# Pointer-CAD v2: Plan-Then-Construct CAD Generation with Dimension-Aware Parametric Precision

Dacheng Qi[1,2,3], Chenyu Wang[1,2,3], Jingwei Xu[4], Yi Ma[1,2,3,5], and Shenghua Gao[⋆,1,2,3]

[1] The University of Hong Kong, Hong Kong SAR
[2] Shenzhen Loop Area Institute, Shenzhen, China
[3] TranscEngram, Shenzhen, China
[4] Monash University, Melbourne, Australia
[5] University of California, Berkeley, USA

**Abstract.** Computer-aided design (CAD) plays a fundamental role in modern manufacturing by providing the high precision required for industrial production. Recent large language model based approaches formulate CAD generation as a sequence prediction problem and have achieved promising results. However, existing methods and evaluation protocols primarily emphasize visual similarity, while overlooking precise geometric parameters and correct metric scale. Small numerical deviations that are negligible at the shape-level may still violate industrial tolerance requirements, a problem further compounded by current autoregressive paradigms that utilize command sequence representations, aggressively quantize numerical parameters to ease LLM prediction. In this work, we present **Pointer-CAD v2**. Compared with **v1** [36], this version directly predicts continuous values, bypassing the need for quantized numerical parameters and thereby eliminating quantization errors. Specifically, we propose a unified framework that decouples parameter reasoning from geometric construction through a **Plan-Then-Construct paradigm**. Our method first produces a structured design plan with explicit metric scale parameters. These parameters are organized into a dictionary and directly referenced during sequence generation via a pointer mechanism, eliminating discretization errors and ensuring dimensionally consistent execution. In addition, we construct a new large-scale dataset with plan-level annotation and introduce three hierarchical geometry accuracy metrics to evaluate parametric fidelity at the vertex, edge, and face levels. Extensive experiments demonstrate that Pointer-CAD v2 consistently outperforms existing baselines and achieves substantial improvements in geometric accuracy, enabling reliable CAD generation for precision-critical engineering applications. Our code is available at `https://github.com/Snitro/Pointer-CAD-v2`.

**Keywords:** Computer-Aided Design · Large Language Model

⋆ Corresponding author: `gaosh@hku.hk`

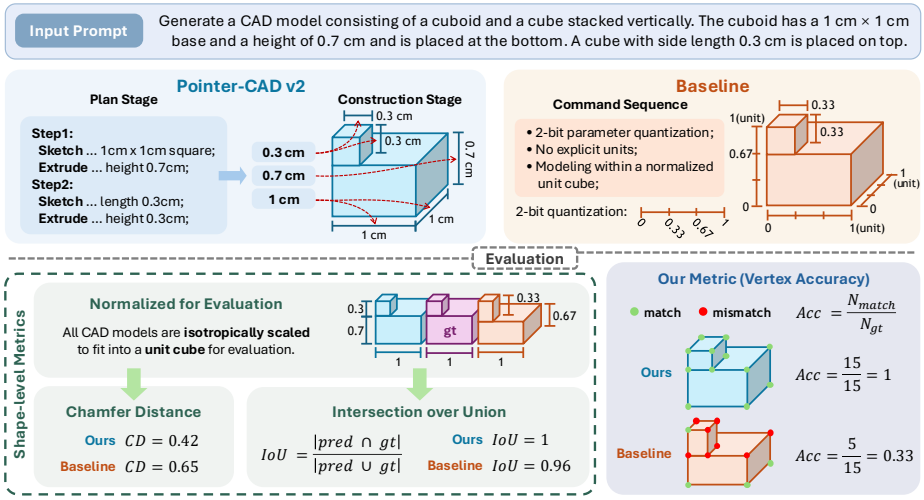


**Fig. 1: Strength of Our Proposed Method and Metric.** Pointer-CAD v2 generates CAD models with full-precision parameters and dimensional consistency by explicitly separating parameter reasoning from command construction. In contrast, command-sequence baselines rely on quantization and cannot accurately represent precise dimensions. Unlike shape-level metrics such as Chamfer Distance and Intersection over Union, our proposed metric spatially measures parametric correctness and is more sensitive to dimensional errors rather than visual similarity.

## 1 Introduction

Computer-Aided Design (CAD) plays a fundamental role in modern industrial design and manufacturing, enabling simulation and fabrication workflows under strict, near-zero tolerance precision requirements. Recent advances in large-scale CAD datasets [5, 7, 46] and generative modeling [8, 11, 19] have enabled automatic synthesis of parametric CAD programs [10, 23, 27], yet ensuring accurate and dimensionally consistent parameters remains an open challenge, as errors in dimensional values can severely impact production-level engineering processes.

Existing methods [18, 37, 39, 61] formulate CAD generation as sequence prediction under an autoregressive paradigm and can be broadly categorized into two groups: command sequence representations and code representations. Code representations generate programs in scripting environments such as CADQuery [48] or the FreeCAD API [31, 33]. They align naturally with large language models (LLMs) [42, 52] and programming ecosystems [2, 13], but generating a single model is often token-intensive due to the verbosity of uncompressed code [36]. In contrast, command sequence representations [49] use specialized token vocabularies to describe parametric operations, offering higher token efficiency. However, they remain largely limited to sketch-extrude pair operations and struggle with high-precision modeling due to the quantization of dimensions introduced to simplify autoregressive generation. Recently, Pointer-CAD [36] extends the

representation capability of command sequence models through explicit entity referencing, enabling support for more diverse CAD operations. Precise dimensional control, however, remains unresolved and limits its applicability in industrial and precision-sensitive scenarios.

