a better balance between semantic understanding and parametric precision. While it does not reach the lowest Invalidity Ratio (IR), this is largely due to inherent LLM output uncertainty. The CAD sequence structure extraction relies on closed-loop geometries sensitive to small errors, a problem worsened by strict CAD compiler constraints. Nonetheless, CAD-Judge significantly reduces IR compared to other LLM-based approaches such as CADFusion, while preserving high fidelity and validity.

In Table 2, we evaluate the generalization capability of the model trained on DeepCAD across different datasets. It can be observed that our method significantly outperforms CADmium in both the CD metric and the IR metric. The slight underperformance in the “arc” may stem from the fact that when we utilize rule-based preference as the optimization signal, complex arcs become a “high-cost option”. This incentivizes the model to adopt a shortcut strategy by preferentially using “lines” and “circles” to minimize CD value.

Qualitative Evaluation. Figure 3 compares the results among the ground-truth, our method, and Text2CAD on the test set. Text2CAD generate well-formed shapes without irregular edges or corners. However, it often produces oversimplified shapes and, for more complex prompts, tends to generate multiple cubes or panels instead of accurately cap-

turing the intended structure. Our CAD-Judge provides the most precise response to input instructions and achieves the highest similarity to the ground truth. It successfully captures complex shapes, including rectangles, hexagons, and nested structures, such as a hexagonal hole within a cylinder. Additionally, it exhibits a strong understanding of language cues, accurately interpreting numerical and qualitative descriptors like “long” or “T-shape”.

Ablation Studies

Main Components. To dissect the contributions of our framework’s core components, we perform an ablation study focusing on the proposed two-stage fine-tuning and agentic CAD generation, as shown in Table 3. By comparing Experiment 1 and Experiment 2, we observe that the introduction of two-stage fine-tuning leads to significant improvement in our model, primarily reflected in the CD metric. This outcome reveals that the two-stage preference dataset constructed through the CJM enhances the accuracy of parameters in the parameterized sequence, such as element positioning and extrusion size parameters. When comparing Experiment 1 and Experiment 3, it is evident that our proposed agentic CAD generation mechanism significantly reduces the model’s invalid ratio. Furthermore, as this mechanism eliminates erroneous sequences, it ensures that the regenerated sequences are more parameter-consistent with real-world scenarios. Experiment 4 serves as our default setting. By combining the two judgment-based modules, it improves the accuracy of sequence parameters and reduces the

Table 3: Ablation studies on the proposed main components. Default settings are marked in gray.

Exp	Two Stage		Agentic	F1↑				CD↓		IR↓
	Fine-tuning	CAD Generation		Line	Arc	Circle	Extrusion	Median	Mean	
1	-	-		0.89	0.46	0.93	0.99	0.91	56.60	3.07
2	✓	-		0.89	0.47	0.93	1.00	0.68	52.96	3.23
3	-	✓		0.89	0.47	0.93	1.00	0.74	55.12	2.52
4	✓	✓		0.89	0.49	0.94	1.00	0.62	52.55	1.54

Table 4: Ablation studies between preference data construction pipelines. Default settings are marked in gray.

Type	F1↑				CD↓		IR↓
	Line	Arc	Circle	Extrusion	Median	Mean	
VLM-based Paired Preference	0.76	0.42	0.76	1.00	18.52	122.87	6.02
Rule-based Paired Preference (Ours)	0.87	0.48	0.91	1.00	2.29	69.58	2.76
Rule-based Binary Preference (Ours)	0.89	0.49	0.94	1.00	0.62	52.55	1.54

Table 5: Ablation study on the numbers of iterations. Default settings are marked in gray.

Iterations	CD↓		IR↓
	Median	Mean	
0	0.63	53.11	2.62
1	0.62	52.55	1.54
2	0.61	51.71	1.45
3	0.61	51.72	1.38

invalid ratio in sequence generation.

Preference Data Construction Comparison. We compared model performance across different preference data construction methods. The results in Table 4 show that both rule-based construction methods outperform the VLM-based method. The VLM-based ranking uses GPT-4o to evaluate the consistency between rendered CAD images and text prompts for pairwise rankings. Rule-based ranking leverages CD metrics as reward signals to rank sequences, while our rule-based grading uses binary labels to indicate whether sequences meet the criteria. Rule-based ranking outperforms VLM-based methods in F1 (sequence element accuracy) and CD (model similarity) metrics, demonstrating the value of compiler-derived signals. Further gains from binary grading arise from reduced cognitive load and more consistent, noise-robust signals without data.

