CAD-HLLM: Generating Executable CAD from Text with Hierarchical LLM Planning

Zhuo Zuo PETER_Z@STU.SCU.EDU.CN
Yantao Gan GANYANTAO@STU.SCU.EDU.CN
Junfeng Long PEPPER@STU.SCU.EDU.CN
Xianggen Liu LIUXIANGGEN@SCU.EDU.CN
College of Computer Science, Sichuan University, Chengdu 610065, China

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Translating natural language into precise and executable Computer-Aided Design (CAD) programs remains a challenging task, requiring both semantic understanding and geometric fidelity. In this paper, we present CAD-HLLM, a hierarchical LLM framework for structured CAD command generation. Our approach decomposes the task into two stages: a *Plan Generator* that infers high-level symbolic plans from text, and a *Parameter Completor* that generates detailed parametric commands conditioned on both the original description and the inferred plan. To enhance robustness, we introduce a lightweight ensemble selection mechanism that ranks and selects among multiple candidates based on model log-likelihoods. Experiments on benchmark datasets show that our method outperforms existing baselines in both parametric precision and 3D shape similarity, demonstrating the effectiveness of hierarchical reasoning and LLM-based planning in bridging the gap between human design intent and executable CAD sequences.

Keywords: Text-to-CAD; Hierarchical Generation; Large Language Models; 3D Design Automation

1. Introduction

Computer-Aided Design (CAD) plays a critical role in modern engineering, manufacturing, and creative industries (Cherng et al., 1998). While contemporary CAD systems enable precise and flexible modeling, the process of creating detailed models still demands significant expertise. In practice, designers must translate high-level design intentions into complex, low-level modeling operations (Zou and Luo, 2025). This translation process contributes to a steep learning curve and hinders the scalability of CAD tools for rapid prototyping and design automation (Camba et al., 2016).

Recent advances in Large Language Models (LLMs) have shown promise in program synthesis (Li et al., 2022; Gao et al., 2023), multimodal reasoning (Li et al., 2023b), and structured output generation (Zhang et al.). In particular, text-to-code and text-to-3D tasks demonstrate that LLMs can translate abstract language into interpretable and executable outputs (Li et al., 2024; Austin et al., 2021), offering a promising path toward natural language-driven CAD modeling.

However, generating executable CAD programs from natural language is particularly challenging because outputs must be both syntactically valid and geometrically feasible.

Key difficulties are: (1) small parametric errors can make shapes invalid; (2) symbolic commands must align with physically consistent geometry; and (3) tightly coupled parameters mean a single inconsistency can invalidate the whole sequence. These constraints make CAD generation more fragile and high-dimensional than general code generation tasks.

To address this, we propose **CAD-HLLM**, a hierarchical LLM-based framework that decomposes the generation process into two stages. First, the *Plan Generator* predicts a coarse symbolic modeling plan that outlines high-level geometric operations. Then, the *Parameter Completer* fills in precise numerical parameters conditioned on both the original prompt and the predicted plan. This coarse-to-fine design mirrors how human designers conceptualize structures before specifying technical details (Han and Lyu, 2025; Li et al., 2023a), and enables the model to better align linguistic intent with geometric execution. To improve robustness, we incorporate an ensemble scoring mechanism that ranks multiple LLM generations using log-likelihoods (Wei et al., 2022; Yao et al., 2023), enhancing consistency without additional supervision. We evaluate our framework on two benchmark datasets that differ in linguistic style and modeling complexity, demonstrating superior accuracy and generalization over existing baselines.

To summarize, our key contributions are:

- We propose CAD-HLLM, a hierarchical framework for text-to-CAD generation, which formulates the task as structured program synthesis and leverages a modular pipeline with a Plan Generator and a Parameter Completer.
- We introduce a lightweight ensemble scoring mechanism that improves output robustness by selecting among multiple LLM generations without additional training or supervision.
- We conduct comprehensive experiments showing that CAD-HLLM outperforms strong baselines in accuracy and generalization, highlighting the potential of large language models to bridge natural language and executable CAD modeling.

2. Related Work

2.1. Text-to-CAD Generation

Translating natural language into executable CAD programs lies at the intersection of computer-aided design, natural language understanding, and program synthesis (Ellis et al., 2019; Nye et al., 2021). Unlike general text-to-3D generation tasks (Poole et al., 2022), text-to-CAD requires structured, parametric commands that precisely satisfy geometric and syntactic constraints (Wu et al., 2021; Khan et al., 2024b). Early approaches leveraged symbolic grammars and domain-specific languages (DSLs) to produce coarse-grained shape representations from text (Talton et al., 2012; Jones et al., 2020).

Recent efforts have adopted neural models, particularly Transformer-based architectures, to learn mappings from language to CAD programs (Chen et al., 2019; Wu et al., 2021), often incorporating attention mechanisms to improve semantic grounding. Khan et al. (2024b) introduced a hierarchical prompting strategy using pretrained LLMs to bridge high-level design intent and low-level CAD commands, reflecting the hierarchical and multi-granularity nature of human conceptualization.