The design of command sequence representations inherently constrains precise geometric and dimensional accuracy, since parameters and geometric operations share the same tokenized space [36, 46]. As a result, continuous geometric quantities must be quantized into a finite vocabulary, leading to a loss of parametric fidelity. Existing evaluation metrics further obscure this issue. Widely adopted metrics such as Chamfer Distance (CD) and Intersection over Union (IoU), illustrated in Fig. 1, mainly measure shape-level similarity and are insensitive to small numerical deviations. However, such deviations are unacceptable in industrial CAD settings that require strict dimensional tolerances. Although Pointer-CAD [36] mitigates quantization errors through its pointer mechanism, particularly for sketch plan localization and special edge endpoint positioning (e.g., midpoints or existing edge endpoints), small numerical deviations may still arise in other unconstrained geometric configurations. These deviations are subtle and are not explicitly captured by shape-level evaluation metrics. As a result, neither existing methods nor evaluation metrics are explicitly optimized for accurate metric scale, which limits their applicability in precision-critical engineering design. To address these limitations, we propose a unified framework that decouples parameter reasoning from geometric construction while preserving accurate parameters together with their associated units. Our method, termed **Pointer-CAD v2**, extends Pointer-CAD with a structured Plan-Then-Construct paradigm. Pointer-CAD generates CAD models in a step-wise manner and employs a pointer mechanism to reference intermediate geometric entities. Our method augments each step with a plan stage that produces a structured, dimension-aware design plan. The parameters defined in the plan are extracted to form a parameter dictionary, which is then used to generate the command sequence. Instead of predicting parameters in a quantized space, a pointer-based mechanism directly selects values from the parameter dictionary during sequence generation. This design eliminates discretization errors, decouples numerical reasoning from geometric construction, and ensures dimensionally consistent parameter usage.

To enable structured parameter planning with explicit unit grounding and rigorous evaluation, we construct a new dataset and introduce geometry accuracy metrics. Built upon Recap-OmniCAD and Recap-OmniCAD$^+$ [36], where the latter additionally includes models with *chamfer* and *fillet*, we establish plan-level supervision by annotating structured design plans for each CAD model using Qwen3 [52]. This results in OmniCAD-Plan with 202K unique valid models and OmniCAD-Plan$^+$ with unique 209K valid models. We further propose geometry accuracy metrics that spatially measure parametric correctness rather than visual similarity. The evaluation protocol verifies whether generated models satisfy the intended dimensional constraints within predefined tolerances at three geometric levels, including vertex, edge, and face. As illustrated in Fig. 1,

these metrics are more sensitive to dimensional deviations and better distinguish visually similar models with different numerical parameters. Under the proposed dataset and evaluation protocol, Pointer-CAD v2 significantly outperforms existing baselines across three geometric levels, demonstrating superior parametric accuracy and dimensional consistency.

Our contributions are summarized as follows: (1) We propose a Plan-Then-Construct framework for CAD generation and introduce Pointer-CAD v2, which explicitly separates parameter reasoning from geometric construction and enables the generation of dimensionally accurate CAD models; (2) We introduce a new dataset and hierarchical metrics for training and evaluating parametric fidelity beyond visual similarity; (3)Extensive experiments demonstrate that Pointer-CAD v2 outperforms baseline methods by eliminating the need to predict quantized numerical parameters, improving the dimensional precision.

## 2 Related Work

### 2.1 Command Sequence Representations

Command sequence representations have become a popular paradigm for CAD generation, largely enabled by the availability of large-scale and diverse datasets [20, 45, 46]. Early work such as DeepCAD [46] formulates CAD modeling as sequence prediction over parametric operations, while subsequent methods improve structural modeling capacity through hierarchical designs [50,51] and text-conditioned generation [20]. Recent advances further enhance generation fidelity with more expressive architectures, including diffusion-based models [30, 55, 57] and LLMs [26,44,60]. In parallel, reverse-engineering approaches aim to translate alternative representations, including B-rep [28, 53, 58], point clouds [9, 21, 30], and images [3, 5, 47], into executable command sequences. CAD-MLLM [49] further unifies multiple modalities within a single framework. Pointer-CAD [36] introduces a pointer mechanism for explicit geometric referencing, expanding the expressive capacity of command sequence. Despite these advances, most existing methods operate in normalized and quantized parameter spaces, leading to discretization errors and missing explicit metric dimension information.

### 2.2 Code Representations

With the rapid progress of LLMs in code generation [4, 59], code-based representations have become an alternative paradigm for CAD modeling. Early studies often designed domain-specific languages for CAD representation [12, 15], while recent works adopt mature CAD scripting environments such as CadQuery [16, 24, 33, 48], PythonOCC [56], and FreeCAD [31, 33]. By grounding generation in executable APIs, these approaches leverage the coding capabilities of pre-trained LLMs. Code representations have also been extended beyond text-to-CAD synthesis to reverse engineering from point clouds or images [38,54], as well as multi-modal and agent-based frameworks [22, 31]. Leveraging mature

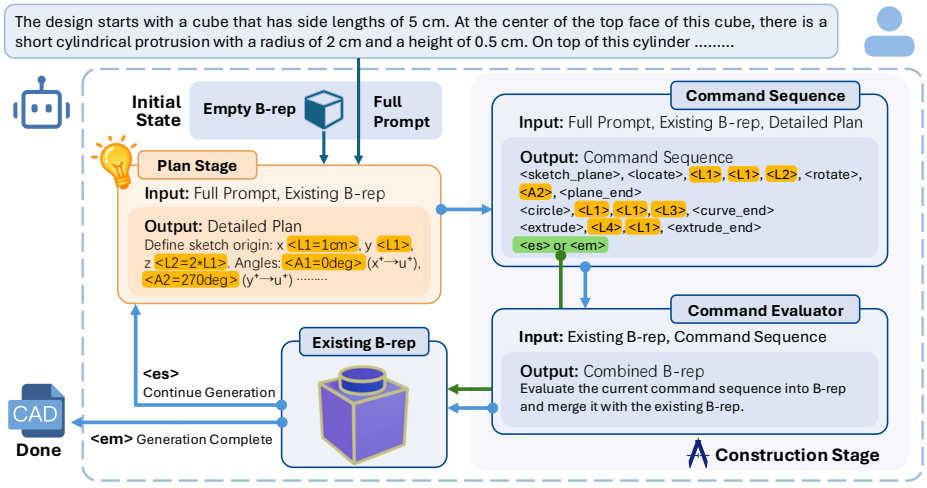


**Fig. 2: Pointer-CAD v2 Pipeline.** Pointer-CAD v2 generates CAD models in a step-wise manner under a Plan-Then-Construct framework. Each step includes a plan stage and a construction stage. The plan stage produces a structured, dimension-aware design plan conditioned on the full prompt and the existing B-rep. The construction stage retrieves the planned parameters to form a command sequence, which is constructed to update the B-rep iteratively until completion.