Iteration Limit of Agentic CAD Generation. We compare the iteration limit of agentic CAD generation. The results are summarized in Table 5. Increasing the number of iterations steadily decreases the Invalidity Ratio (IR). However, the CD remains stable across different configurations. This observation suggests that while test-time rollback can reduce IR through compiler optimizations, increasing iterations does not lead to significant further improvements. We argue that excessive rollbacks are unnecessary. Therefore, in our implementation, we perform a single round of iteration.

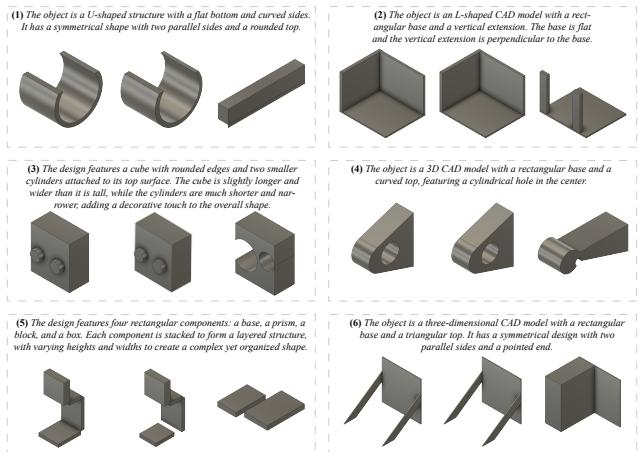


Figure 3: Qualitative comparison of generated results. For each subsection, the input prompt is displayed at the top. From left to right, we show the ground truth, our CAD-Judge, and Text2CAD.

Conclusion

We propose CAD-Judge, a novel alignment framework for text-to-CAD generation that rethinks the fine-tuning of LLMs in structured command synthesis. Our CAD-Judge leverages per-sample binary feedback through a Compiler-as-a-Judge Module, enabling efficient and interpretable training. We further introduce a Compiler-as-a-Review Module that monitors compilation outcomes and performs intelligent corrections upon detecting syntactic or geometric errors during the test phase. Our extensive experiments demonstrate that CAD-Judge outperforms existing approaches in both effectiveness and efficiency. Future work will explore reward-shaping strategies that incorporate semantic-level feedback beyond geometric fidelity, aiming to further enhance generalization and robustness.