Despite these advances, semantic ambiguity, limited contextual grounding, and data scarcity remain key challenges. Most available CAD datasets either lack programmatic annotations or offer insufficient linguistic diversity (Lambourne et al., 2022; Wu et al., 2021). Resources like Fusion360Gallery (Willis et al., 2021) provide diverse modeling scenarios but typically rely on semi-automated annotation pipelines to construct paired text-program datasets.

2.2. Structured Output Generation with LLMs

Large language models (LLMs) have achieved strong performance in structured generation tasks such as program synthesis (Li et al., 2022), symbolic reasoning (Welleck et al., 2022), and database querying (Rajkumar et al., 2022b). These tasks demand not only syntactic fidelity, but also long-range coherence and precise numerical reasoning (Zhang et al., 2022). Models like Codex (Rajkumar et al., 2022a) and AlphaCode (Li et al., 2022) have demonstrated near-human coding performance, while grammar-constrained decoding techniques (e.g., SG-SQL (Zhang et al.)) further enhance structured output validity.

Hierarchical reasoning strategies have shown promise in improving output quality. Chain-of-Thought prompting (Wei et al., 2022; Luo et al., 2023) decomposes complex tasks into intermediate steps, which has inspired multi-stage generation pipelines in domains like code synthesis (Wang et al., 2023; Jain et al., 2022) and planning (Han and Lyu, 2025). These approaches enhance interpretability and robustness by decoupling complex tasks into modular generation stages.

While promising, extending these techniques to the CAD domain presents new challenges, including geometric precision, spatial reasoning, and sensitivity to parametric errors (Fan et al., 2022). Unlike code, minor numerical inaccuracies in CAD programs can cause severe semantic failures or invalid geometry, making CAD a particularly challenging domain for structured generation with LLMs.

3. Method

3.1. Problem Definition

The goal of text-to-CAD generation is to translate a natural language description into a sequence of symbolic commands in the Sketch-Extrude language, which can then be executed by CAD toolkits such as OpenCascade(Paviot, 2022) to produce the final 3D model. The Sketch-Extrude language, introduced by DeepCAD (Wu et al., 2021), linearizes CAD modeling procedures into discrete and sequential symbolic operations. Each CAD model consists of one or more 2D sketches, followed by a 3D extrusion. A sketch is composed of one or more closed *loops*, each of which is defined by a series of curve primitives—`Line (L)`, `Arc (A)`, or `Circle (R)`. Each loop begins with a special token `⟨SOL⟩`, and the entire sequence ends with `⟨EOS⟩`. The sketch defines a profile, and the `Extrude (E)` command lifts this profile into a 3D solid. Altogether, the command vocabulary includes six symbolic operations: {`⟨SOL⟩`, `L`, `A`, `R`, `E`, `⟨EOS⟩`}. Each command is associated with a set of geometric or categorical parameters, as summarized in Table 1.

Unlike typical code generation, here the output must not only be syntactically valid but also geometrically feasible. The search space is the joint distribution over symbolic

Command	Associated Parameters
⟨SOL⟩	(none) – start of a closed sketch loop
L (Line)	x, y: endpoint coordinates
A (Arc)	x, y: arc endpoint α: sweep angle f: clockwise flag
R (Circle)	x, y: center position r: radius
E (Extrude)	θ, ϕ, γ: orientation angles p_x, p_y, p_z: origin of sketch plane s: profile scaling factor e_1, e_2: extrusion distances on both sides b: boolean operation type u: extrusion mode (e.g., OneSide, TwoSide)
⟨EOS⟩	(none) – end of full command sequence

Table 1: Modeling commands and parameter semantics used in the Sketch-Extrude language.

operations and continuous parameters, making it high-dimensional and fragile: even a single inconsistent parameter (e.g., radius mismatch, extrusion misalignment) invalidates the entire sequence. This motivates our hierarchical factorization strategy.

Formally, given a natural language input x, the objective is to generate a structured CAD command sequence $y = [y_1, y_2, \ldots, y_N]$. Each command y_k is represented as a tuple $[c_k, p_k]$, where $c_k \in \mathcal{V}$ indicates the type of operation and $p_k = [p_{k,1}, \ldots, p_{k,M_k}]$ denotes the numeric and categorical parameters required by this operation. Here, N is the total number of commands, $\mathcal{V}$ is the vocabulary of available commands defined by the Sketch-Extrude grammar, and M_k specifies the parameter count for the corresponding operation c_k. Thus, the command sequence y fully encodes the procedural and parametric information necessary for unambiguously reconstructing the intended CAD geometry.

3.2. Overview of CAD-HLLM Framework

CAD-HLLM is a hierarchical framework that generates CAD command sequences from natural language by decomposing the task into two semantically distinct but sequentially dependent modules. The first module, the **Plan Generator**, produces a symbolic modeling plan that outlines the high-level structure of the design. The second module, the **Parameter Completer**, grounds this plan into concrete geometric parameters based on both the symbolic structure and the input description. As shown in Figure 1, both modules are implemented using large language models (LLMs) fine-tuned on domain-specific CAD data. Each module generates multiple candidates, from which a hierarchical ensemble mechanism selects the most plausible output based on joint likelihood, aligning with the structure of our layered framework.

The subsequent subsections provide detailed descriptions of each component, including the §3.3 Plan Generator, §3.4 Parameter Completer, and §3.5 ensemble selection strategy.