CAD scripting environments, code-based methods can naturally represent high-precision parameters with explicit dimensions. However, they typically require around four times more tokens than command sequence representations [36], reducing both training and inference efficiency.

### 2.3 Planning-Based CAD Generation

Planning-based CAD generation seeks to decouple high-level design reasoning from low-level geometric construction [17]. Prior works introduce intermediate planning stages to improve interpretability and long-horizon reasoning. For instance, [25] represent 2D sketching as sequential geometric constructions, while FreeCAD [29] and CAD-Assistant [31] generate structured plans to guide subsequent tool calls. Seek-CAD [26] integrates visual feedback with structured reasoning for iterative self-refinement. In parallel, concurrent works [16, 34, 43] incorporate Chain of Thought (CoT) reasoning and reinforcement learning to enhance long-horizon generation. However, these approaches do not establish an explicit linkage between planning outputs and metric scale parameters.

## 3 Method

To enable precise and flexible CAD generation with explicit dimension-aware parameter reasoning, we propose a **Plan-Then-Construct** framework built upon

**Dimension-Aware Design Plan**

Create a sketch on the Bottom plane. Define sketch origin: x <L1=0mm>, y <L1>, z <L1> Rotation: $x^+$ and $y^+$ project onto the plane; $z^+$ is perpendicular → ignored. Angles: <A1=0deg> ($x^+ \rightarrow u^+$), <A2=90deg> ($y^+ \rightarrow u^+$). Define inner circle center: u <L1>, v <L1>. Radius <L2=5mm>. Define outer circle center: u <L1>, v <L1>. Radius <L3=2 * L2>. Extrude: normal <L4=3mm>, reverse-normal <L1>

Corresponding model

**Fig. 3: Example of a dimension-aware design plan.** The plan lists geometric parameters in symbolic form. Each parameter is enclosed in <> and labeled as $L$ for length or $A$ for angle. All parameters are associated with units and may reference previously defined parameters of the same type.

Pointer-CAD [36]. As illustrated in Fig. 2, the framework decomposes generation into two stages: a plan stage and a construction stage. The plan stage performs structured reasoning in the textual domain and produces an explicit geometric plan with complete dimensions. The construction stage converts the plan into executable CAD command sequences. We first review Pointer-CAD in Sec. 3.1, and then describe the plan and construction stages in Sec. 3.2 and Sec. 3.3.

### 3.1 Preliminary: Pointer-CAD

Pointer-CAD [36] extends the representation capacity of command sequences by introducing a pointer mechanism that enables direct referencing of geometric entities in the intermediate B-rep. Instead of generating the entire CAD model in a single forward pass, it adopts a step-wise paradigm to progressively construct the model. At each step, the model predicts one fundamental operation, including a sketch-extrude pair, a chamfer, or a fillet, conditioned on the current B-rep built from all previous operations. To support explicit geometric referencing, Pointer-CAD encodes all entities in the current B-rep into a set of embeddings, forming a candidate pool for selection. During command sequence generation, when an operation needs to reference an existing geometric entity, the language model predicts an embedding vector in the shared space. The entity with the highest cosine similarity is selected as the output. This design enables accurate and flexible entity referencing across diverse shapes and topologies, enhancing the representation ability of command sequences for complex CAD modeling.

### 3.2 Dimension-Aware Plan Generation

To enable command sequence representations to express continuous parameters with metric dimensions in the step-wise paradigm, we introduce a structured plan representation in the textual domain. Unlike conventional command sequence methods [20, 46, 49], which predict discretized numerical tokens, our plan stage conducts dimension-aware parameter reasoning before command construction.

At each step, the model generates a structured design plan that specifies all associated geometric parameters in symbolic form. As illustrated in Fig. 3, all

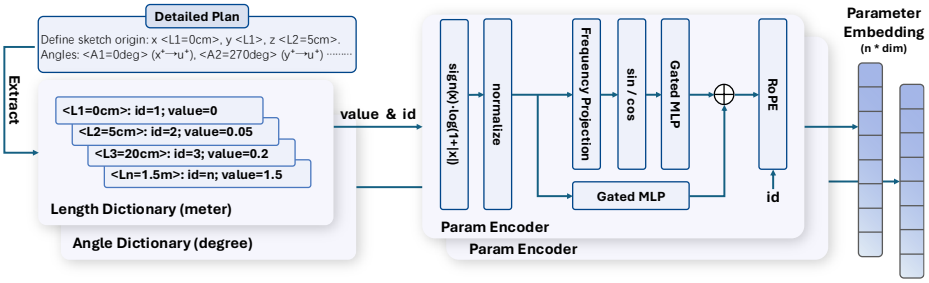


**Fig. 4: Plan parameter encoding pipeline.** All parameters are first extracted from the generated plan and organized into separate length and angle dictionaries. Each dictionary is processed by a parameter encoder that captures both frequency and contextual information. The encoder maps every parameter to a dedicated embedding, forming a set of parameter embeddings for subsequent modeling.

referable parameters are enclosed in <> and explicitly typed as either $L$ (length) or $A$ (angle). Each parameter is bound to a unit of measurement (e.g., degrees, meters, millimeters) and supports arithmetic operations as well as references to previously defined parameters of the same type. This design represents geometric quantities in a metric-scale accurate and non-quantized manner, providing explicit guidance for the subsequent construction stage.

### 3.3 Plan-Based Construction Stage

During command sequence generation, parameters defined in the plan are retrieved through a similarity-based pointer mechanism. This improves the dimension precision of the predicted command sequence.