References

- Alrashedy, K.; Tambwekar, P.; Zaidi, Z.; Langwasser, M.; Xu, W.; and Gombolay, M. 2025. Generating CAD Code with Vision-Language Models for 3D Designs.
- Chen, J.; Cai, Z.; Ji, K.; Wang, X.; Liu, W.; Wang, R.; Hou, J.; and Wang, B. 2024. HuatuoGPT-o1, Towards Medical Complex Reasoning with LLMs. arXiv:2412.18925.
- Cherng, J. G.; Shao, X.-Y.; Chen, Y.; and Sferro, P. R. 1998. Feature-based part modeling and process planning for rapid response manufacturing. *Computers & industrial engineering*, 34(2): 515–530.
- DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; Zhang, X.; Yu, X.; Wu, Y.; Wu, Z. F.; Gou, Z.; Shao, Z.; Li, Z.; Gao, Z.; Liu, A.; Xue, B.; Wang, B.; Wu, B.; Feng, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; Dai, D.; Chen, D.; Ji, D.; Li, E.; Lin, F.; Dai, F.; Luo, F.; Hao, G.; Chen, G.; Li, G.; Zhang, H.; Bao, H.; Xu, H.; Wang, H.; Ding, H.; Xin, H.; Gao, H.; Qu, H.; Li, H.; Guo, J.; Li, J.; Wang, J.; Chen, J.; Yuan, J.; Qiu, J.; Li, J.; Cai, J. L.; Ni, J.; Liang, J.; Chen, J.; Dong, K.; Hu, K.; Gao, K.; Guan, K.; Huang, K.; Yu, K.; Wang, L.; Zhang, L.; Zhao, L.; Wang, L.; Zhang, L.; Xu, L.; Xia, L.; Zhang, M.; Zhang, M.; Tang, M.; Li, M.; Wang, M.; Li, M.; Tian, N.; Huang, P.; Zhang, P.; Wang, Q.; Chen, Q.; Du, Q.; Ge, R.; Zhang, R.; Pan, R.; Wang, R.; Chen, R. J.; Jin, R. L.; Chen, R.; Lu, S.; Zhou, S.; Chen, S.; Ye, S.; Wang, S.; Yu, S.; Zhou, S.; Pan, S.; Li, S. S.; Zhou, S.; Wu, S.; Ye, S.; Yun, T.; Pei, T.; Sun, T.; Wang, T.; Zeng, W.; Zhao, W.; Liu, W.; Liang, W.; Gao, W.; Yu, W.; Zhang, W.; Xiao, W. L.; An, W.; Liu, X.; Wang, X.; Chen, X.; Nie, X.; Cheng, X.; Liu, X.; Xie, X.; Liu, X.; Yang, X.; Li, X.; Su, X.; Lin, X.; Li, X. Q.; Jin, X.; Shen, X.; Chen, X.; Sun, X.; Wang, X.; Song, X.; Zhou, X.; Wang, X.; Shan, X.; Li, Y. K.; Wang, Y. Q.; Wei, Y. X.; Zhang, Y.; Xu, Y.; Li, Y.; Zhao, Y.; Sun, Y.; Wang, Y.; Yu, Y.; Zhang, Y.; Shi, Y.; Xiong, Y.; He, Y.; Piao, Y.; Wang, Y.; Tan, Y.; Ma, Y.; Liu, Y.; Guo, Y.; Ou, Y.; Wang, Y.; Gong, Y.; Zou, Y.; He, Y.; Xiong, Y.; Luo, Y.; You, Y.; Liu, Y.; Zhou, Y.; Zhu, Y. X.; Xu, Y.; Huang, Y.; Li, Y.; Zheng, Y.; Zhu, Y.; Ma, Y.; Tang, Y.; Zha, Y.; Yan, Y.; Ren, Z. Z.; Ren, Z.; Sha, Z.; Fu, Z.; Xu, Z.; Xie, Z.; Zhang, Z.; Hao, Z.; Ma, Z.; Yan, Z.; Wu, Z.; Gu, Z.; Zhu, Z.; Liu, Z.; Li, Z.; Xie, Z.; Song, Z.; Pan, Z.; Huang, Z.; Xu, Z.; Zhang, Z.; and Zhang, Z. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.
- Deng, Y.; Chen, J.; and Olechowski, A. 2024. What sets proficient and expert users apart? Results of a Computer-Aided Design experiment. *Journal of Mechanical Design*, 146(1).
- Du, T.; Inala, J. P.; Pu, Y.; Spielberg, A.; Schulz, A.; Rus, D.; Solar-Lezama, A.; and Matusik, W. 2018. Inversecsg: Automatic conversion of 3d models to csg trees. *ACM Transactions on Graphics (TOG)*, 37(6): 1–16.
- Ethayarajh, K.; Xu, W.; Muennighoff, N.; Jurafsky, D.; and Kiela, D. 2024. Kto: Model alignment as prospect theoretic optimization. In *ICML*.
- Govindarajan, P.; Baldelli, D.; Pathak, J.; Fournier, Q.; and Chandar, S. 2025. CADmium: Fine-Tuning Code Language Models for Text-Driven Sequential CAD Design. *arXiv preprint arXiv:2507.09792*.
- Guan, Y.; Wang, X.; Ming, X.; Zhang, J.; Xu, D.; and Yu, Q. 2025. CAD-Coder: Text-to-CAD Generation with Chain-of-Thought and Geometric Reward. *arXiv preprint arXiv:2505.19713*.
- Han, J.; Du, L.; Du, H.; Zhou, X.; Wu, Y.; Zhang, Y.; Zheng, W.; and Han, D. 2025. SLIM: Let LLM Learn More and Forget Less with Soft LoRA and Identity Mixture. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.
- Huang, Y.; Song, J.; Wang, Z.; Zhao, S.; Chen, H.; Juefei-Xu, F.; and Ma, L. 2023. Look before you leap: An exploratory study of uncertainty measurement for large language models. *arXiv preprint arXiv:2307.10236*.
- Jayaraman, P. K.; Lambourne, J. G.; Desai, N.; Willis, K. D.; Sanghi, A.; and Morris, N. J. 2022. Solidgen: An autoregressive model for direct b-rep synthesis. *arXiv preprint arXiv:2203.13944*.
- Kania, K.; Zieba, M.; and Kajdanowicz, T. 2020. UCSG-NET-unsupervised discovering of constructive solid geometry tree. *NeurIPS*, 33: 8776–8786.
- Khan, M. S.; Dupont, E.; Ali, S. A.; Cherenkova, K.; Kacem, A.; and Aouada, D. 2024a. Cad-signet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In *CVPR*, 4713–4722.
- Khan, M. S.; Sinha, S.; Uddin, S. T.; Stricker, D.; Ali, S. A.; and Afzal, M. Z. 2024b. Text2CAD: Generating Sequential CAD Designs from Beginner-to-Expert Level Text Prompts. In *NeurIPS*.
- Li, X.; Song, Y.; Lou, Y.; and Zhou, X. 2024. CAD Translator: An Effective Drive for Text to 3D Parametric Computer-Aided Design Generative Modeling. In *ACM MM*.
- Liu, Z.; Guo, X.; Lou, F.; Zeng, L.; Niu, J.; Wang, Z.; Xu, J.; Cai, W.; Yang, Z.; Zhao, X.; Li, C.; Xu, S.; Chen, D.; Chen, Y.; Bai, Z.; and Zhang, L. 2025a. Fin-R1: A Large Language Model for Financial Reasoning through Reinforcement Learning. arXiv:2503.16252.
- Liu, Z.; Sun, Z.; Zang, Y.; Dong, X.; Cao, Y.; Duan, H.; Lin, D.; and Wang, J. 2025b. Visual-RFT: Visual Reinforcement Fine-Tuning. arXiv:2503.01785.
- Makatura, L.; Foshey, M.; Wang, B.; HähnLein, F.; Ma, P.; Deng, B.; Tjandrasuwita, M.; Spielberg, A.; Owens, C. E.; Chen, P. Y.; et al. 2023. How can large language models help humans in design and manufacturing? *arXiv preprint arXiv:2307.14377*.
- Paviot, T. 2022. pythonocc. Zenodo.
- Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *NeurIPS*.