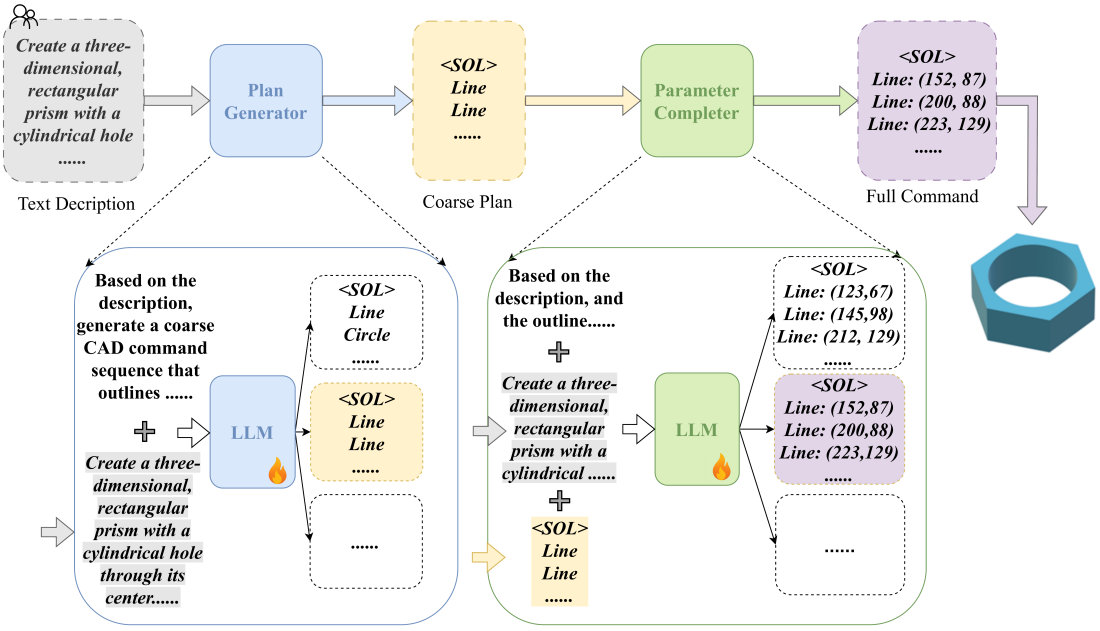

Figure 1: Overview of our CAD-HLLM framework for the generation of CAD command sequences.

3.3. Plan Generator

The first component of the CAD-HLLM framework is the **Plan Generator** (denoted as PlanGen), which maps a natural language input x to a symbolic modeling plan $c = [c_1, c_2, \ldots, c_n]$, where each $c_k \in \mathcal{V}$ denotes a modeling operation token from the predefined vocabulary $\mathcal{V}$ (see Section 3.1). The plan captures the high-level structure of the CAD model—such as which primitives to create and in what order—while omitting all geometric or numeric parameters.

Formally, the Plan Generator is a function:

$$c = \text{PlanGen}(x) \tag{1}$$

where $\text{PlanGen}(\cdot)$ denotes a language model that generates symbolic operation sequences from natural language input.

We implement $\text{PlanGen}(\cdot)$ as a LLM fine-tuned on a corpus of CAD command sequences, where each training sample is converted into a parameter-free version by removing all parameters. The model is trained to predict only the symbolic components c_k extracted from the full command format $y_k = [c_k, p_k]$. It uses the original tokenizer and vocabulary of the pre-trained LLM, and learns to produce valid symbolic plans from supervision alone.

The model is optimized to autoregressively predict symbolic tokens with the standard maximum likelihood objective:

$$\mathcal{L}_1 = -\sum_{k=1}^{n} \log P(c_k \mid x, c_{<k}) \tag{2}$$

At inference time, we apply top-k sampling to produce a diverse set of candidate plans:

$$\{c^{(1)}, c^{(2)}, \ldots, c^{(k_1)}\} \tag{3}$$

Each $c^{(i)}$ represents a distinct interpretation of the input, allowing the Parameter Completer to handle ambiguity.

3.4. Parameter Completer

The second component of the CAD-HLLM framework is the **Parameter Completer** (denoted as ParamComp), which instantiates the symbolic modeling plan $c = [c_1, c_2, \ldots, c_n]$ with concrete geometric parameters to produce the final executable CAD command sequence.

Formally, given a natural language input x and a symbolic plan c produced by the Plan Generator, the goal is to generate a complete parametric sequence:

$$y = \text{ParamComp}(x, c) \tag{4}$$

where $y = [y_1, y_2, \ldots, y_N]$ and each $y_k = [c_k, p_k]$ combines a symbolic operation $c_k \in \mathcal{V}$ with its corresponding parameter vector $p_k = [p_{k,1}, \ldots, p_{k,M_k}]$. The dimensionality M_k varies depending on the operation type c_k, as defined in the Sketch-Extrude grammar (Table 1).