*Parameter Extraction and Normalization.* As illustrated in Fig. 4, we first extract all parameters from the generated plan and separate them into length and angle types. Each parameter is converted into a unified unit system (meters for length and degrees for angle). Since numerical parameter magnitudes may vary significantly after unit conversion, we apply a sign-preserving logarithmic normalization to alleviate scale imbalance:

$$\tilde{x} = \frac{\operatorname{sign}(x) \cdot \log\left(1 + |x|\right)}{\max_{x' \in \mathcal{P}} \left|\operatorname{sign}(x') \cdot \log\left(1 + |x'|\right)\right| + \epsilon}, \tag{1}$$

where $\mathcal{P}$ denotes the parameter set extracted from the plan and $\epsilon$ is a small constant for numerical stability.

*Frequency Encoding.* To enhance representation capacity across diverse numerical ranges, we encode the normalized value $\tilde{x}$ using exponentially increasing frequency bands:

$$\mathbf{z} = \tilde{x} \cdot s \cdot \mathbf{b}, \tag{2}$$

where $s \in \mathbb{R}$ is a learnable scale and $\mathbf{b} = [2^0, 2^1, \ldots, 2^{K-1}]$ denotes predefined frequency bands. The resulting Fourier feature [32, 41] is

$$\phi(\tilde{x}) = [\sin(\mathbf{z}), \cos(\mathbf{z})]. \tag{3}$$

*Parameter Embedding.* To unify dimensionality, both $\tilde{x}$ and $\phi(\tilde{x})$ are projected into a shared embedding space via a gated MLP block $\mathcal{H}(\mathbf{u})$ [6], defined as:

$$\mathcal{H}(\mathbf{u}) = \text{LayerNorm}\left(\sigma(W_g\mathbf{u}) \odot \text{MLP}(\mathbf{u})\right), \tag{4}$$

where $\sigma(\cdot)$ denotes the sigmoid function and $\odot$ represents element-wise multiplication. The final parameter embedding is computed as

$$\mathbf{e} = \frac{1}{2}\left(\mathcal{H}(\tilde{x}) + \mathcal{H}(\phi(\tilde{x}))\right). \tag{5}$$

To encode parameter identity and positional information, we further apply Rotary Positional Encoding (RoPE) [40] to $\mathbf{e}$.

*Similarity-Based Retrieval.* During command sequence generation, extending Pointer-CAD [36], where geometry entity selection activates the pointer to retrieve a target, our mechanism is also triggered whenever a numerical parameter is required, then the LLM predicts an embedding vector $\tilde{\mathbf{e}}$ in the shared embedding space. We compute cosine similarity between $\tilde{\mathbf{e}}$ and all candidate parameter embeddings $\mathbf{e}_i$:

$$\text{similarity}(\tilde{\mathbf{e}}, \mathbf{e}_i) = \frac{\tilde{\mathbf{e}} \cdot \mathbf{e}_i}{|\tilde{\mathbf{e}}| \cdot |\mathbf{e}_i|}. \tag{6}$$

The parameter with the highest similarity is selected and inserted into the command sequence.

*Progressive Command construction.* Following [36], after similarity-based retrieval, the command sequence is executed immediately to update the current B-rep. The updated geometry is then fed back to the model to condition the next prediction, enabling progressive construction of the CAD model. At each step, the plan tokens and the command tokens are generated sequentially within a single forward pass using two modality-specific decoding heads. A dedicated control token separates the two modalities, ensuring coordinated yet disentangled generation.

Overall, this plan-based progressive command construction strategy ensures that all geometric operations strictly follow the metric scale continuous parameters defined in the plan stage, eliminating quantization errors in conventional command representations.

## 4 Experiments

### 4.1 Baselines

To the best of our knowledge, no prior work explicitly models metric scale geometric parameters for CAD generation. We therefore adapt two representative

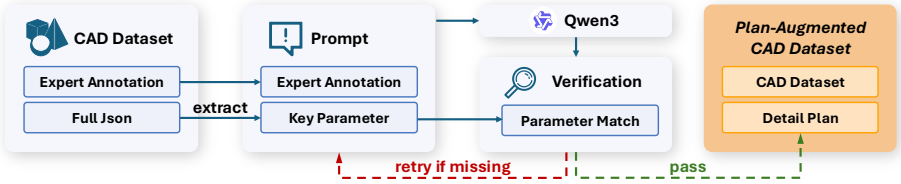


**Fig. 5: Plan Construction Pipeline.** Key parameters are extracted from raw JSON files and combined with expert annotations to form structured prompts. These prompts are fed into Qwen3 to generate detailed design plans. The generated plans are verified for parameter completeness. If required parameters are missing, Qwen3 is re-prompted up to two times, and any remaining failures are discarded.

methods with different parameter representations as our baselines: the command-sequence-based Pointer-CAD [36] and the code-based CADmium [15].

Pointer-CAD generates geometry in a normalized unit cube without explicit units. For fair comparison under metric scales, we treat it as a $1 \times 1 \times 1$ meter volume and add an autoregressive head to predict a global scaling factor with an $\ell_1$ loss. In addition, we employ a dedicated text-generation head to predict all involved parameters together with their units in textual form, forming a parameter book. After rescaling, each geometric parameter is replaced by its nearest entry in the book to maintain consistency between geometry and parameter prediction. For CADmium, we replace all normalized parameters in the dataset with their original values and units for both training and testing.

In parallel, recent LLMs have shown the ability to perform text-conditioned CAD generation by producing executable CADQuery code. We additionally evaluate several LLMs under this paradigm, including Qwen3 [52], Gemini [14], GPT [35], and Claude [1].

### 4.2 Dataset

We construct our dataset following the pipeline in Fig. 5, based on the datasets released with Pointer-CAD [36], which preserve true geometric dimensions without normalization. Operation-specific parameters are extracted from the JSON files and organized into a compact JSON hint. The hint, expert annotation, and our designed prompt are fed into Qwen3 to generate a detailed design plan, which is then verified for parameter completeness. If any parameters are missing, Qwen3 is re-prompted with a non-zero temperature for up to two additional attempts. The final dataset contains only validated plans. Further details and data statistics are provided in the supplementary material.