- Robertson, D.; and Allen, T. J. 1993. CAD system use and engineering performance. *IEEE Transactions on Engineering Management*, 40(3): 274–282.
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Tversky, A.; and Kahneman, D. 1992. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*, 5: 297–323.
- Wang, K.; Zheng, J.; and Zhou, Z. 2022. Neural face identification in a 2d wireframe projection of a manifold object. In *CVPR*, 1622–1631.
- Wang, R.; Yuan, Y.; Sun, S.; and Bian, J. 2025a. Text-to-CAD Generation Through Infusing Visual Feedback in Large Language Models. In *ICML*.
- Wang, S.; Chen, C.; Le, X.; Xu, Q.; Xu, L.; Zhang, Y.; and Yang, J. 2025b. CAD-GPT: Synthesising CAD Construction Sequence with Spatial Reasoning-Enhanced Multimodal LLMs. In *AAAI*.
- Willis, K. D.; Jayaraman, P. K.; Lambourne, J. G.; Chu, H.; and Pu, Y. 2021a. Engineering sketch generation for computer-aided design. In *CVPR*, 2105–2114.
- Willis, K. D. D.; Pu, Y.; Luo, J.; Chu, H.; Du, T.; Lambourne, J. G.; Solar-Lezama, A.; and Matusik, W. 2021b. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences. *ACM Transactions on Graphics (TOG)*, 40(4).
- Wu, R.; Xiao, C.; and Zheng, C. 2021. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In *ICCV*.
- Xu, X.; Lambourne, J.; Jayaraman, P.; Wang, Z.; Willis, K.; and Furukawa, Y. 2024. BrepGen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4): 1–14.
- Yamamoto, Y.; and Nakakoji, K. 2005. Interaction design of tools for fostering creativity in the early stages of information design. *International Journal of Human-Computer Studies*, 63(4-5): 513–535.
- Yavartanoo, M.; Hong, S.; Neshatavar, R.; and Lee, K. M. 2024. Text2CAD: Text to 3D CAD Generation via Technical Drawings. *arXiv preprint arXiv:2411.06206*.
- Yu, F.; Chen, Q.; Tanveer, M.; Mahdavi Amiri, A.; and Zhang, H. 2023. D $\hat{\rightarrow}$ CSG: Unsupervised learning of compact CSG trees with dual complements and dropouts. *NeurIPS*, 36: 22807–22819.
- Yu, F.; Chen, Z.; Li, M.; Sanghi, A.; Shayani, H.; Mahdavi-Amiri, A.; and Zhang, H. 2022. Capri-net: Learning compact cad shapes with adaptive primitive assembly. In *CVPR*, 11768–11778.
- Yue, S.; Liu, S.; Zhou, Y.; Shen, C.; Wang, S.; Xiao, Y.; Li, B.; Song, Y.; Shen, X.; Chen, W.; et al. 2024. LawLLM: Intelligent Legal System with Legal Reasoning and Verifiable Retrieval. In *International Conference on Database Systems for Advanced Applications*, 304–321. Springer.