We implement ParamComp$(\cdot, \cdot)$ as another LLM, fine-tuned on fully parameterized CAD command sequences. Each training example consists of a natural language prompt x and its symbolic plan c, concatenated into a structured prompt representation. The model is trained to autoregressively generate the full command sequence y using the standard maximum likelihood objective:

$$\mathcal{L}_2 = -\sum_{k=1}^{N} \log P(y_k \mid x, c, y_{<k}) \tag{5}$$

At inference time, we apply top-k sampling to generate k_2 candidate outputs for each symbolic plan:

$$\{y^{(1)}, y^{(2)}, \ldots, y^{(k_2)}\} \tag{6}$$

These candidates represent alternative numeric realizations of the same symbolic structure and are passed to the ensemble selector (Section 3.5) for final ranking and selection.

3.5. Hierarchical Ensemble Mechanism

To improve robustness and reduce hallucination or syntax errors, we employ a hierarchical ensemble mechanism. During inference, the Plan Generator first samples k_1 symbolic plans:

$$\{c^{(1)}, c^{(2)}, \ldots, c^{(k_1)}\} \tag{7}$$

where each $c^{(i)} = \{c_1^{(i)}, c_2^{(i)}, \ldots, c_{n_i}^{(i)}\}$ is a sequence of symbolic tokens. For each symbolic plan $c^{(i)}$, the Parameter Completer generates k_2 fully parameterized CAD sequences:

$$\{y^{(i,1)}, y^{(i,2)}, \ldots, y^{(i,k_2)}\} \tag{8}$$

where each $y^{(i,j)} = \{y_1^{(i,j)}, y_2^{(i,j)}, \ldots, y_{N_{i,j}}^{(i,j)}\}$ represents a full CAD command sequence.

This results in $k_1 \times k_2$ candidate sequences in total. We compute a composite score for each candidate based on the sum of log-likelihoods from both stages:

$$\text{Score}(y^{(i,j)}) = \log P(c^{(i)} \mid x) + \log P(y^{(i,j)} \mid x, c^{(i)}) \tag{9}$$

The final prediction is selected by maximizing this score:

$$y^* = \arg\max_{i,j} \text{Score}(y^{(i,j)}) \tag{10}$$

Viewed probabilistically, our mechanism can be interpreted as factorizing the joint distribution $p(c, y \mid x)$ into $p(c \mid x)$ and $p(y \mid x, c)$. The Plan Generator and Parameter Completer together define a proposal distribution $q(c, y \mid x)$, while our composite log-likelihood score approximates the target distribution. Selecting the highest-scoring candidate is therefore equivalent to choosing the sample with the largest importance weight p/q, which naturally favors structurally and parametrically consistent generations. To further ensure geometric validity, we filter out candidates with invalid syntax or illegal parameters before scoring, so that only symbolically and geometrically consistent sequences are considered in the final selection.

4. Experiments

Datasets We evaluate CAD-HLLM on two benchmark datasets with distinct annotation styles, allowing us to assess model performance under both descriptive and procedural natural language conditions.

- *Fusion360Gallery* (Willis et al., 2021): Derived from the Reconstruction subset of the Fusion360Gallery dataset, we filter out samples involving unsupported primitives. Natural language annotations are generated via GPT-4o, conditioned on nine rendered views per sample, focusing on the external shape and visual appearance of the final CAD models. We manually inspected the generated prompts to ensure readability and task relevance. The final dataset contains 4,803 training, 847 validation, and 1,412 test samples.

- *Text2CAD* (Khan et al., 2024b): A publicly released dataset based on the DeepCAD corpus (Wu et al., 2021), extended with four levels of natural language instructions via LLM- and VLM-based annotation. It contains approximately 150K training and 8K validation/test samples. All annotations use the expert-level prompt, providing step-by-step procedural guidance aligned with CAD construction sequences.

Fusion360Gallery descriptions emphasize visual and geometric appearance, while Text2CAD captures precise construction steps, offering complementary perspectives for evaluating text-to-CAD generation.

Representative examples of text prompts and rendered CAD models are shown in Figure 2, and full training input–output pairs are included in the prompting template provided in Appendix A.

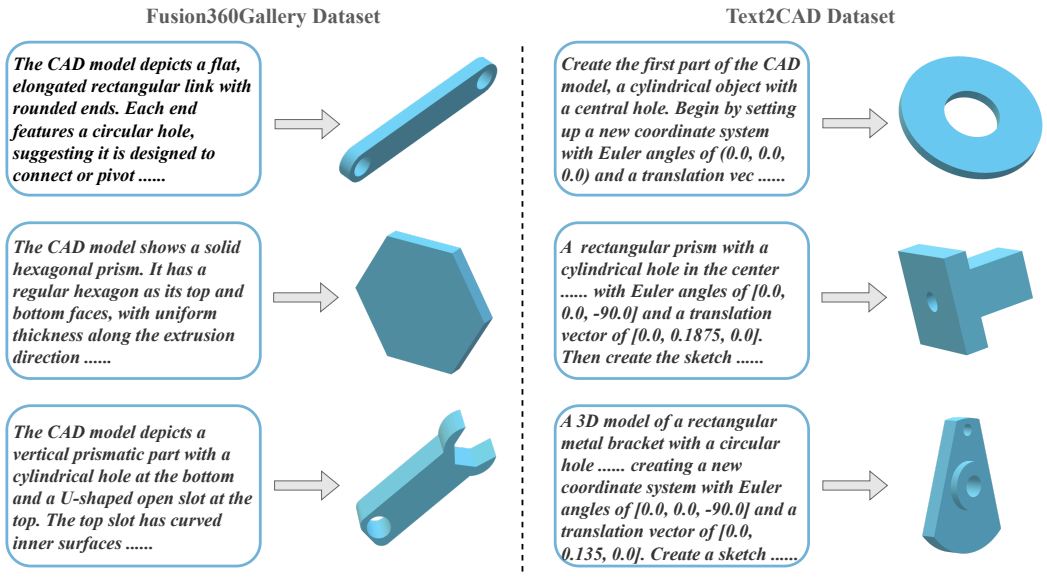

Figure 2: Illustrative examples from the two datasets.