Applying this pipeline to Recap-OmniCAD and its extended version Recap-OmniCAD$^+$, which includes *chamfer* and *fillet* operations, we obtain the plan-augmented datasets OmniCAD-Plan and OmniCAD-Plan$^+$, used for all subsequent training and evaluation.

### 4.3 Metrics

Standard geometric distance metrics, such as Chamfer Distance(CD), are often insufficient for evaluating CAD generation from textual instructions. CD mainly captures coarse geometric structure, but is less sensitive to fine-grained geometric details (e.g., specific radii or point positions) mentioned in textual prompts and it cannot distinguish whether the underlying B-Rep structure (vertices, edges, and faces) faithfully reconstructs these precise geometric features or merely approximates the overall shape.

To bridge this gap and strictly assess whether the generated models adhere to the dimensional and structural constraints specified in text, we propose three hierarchical metrics: Vertex Accuracy, Edge Accuracy, and Face Accuracy. These metrics shift the evaluation from coarse structural similarity to fine-grained parametric alignment.

For all three metrics, we define the Accuracy Ratio ($Acc$) as:

$$Acc = \frac{N_{match}}{N_{gt}} \times 100 \tag{7}$$

where $N_{match}$ denotes the number of predicted elements correctly matched with the ground truth (GT), and $N_{gt}$ denotes the total number of corresponding elements in the GT model. Unlike CD, our matching process requires strict parametric consistency:

- **Vertex Accuracy:** This metric evaluates the fidelity of edge endpoints. A predicted vertex is considered correctly matched if its Euclidean distance to the nearest GT point is within a predefined tolerance $\epsilon$. For circular primitives, the center point is utilized as the representative coordinate for matching.
- **Edge Accuracy:** An edge is identified as a match only if its endpoints are correctly localized and its underlying *geometric parameters*, (e.g., the center and radius for arcs, or the normal direction for circles) align with the GT within the tolerance $\epsilon$. This metric serves as a direct proxy for the model's capability to adhere to precise dimensional constraints (e.g., "a cylinder with a 5mm radius") specified in textual instructions.
- **Face Accuracy:** A face is deemed correctly matched if all its constituent boundary edges are successfully identified and its *primitive type* (e.g., plane, cylinder, cone, sphere, or torus) is consistent with the GT.

To maintain consistency across various object scales, the tolerance $\epsilon$ is defined relative to the GT bounding box:

$$\epsilon = 0.001 \times \min\left(\mathbf{b}_{gt}^{max} - \mathbf{b}_{gt}^{min}\right), \tag{8}$$

where $\mathbf{b}_{gt}^{min}$ and $\mathbf{b}_{gt}^{max}$ represent the bounding box extrema. Crucially, all models are evaluated in their original location and scale without normalization, forcing the model to demonstrate a true understanding of metric scale parameters as described in the textual instructions.

**Table 1: Quantitative comparison on the OmniCAD-Plan dataset.** Our method consistently outperforms all baselines across three geometric levels.

| Method | RMR@3 | Vertex | Edge | | | | Face | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Avg | Line | Circle | Arc | Avg | Plane | Cylinder |
| CADmium-0.5B | 78.60 | 76.37 | 74.34 | 76.05 | 72.41 | 4.55 | 70.57 | 71.47 | 66.56 |
| Pointer-CAD-0.5B | 66.03 | 58.87 | 54.81 | 54.30 | 59.18 | 7.34 | 54.70 | 54.44 | 55.88 |
| Ours-0.5B | 87.29 | 90.10 | 86.40 | 85.19 | 92.21 | 64.54 | 84.85 | 83.89 | 89.05 |
| CADmium-1.5B | 81.15 | 79.36 | 78.25 | 80.58 | 74.03 | 5.61 | 74.19 | 75.56 | 67.99 |
| Pointer-CAD-1.5B | 72.10 | 64.67 | 62.35 | 62.89 | 62.40 | 24.83 | 61.69 | 62.19 | 59.50 |
| Ours-1.5B | **91.25** | **92.76** | **90.33** | **89.65** | **94.23** | **68.53** | **89.18** | **88.73** | **91.14** |

**Table 2: Quantitative comparison on the OmniCAD-Plan$^+$ dataset.** Our method also outperforms all baselines on the more complex dataset.

| Method | RMR@3 | Vertex | Edge | | | | Face | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Avg | Line | Circle | Arc | Avg | Plane | Cylinder |
| CADmium-0.5B | 42.43 | 38.92 | 34.82 | 36.10 | 35.67 | 0.74 | 28.25 | 29.05 | 26.06 |
| Pointer-CAD-0.5B | 54.65 | 47.41 | 42.96 | 44.40 | 43.90 | 5.92 | 42.16 | 43.41 | 38.52 |
| Ours-0.5B | 83.33 | 86.95 | 81.84 | 80.41 | 90.03 | 61.21 | 80.17 | 79.22 | 84.16 |
| CADmium-1.5B | 43.41 | 40.15 | 36.96 | 38.88 | 35.88 | 0.28 | 29.80 | 30.99 | 26.15 |
| Pointer-CAD-1.5B | 61.42 | 53.53 | 50.63 | 52.69 | 48.19 | 17.52 | 49.62 | 51.34 | 43.94 |
| Ours-1.5B | **90.97** | **92.60** | **90.13** | **89.91** | **94.03** | **72.29** | **88.78** | **88.77** | **89.64** |

We further introduce a repair-aware metric, termed Repairable Model Rate (RMR). In practical CAD workflows, models with only a few geometric errors can easily be corrected through post-processing. To reflect this property, a generated model is regarded as repairable under RMR@3 if the number of incorrectly predicted faces does not exceed three. RMR@3 measures the proportion of such repairable models over the entire test set, and complements strict parametric evaluation by capturing practical usability.

### 4.4 Implementation Details

We use Qwen2.5-0.5B [52] as the backbone LLM for Pointer-CAD v2 unless stated otherwise. The model is trained for 10 epochs on NVIDIA H800 GPUs. Additional implementation details are provided in the supplementary material.

### 4.5 Comparison with Specialized Text-to-CAD Methods

In this section, we compare our method with Pointer-CAD [36] and CADmium [15], under proposed geometry-aware metrics and conventional metrics.

*Evaluation on OmniCAD-Plan.* As shown in Tab. 1, our method outperforms all baselines on OmniCAD-Plan across all three metric levels. The 1.5B model improves average accuracy by 13.49% over CADmium-1.5B, demonstrating strong

**Table 3: Quantitative comparison under conventional metrics (F1, CD).** All methods are evaluated on OmniCAD-Plan with 0.5B models. Our method matches Pointer-CAD on Line and Circle F1, which are near saturation, while significantly improving Arc F1. CD gains are marginal, as enhanced parameter accuracy is not adequately captured by rough shape-level geometric metrics.

| Method | Line F1↑ | Arc F1↑ | Circle F1↑ | Extrusion F1↑ | Mean CD↓ | Median CD↓ |
|---|---|---|---|---|---|---|
| CADmium | 85.30 | 14.58 | 76.04 | 88.49 | 7.39 | 0.27 |
| Pointer-CAD | 96.62 | 51.00 | 98.08 | **99.73** | 3.69 | 0.24 |
| Ours | **96.75** | **63.59** | **98.48** | 99.66 | **3.15** | **0.19** |

**Table 4: Quantitative comparison with general-purpose LLMs on OmniCAD-Plan.** Our method consistently outperforms all evaluated general-purpose LLMs across the three-level metrics.

| Method | RMR@3 | Vertex | Edge | | | | Face | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Avg | Line | Circle | Arc | Avg | Plane | Cylinder |
| Qwen3-235B-A22B | 58.34 | 53.28 | 49.62 | 48.09 | 56.03 | 30.14 | 46.75 | 46.95 | 45.93 |
| Claude Opus 4.5 | 61.34 | 55.41 | 52.77 | 49.21 | 65.86 | 30.00 | 52.31 | 51.09 | 57.21 |
| GPT-5.2 | 69.16 | 61.03 | 63.55 | 63.11 | 66.59 | 38.36 | 60.52 | 61.41 | 56.94 |
| Gemini 3 Pro | 74.65 | 68.93 | 69.51 | 68.96 | 73.12 | 39.73 | 67.02 | 67.54 | 64.93 |
| Ours-0.5B | 87.77 | 89.24 | 84.34 | 82.17 | 92.61 | 56.16 | 83.19 | 81.88 | 88.44 |
| Ours-1.5B | **90.89** | **92.18** | **89.26** | **88.21** | **93.60** | **67.12** | **87.93** | **87.57** | **89.37** |

parametric precision. It also achieves an RMR@3 of 91.25%, indicating high usability in practical CAD workflows.

*Evaluation on OmniCAD-Plan$^+$.* For OmniCAD-Plan$^+$, we train CADmium on OmniCAD-Plan, as it does not support the additional operations. Results in Tab. 2 show that our method consistently outperforms all baselines, demonstrating robustness to increased geometric complexity and richer operations.

*Performance under Conventional Metrics.* Following prior work [15, 20, 36], we also report conventional metrics, including F1 score and CD. All 0.5B models are evaluated on OmniCAD-Plan. As shown in Tab. 3, while F1 scores for Line and Circle primitives have reached near-saturation, our method still achieves a significant improvement in Arc F1. Pointer-CAD v2 shows only marginal improvements over Pointer-CAD under CD, which is expected since CD is computed after normalization and is insensitive to small dimensional differences. Thus, minor deviations in length or angle have limited impact on shape similarity, and improvements in parameter accuracy are not fully reflected.

### 4.6 Comparison with General-Purpose LLMs

We randomly sample 2,000 models from the test sets of OmniCAD-Plan and OmniCAD-Plan$^+$ and prompt each LLM to generate CadQuery code. Results

**Table 5: Quantitative comparison with general-purpose LLMs on OmniCAD-Plan$^+$.** On the more complex dataset, our method consistently outperforms all evaluated general-purpose LLMs by a larger margin.

| Method | RMR@3 | Vertex | Edge | | | | Face | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Avg | Line | Circle | Arc | Avg | Plane | Cylinder |
| Qwen3-235B-A22B | 56.18 | 49.01 | 45.37 | 43.07 | 54.21 | 49.62 | 43.51 | 42.57 | 47.36 |
| Claude Opus 4.5 | 61.28 | 54.81 | 52.73 | 48.48 | 67.90 | 66.17 | 53.20 | 51.05 | 62.04 |
| GPT-5.2 | 66.81 | 58.66 | 59.23 | 56.89 | 68.26 | 64.66 | 57.37 | 56.73 | 60.00 |
| Gemini 3 Pro | 74.13 | 67.40 | 68.11 | 66.96 | 73.88 | 55.64 | 66.51 | 66.33 | 67.20 |
| Ours-0.5B | 83.42 | 84.98 | 79.07 | 76.73 | 90.36 | 52.87 | 78.41 | 76.93 | 85.05 |
| Ours-1.5B | **90.98** | **91.24** | **88.02** | **86.44** | **95.55** | **65.41** | **87.39** | **86.37** | **91.73** |

**Table 6: Ablation study of the pointer mechanism.** The results show that parameter reasoning and plan-based reference are essential, while the pointer mechanism remains robust to scale sensitivity in continuous parameter retrieval.

| Variant | RMR@3 | Vertex | Edge | | | | Face | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Avg | Line | Circle | Arc | Avg | Plane | Cylinder |
| w/o Ref. | 65.55 | 66.30 | 58.45 | 55.55 | 71.14 | 17.65 | 56.96 | 54.86 | 66.30 |
| w/o Plan | 80.93 | 76.13 | 73.44 | 73.44 | 75.50 | 29.15 | 72.89 | 73.23 | 71.38 |
| w/o Freq. | 86.77 | 89.02 | 85.20 | 84.10 | 90.67 | 62.24 | 83.68 | 82.85 | 87.31 |
| w/o Norm. | 86.76 | 89.63 | 85.89 | 84.69 | 91.96 | 56.99 | 84.37 | 83.42 | 88.51 |
| Full | **87.29** | **90.10** | **86.40** | **85.19** | **92.21** | **64.54** | **84.85** | **83.89** | **89.05** |

are reported in Tab. 4 and Tab. 5. Among general-purpose models, Gemini-3-Pro performs best on both datasets, yet our method consistently surpasses it. Our 1.5B model improves over Gemini-3-Pro by 21.30% on OmniCAD-Plan and 21.54% on OmniCAD-Plan$^+$ averaged over the three-level metrics. General-purpose LLMs also achieve much lower average RMR@3 on both datasets (65.88% and 64.60%) than Pointer-CAD v2. These suggest that current LLMs remain unreliable for CAD generation in terms of parametric precision and robustness.