Evaluation Metrics We adopt the standard metrics used in Text2CAD (Khan et al., 2024b), covering both geometric and parametric fidelity. *F1 Score* evaluates the accuracy of both 2D sketch primitives (Line, Arc, Circle) and 3D extrusion commands. Curve-level alignment is performed using the Hungarian algorithm (Kuhn, 1955), while extrusion matching considers direction, extrusion depth, and Boolean operation type jointly (Khan et al., 2024a). *Chamfer Distance (CD, multiplied by* 10^3 *throughout this paper)* quantifies geometric similarity between predicted and reference shapes by computing the average bidirectional distance between surface-sampled point clouds. *Invalid Ratio (IR)* measures the proportion of generated CAD sequences that fail to be parsed or rendered into valid models by the OpenCascade (Paviot, 2022) backend.

Baselines To evaluate the effectiveness of our method, we compare against several representative baseline models, including classical sequence models, transformer-based architectures, and prior state-of-the-art approaches in text-to-CAD generation.

- *LSTM*: A recurrent sequence generation model that encodes the input prompt and autoregressively decodes CAD commands without attention. This baseline captures the inductive bias of sequential modeling, but lacks explicit alignment mechanisms.
- *Transformer*: A standard encoder-decoder Transformer adapted to the text-to-CAD task. The encoder processes the input prompt, and the decoder generates the CAD command sequence with cross-attention at each step.

- *DeepCAD* (Wu et al., 2021): A latent-space CAD generator originally designed for reconstructing sketch-and-extrude sequences from embeddings. Following (Khan et al., 2024b), we extend it for language input by projecting text embeddings into the latent space for decoding.
- *Text2CAD* (Khan et al., 2024b): A state-of-the-art model that integrates pretrained language encoders, latent-space projection, and a decoder-only Transformer with cross-attention to generate full CAD construction sequences from natural language.

Implementation Details CAD-HLLM is implemented using a hierarchical architecture based on Qwen2.5-7B. Both modules—the Plan Generator and Parameter Completer—are fine-tuned using Low-Rank Adaptation (LoRA) via the `unsloth` (Daniel Han and team, 2023) framework for 3 epochs. We use the `adamw_8bit` optimizer with a learning rate of 2×10^{-4}, 5 warmup steps, and linear decay. Training is conducted in bfloat16 with batch size 4. The Plan Generator samples $k_1 = 3$ symbolic plans per input using top-k decoding with temperature 0.5 and nucleus threshold `min_p` $= 0.1$. For each symbolic plan, the Parameter Completer generates $k_2 = 2$ fully parameterized CAD command sequences. We compute joint log-likelihood as $\log P(c \mid x) + \log P(y \mid x, c)$, and select the highest-scoring valid candidate as final output. If the top-ranked sample is invalid, we fall back to the next best valid output.

4.1. Results

Quantitative Results We report quantitative results on both the Fusion360Gallery and Text2CAD datasets in Table 2 and Table 3, respectively. On the Fusion360Gallery benchmark, where natural language descriptions emphasize visual appearance, CAD-HLLM achieves the best overall performance among valid models. It outperforms DeepCAD and Text2CAD on all curve-type F1 scores and achieves the lowest mean and median Chamfer Distance (CD), indicating improved geometric accuracy. For simple primitives such as *Circle*—which require only a single symbolic operation with few parameters—flat sequence models like Transformer may occasionally perform better. However, CAD-HLLM shows clear gains on more compositional primitives (Arc, Extrusion), where hierarchical structure is more beneficial. While Text2CAD attains the lowest Invalid Ratio (IR, 3.74%), it underperforms on curve recognition, particularly for arcs and circles. Overall, CAD-HLLM balances symbolic and numeric fidelity, maintaining a low IR (4.53%) while delivering superior shape quality.

On the Text2CAD benchmark, which provides detailed procedural instructions, CAD-HLLM achieves consistent gains across all metrics. It surpasses all baselines in curve-level F1, extrusion accuracy, and 3D shape similarity (lowest CD), while also producing the fewest invalid sequences (1.02% IR). This highlights that our hierarchical decomposition better exploits the compositional structure of expert-level prompts. Notably, CAD-HLLM also outperforms task-specific baselines such as DeepCAD and Text2CAD, underscoring the advantage of modular reasoning and LLM-driven generation.

To further contextualize our results, we also evaluate a prompting-only in-context learning (ICL) baseline using Qwen2.5-7B without any CAD-specific finetuning. Due to the large scale of Text2CAD, we report these results only on Fusion360Gallery; full details are provided in Appendix A.

Table 2: Quantitative evaluation on the Fusion360Gallery dataset. Metrics in gray correspond to models with Invalid Ratio (IR) above 50%.

Method	F1 ↑				CD ↓		IR ↓
	Line	Arc	Circle	Extrusion	Mean	Median	
LSTM	66.12	18.24	80.86	86.49	59.58	52.19	68.06
Transformer	81.30	57.28	90.88	83.58	91.73	79.13	52.69
DeepCAD	62.90	40.53	83.22	84.65	72.34	49.14	49.29
Text2CAD	53.83	10.51	57.55	84.40	77.69	65.11	**3.74**
CAD-HLLM	**69.32**	**47.92**	80.41	**85.23**	**61.86**	**44.36**	4.53

Table 3: Quantitative evaluation on the Text2CAD dataset.

Method	F1 ↑				CD ↓		IR ↓
	Line	Arc	Circle	Extrusion	Mean	Median	
LSTM	85.23	59.08	90.62	90.34	53.75	47.73	22.44
Transformer	89.07	69.55	92.77	87.27	35.40	17.33	29.67
DeepCAD	88.31	72.81	90.95	91.89	18.92	2.77	10.64
Text2CAD	81.17	35.87	74.56	93.35	14.93	0.36	2.83
CAD-HLLM	**89.94**	**79.00**	**92.70**	**94.15**	**10.51**	**0.28**	**1.02**

Qualitative Analysis Figure 3 presents qualitative comparisons on representative cases from the Fusion360Gallery (left) and Text2CAD (right) datasets. Each row shows the ground-truth CAD model alongside outputs from DeepCAD, Text2CAD, and our CAD-HLLM. Across both datasets, CAD-HLLM generates models that are more faithful to the target geometry, capturing both global structure and local details. On Fusion360Gallery, where the input emphasizes appearance-level descriptions, CAD-HLLM reconstructs plausible shapes with clean geometry, while DeepCAD and Text2CAD often yield distorted or invalid models.

On the more instruction-driven Text2CAD dataset, CAD-HLLM exhibits better adherence to the procedural modeling logic implied by the text, accurately reproducing sketches and extrusion steps. In contrast, baselines struggle with either structural coherence or syntactic validity. These visual results further confirm that the hierarchical structure of CAD-HLLM promotes more interpretable and reliable CAD generation.

4.2. Ablation Study

To better understand the contribution of individual components in CAD-HLLM, we conduct ablation studies on the Text2CAD dataset, which provides fine-grained and procedurally rich annotations well-suited for isolating modeling behaviors. This allows us to analyze the impact of symbolic planning and ensemble scoring under more controlled conditions. As shown in Table 4, removing the Plan Generator (*w/o PlanGen*) leads to increased Chamfer Distance (10.51 → 11.38) and a substantially higher invalid ratio (1.02 → 3.73), demonstrating the importance of explicitly modeling symbolic structure for robust and valid

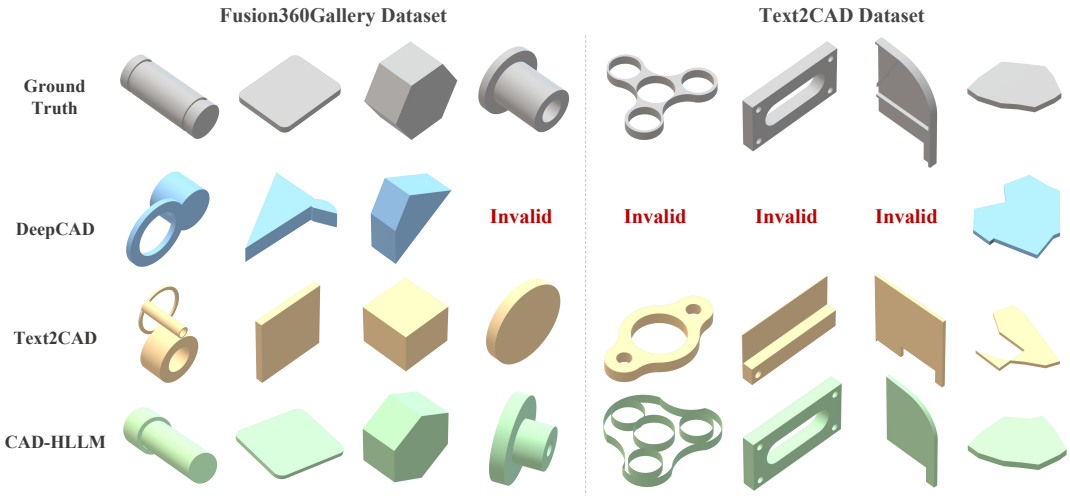

Figure 3: Qualitative comparisons on representative samples from two datasets. Left: Fusion360Gallery; Right: Text2CAD.

generation. Disabling the ensemble module (*w/o Ens*) also slightly degrades performance (e.g., IR increases to 1.65), suggesting that our scoring mechanism improves robustness without extra supervision.

Table 4: Ablation results on the Text2CAD dataset.