### 4.7 Qualitative Comparison of CAD Generation

*Qualitative Comparison with Baselines.* In Fig. 6, we present qualitative comparisons of generated CAD models. The code-based method CADmium [15] strictly follows numerical dimensions but often produces structural errors, such as missing or incorrectly constructed parts. Pointer-CAD [36] generates geometrically plausible shapes resembling the ground truth, yet small dimensional deviations remain. In contrast, our method achieves both structural completeness and exact dimensional consistency. The generated models are visually faithful to the target geometry while strictly satisfying all specified parameters.

*Complex Generation and Failure Cases.* Fig. 7 shows additional CAD models generated by our 1.5B method. Our approach remains robust across diverse

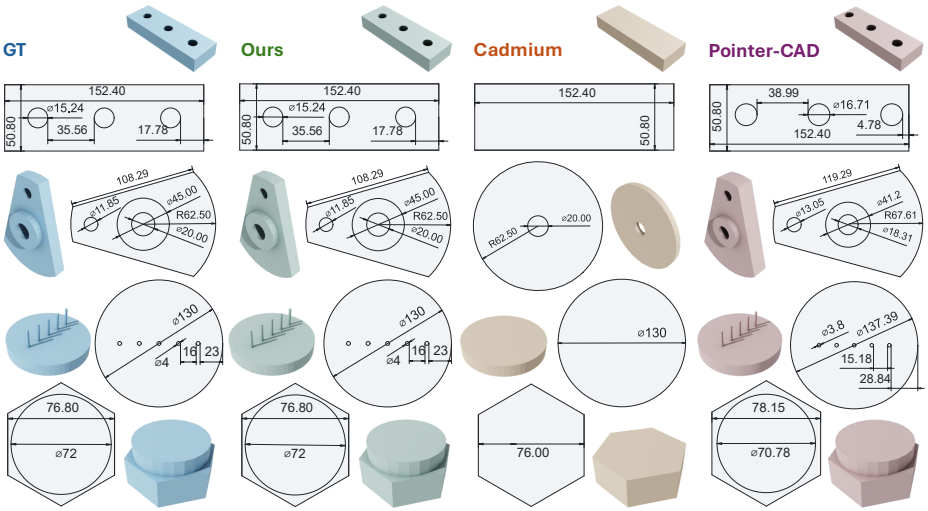


**Fig. 6: Qualitative comparison with specialized text-to-CAD methods.** Cadmium follows the specified dimensions but shows structural errors. Pointer-CAD produces plausible shapes with dimensional deviations. Our method achieves both structural correctness and exact dimensional consistency.

geometries, but occasional inaccuracies appear in very fine features, such as small holes or thin cylinders. More qualitative results and analysis are provided in the supplementary material.

### 4.8 Ablation Study of the Pointer Mechanism

We ablate the pointer mechanism on OmniCAD-Plan using the 0.5B model. As shown in Tab. 6, we evaluate four variants: w/o Ref., which replaces parameter references with direct regression and nearest-value matching; w/o Plan, which removes the planning stage and directly extracts parameters from the raw prompt; w/o Freq., which removes the frequency encoding mechanism; and w/o Norm., which removes the sign-preserving logarithmic normalization.

The w/o Ref. and w/o Plan variants lead to notable degradations of 26.54% and 12.96%, respectively. This suggests that explicit parameter reasoning and plan-based parameter reference are necessary, as raw prompts often omit computed construction parameters and direct regression is easily affected by nearby values. In contrast, w/o Freq. and w/o Norm. cause only minor drops of 1.15% and 0.48%, respectively, suggesting that the pointer mechanism is robust to scale sensitivity in continuous parameter retrieval.

### 4.9 Application: Plan-Based Model Editing

Our plan representation also enables direct editing of generated CAD models. Since geometric parameters are explicitly defined in the plan, modifying them

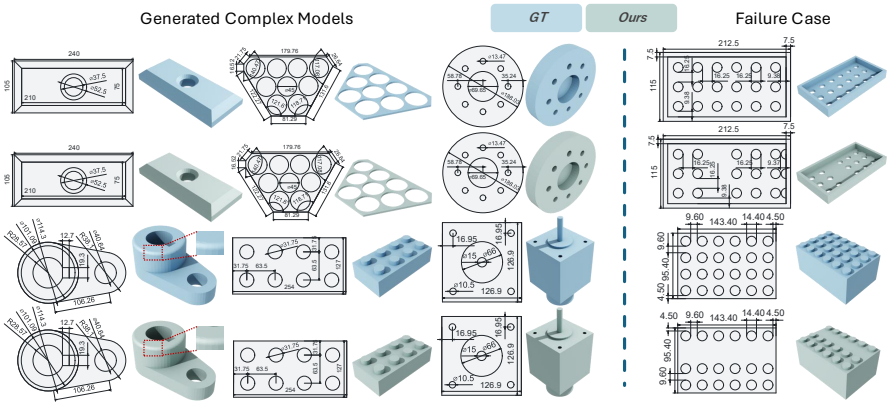


**Fig. 7: Visualization of generation and failure cases.** Our 1.5B method shows strong robustness across diverse geometric compositions, while occasionally failing to capture extremely fine-grained features.