Method	F1 ↑				CD ↓		IR ↓
	Line	Arc	Circle	Extrusion	Mean	Median	
w/o PlanGen	88.51	76.65	91.44	93.13	11.38	0.34	3.73
w/o Ens	**90.18**	78.12	92.57	93.97	11.14	0.35	1.65
CAD-HLLM	89.94	**79.00**	**92.70**	**94.15**	**10.51**	**0.28**	**1.02**

To further evaluate symbolic precision, we compare command-type accuracies (`ACC_cmd`) across settings. As shown in Figure 4, CAD-HLLM consistently achieves the highest accuracy across all primitive types, with notable improvements on underrepresented categories such as `Arc` (+1.65) and `Extrusion` (+5.45). These results highlight the complementary benefits of hierarchical decomposition and ensemble selection.

5. Conclusion

We presented CAD-HLLM, a hierarchical framework for text-to-CAD generation that explicitly separates symbolic planning from parametric prediction. By decomposing the generation process into a Plan Generator and a Parameter Completer, our approach achieves

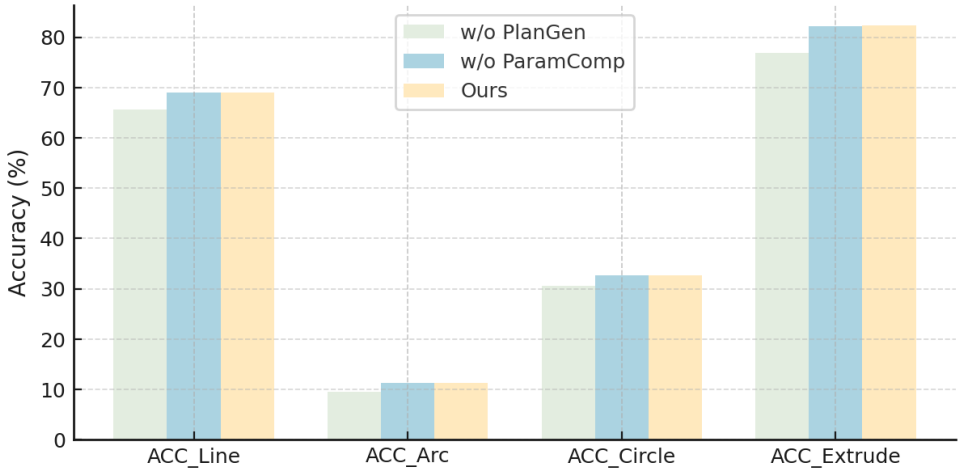

Figure 4: Command-type accuracy across different ablations on Text2CAD dataset.

strong performance on both appearance-based and instruction-driven benchmarks, significantly improving geometric fidelity while reducing invalid generations.

Beyond improved performance, the modular structure of CAD-HLLM enhances interpretability, facilitates debugging, and enables flexible integration with other modeling components. This hierarchical design can serve as a foundation for future text-driven CAD systems that support interactive editing, step-by-step guidance, or constraint-based control.

In future work, we plan to incorporate visual inputs for multimodal grounding and extend the CAD grammar to support a broader range of geometric operations and primitives. We also aim to explore applications of CAD-HLLM in broader 3D modeling scenarios such as inverse engineering, procedural generation, and mixed-initiative design.

Acknowledgements

This work was supported by the Natural Science Foundation of Sichuan Province (2024NSFTD0048), the Science and Technology Major Project of Sichuan Province (2024ZDZX0003), and the National Key R&D Program of China (2024YFB3312503). We also acknowledge the support of Sichuan Province Engineering Technology Research Center of Broadband Electronics Intelligent Manufacturing.

References

Jacob Austin et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Jorge D Camba, Manuel Contero, and Pedro Company. Parametric cad modeling: An analysis of strategies for design reusability. *Computer-aided design*, pages 18–31, 2016.

Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 100–116, 2019.

John G. Cherng, Xin-Yu Shao, Yubao Chen, and Peter R. Sferro. Feature-based part modeling and process planning for rapid response manufacturing. *Computers Industrial Engineering*, pages 515–530, 1998.

Michael Han Daniel Han and Unsloth team. Unsloth, 2023.

Kevin Ellis, Maxwell Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama. Write, execute, assess: Program synthesis with a repl. *Advances in Neural Information Processing Systems*, 2019.

Zhiwen Fan, Tianlong Chen, Peihao Wang, and Zhangyang Wang. Cadtransformer: Panoptic symbol spotting transformer for cad drawings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10986–10996, 2022.

Yujia Gao et al. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2023.

Yewei Han and Chen Lyu. Multi-stage guided code generation for large language models. *Engineering Applications of Artificial Intelligence*, page 109491, 2025.

Naman Jain, Skanda Vaidyanath, Arun Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram Rajamani, and Rahul Sharma. Jigsaw: Large language models meet program synthesis. In *Proceedings of the 44th International Conference on Software Engineering (ICSE)*, pages 1219–1231, 2022.

Ryan Jones, Siddhartha Banerjee, Minhyuk Sung, Paul Guerrero, Niloy J. Mitra, and Li Yi. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. In *ACM Transactions on Graphics (TOG)*, pages 1–13, 2020.

Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-signet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4713–4722, 2024a.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential cad models from beginner-to-expert level text prompts. *arXiv preprint arXiv:2409.17106*, 2024b.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Joseph George Lambourne, Karl Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. Reconstructing editable prismatic cad from rounded voxel models. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

Boyi Li, Philipp Wu, Pieter Abbeel, and Jitendra Malik. Interactive task planning with language models. *arXiv preprint arXiv:2310.10645*, 2023a.

Junnan Li et al. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023b.

Ming Li, Pan Zhou, Jia-Wei Liu, Jussi Keppo, Min Lin, Shuicheng Yan, and Xiangyu Xu. Instant3d: instant text-to-3d generation. *International Journal of Computer Vision*, pages 4456–4472, 2024.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Alexander Gaunt. Competition-level code generation with alphacode. *Science*, pages 1092–1097, 2022.

Haotian Luo, Yixin Liu, Peidong Liu, and Xianggen Liu. Vector-quantized prompt learning for paraphrase generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13389–13398, Singapore, December 2023. Association for Computational Linguistics.

Maxwell Nye, Michael Tessler, Josh Tenenbaum, and Brenden M Lake. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. *Advances in Neural Information Processing Systems*, pages 25192–25204, 2021.

Thomas Paviot. pythonocc-core: Python bindings for opencascade technology, 2022.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. Evaluating the text-to-sql capabilities of large language models, 2022a.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*, 2022b.

Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. Learning design patterns with bayesian grammar induction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 63–74, 2012.

Bohan Wang, Yuxuan Zhang, Zhuosheng Liu, Qiang Yu, Yiming Zhang, Yanan Wang, and Zheng Li. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Sean Welleck, Peter West, Jize Cao, and Yejin Choi. Symbolic brittleness in sequence models: On systematic generalization in symbolic mathematics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8629–8637, 2022.

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

Chengkai Wu, Karl D.D. Willis, Yewen Zhang, Chuang Gan, and Wojciech Matusik. Deepcad: A deep generative network for computer-aided design models. *Computer Graphics Forum*, pages 65–76, 2021.

Shinn Yao et al. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023.

Qinggang Zhang, Hao Chen, Junnan Dong, Wentao Li, Feiran Huang, and Xiao Huang. Structure-guided large language models for text-to-sql generation.

Wenbo Zhang, likai tang, Site Mo, Xianggen Liu, and Sen Song. Learning robust rule representations for abstract reasoning via internal inferences. In *Advances in Neural Information Processing Systems*, volume 35, pages 33550–33562. Curran Associates, Inc., 2022.

Qiang Zou and Guoyue Luo. Geometric modeling for microstructure design and manufacturing: A review of representations and modeling algorithms. *Computer-Aided Design*, page 103834, 2025.

Appendix A. Prompting-only ICL Baseline

We show here the prompting template used for the in-context learning (ICL) baseline. The model is instructed to translate natural language descriptions into valid Sketch-Extrude language sequences that can be executed by a CAD engine to reconstruct the described geometry.

Prompt

```
You are an expert CAD assistant. Your task is to translate natural language
descriptions of 3D objects into valid Sketch-Extrude language sequences for
reconstruction.

## Sketch-Extrude Language Specification
Each sequence begins with <SOL> to start a sketch and ends with <EOS>.
Within a sketch, lines (L), arcs (A), and circles (R) can be used to define
profiles with their positions and sizes. Finally, an extrusion (E) lifts the
sketch into 3D, with parameters that set its orientation, offsets, scaling,
depth, and whether it creates a new body or modifies an existing one.
```

```
## Output Rules
1. Only output the Sketch-Extrude language sequence.
2. Do not add explanations or comments.
3. Ensure the sequence is syntactically valid and executable.

## Examples
[Input]
The CAD model features a thick, band-like structure reminiscent of a ring,
prominently displaying a circular cross-section. The design comprises two
concentric circles, forming the inner and outer boundaries, with the inner
circle slightly smaller, creating a thin encircling wall that characterizes
the object as a hollow cylinder.

[Output]
<SOL>_1
   Circle_2: (176, 128, 48)
<SOL>_3
   Circle_4: (176, 128, 40)
   Extrude_5: (192, 128, 128, 128, 128, 224, 192, 144, 128, NewBody,
   Symmetric)
<EOS>_6

## Task
Now translate the following description into a Sketch-Extrude language
sequence:

[Input]
The CAD model features a thick, band-like structure reminiscent of a ring,
formed by two concentric circles with a small gap between them.

[Output]
```

Table 5: Prompting-only ICL baseline on *Fusion360Gallery*. Metrics: Chamfer Distance (CD, lower is better) and Invalid ratio (IR, lower is better).

Method	Mean CD ↓	Median CD ↓	Invalid ↓
Qwen2.5-7B (1-shot)	88.37	79.80	66.29%
Qwen2.5-7B (3-shot)	97.80	87.42	54.25%
CAD-HLLM (ours)	**61.86**	**44.36**	**4.53%**

Table 5 reports the quantitative evaluation on Fusion360Gallery dataset. Both 1-shot and 3-shot ICL yield extremely high invalid ratios (>50%), showing that prompt-only approaches are insufficient for reliable text-to-CAD generation. In contrast, our finetuned CAD-HLLM maintains low invalidity and much stronger geometric fidelity.