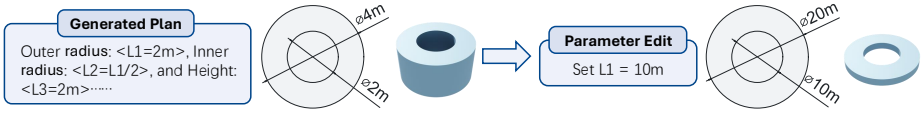


**Fig. 8: Example of CAD model editing via plan modification.** Changing parameters in the plan adjusts the inner and outer radii while preserving relative proportions.

updates the model without regenerating the full sequence while preserving relative proportions (Fig. 8). Details are provided in the supplementary material.

## 5 Conclusion

In this work, we present Pointer-CAD v2, which addresses quantization limitations of command sequence representations through a **Plan-Then-Construct** framework. We generate plan-level annotation, forming a dedicated dataset for structured training. To better align CAD generation with real-world engineering requirements, we introduce a multi-level accuracy metric that explicitly evaluates geometric and parametric correctness. Extensive experiments validate the effectiveness of our approach, achieving state-of-the-art performance.

## 6 Acknowledgments

This work is supported in part by the General Research Fund of the Research Grants Council (grant #17200725), the Shenzhen Loop Area Institute under grants FPF10120250002 and FPF10120260001, and the JC STEM Lab funded by The Hong Kong Jockey Club Charities Trust.

## References


1. Anthropic: Claude opus 4.5 system card (2025), https://www-cdn.anthropic.com/bf10f64990cfda0ba858290be7b8cc6317685f47.pdf
2. CadQuery Contributors: CadQuery (2025), https://doi.org/10.5281/zenodo.14590990
3. Chen, C., Wei, J., Chen, T., Zhang, C., Yang, X., Zhang, S., Yang, B., Foo, C.S., Lin, G., Huang, Q., et al.: CADCrafter: Generating computer-aided design models from unconstrained images. In: CVPR. pp. 11073–11082 (2025)
4. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.D.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021)
5. Chen, T., Yu, C., Hu, Y., Li, J., Xu, T., Cao, R., Zhu, L., Zang, Y., Zhang, Y., Li, Z., et al.: Img2CAD: Conditioned 3-d cad model generation from single image with structured visual geometry. IEEE Transactions on Industrial Informatics (2025)
6. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: ICML. pp. 933–941. PMLR (2017)
7. Dong, X., Li, C., Han, C., Zheng, P., Jing, J., Song, Y., Yang, Z.: HistCAD: Geometrically constrained parametric history-based cad dataset. arXiv preprint arXiv:2602.19171 (2025)
8. Du, Y., Durkan, C., Strudel, R., Tenenbaum, J.B., Dieleman, S., Fergus, R., Sohl-Dickstein, J., Doucet, A., Grathwohl, W.S.: Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In: ICML. pp. 8489–8510. PMLR (2023)
9. Dupont, E., Cherenkova, K., Mallis, D., Gusev, G., Kacem, A., Aouada, D.: TransCAD: A hierarchical transformer for cad sequence inference from point clouds. In: ECCV. pp. 19–36. Springer (2024)
10. Elistratov, M., Barannikov, M., Ivanov, G., Khrulkov, V., Konushin, A., Kuznetsov, A., Zhemchuzhnikov, D.: CADEvolve: Creating realistic cad via program evolution. arXiv preprint arXiv:2602.16317 (2026)
11. Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al.: Scaling rectified flow transformers for high-resolution image synthesis. In: ICML (2024)
12. Fan, R., He, F., Liu, Y., Song, Y., Fan, L., Yan, X.: 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 (2025)
13. FreeCAD Community: FreeCAD (2024), https://www.freecad.org/
14. Google DeepMind: Gemini 3 pro (2025), https://deepmind.google/models/gemini/pro/
15. Govindarajan, P., Baldelli, D., Pathak, J., Fournier, Q., Chandar, S.: CADmium: Fine-tuning code language models for text- driven sequential cad design. TMLR (2026)
16. Guan, Y., Wang, X., Xing, X., Zhang, J., Xu, D., Yu, Q.: CAD-Coder: Text-to-cad generation with chain-of-thought and geometric reward. arXiv preprint arXiv:2505.19713 (2025)
17. Guo, Q., Dai, G., Liu, Z., Huang, S., Hu, Y., Zhang, H., Chen, T.: Plan then Act: Bi-level CAD command sequence generation. In: ICLR (2026), https://openreview.net/forum?id=LhE03r8X8z
18. Jones, B.T., Zhang, Z., Hähnlein, F., Matusik, W., Ahmad, M., Kim, V., Schulz, A.: A solver-aided hierarchical language for llm-driven cad design. In: Computer Graphics Forum. vol. 44, p. e70250. Wiley Online Library (2025)

19. Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. NeurIPS **35**, 26565–26577 (2022)
20. Khan, M.S., Sinha, S., Sheikh, T.U., Stricker, D., Ali, S.A., Afzal, M.Z.: Text2CAD: Generating sequential cad designs from beginner-to-expert level text prompts. NeurIPS **37**, 7552–7579 (2024)
21. Khan, M.S., Dupont, E., Ali, S.A., Cherenkova, K., Kacem, A., Aouada, D.: CAD-SIGNet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In: CVPR. pp. 4713–4722 (2024)
22. Kolodiazhnyi, M., Tarasov, D., Zhemchuzhnikov, D., Nikulin, A., Zisman, I., Vorontsova, A., Konushin, A., Kurenkov, V., Rukhovich, D.: cadrille: Multi-modal cad reconstruction with online reinforcement learning. arXiv preprint arXiv:2505.22914 (2025)
23. Le, T., Nguyen, K., Huang, B., Ta, T.D., Nguyen, A.: CADKnitter: Compositional cad generation from text and geometry guidance. arXiv preprint arXiv:2512.11199 (2025)
24. Li, J., Ma, W., Li, X., Lou, Y., Zhou, G., Zhou, X.: CAD-Llama: leveraging large language models for computer-aided design parametric 3d model generation. In: CVPR. pp. 18563–18573 (2025)
25. Li, S., Lambourne, J.G., Zhang, L., Jayaraman, P.K., Willis, K., et al.: Draw it like Euclid: Teaching transformer models to generate cad profiles using ruler and compass construction steps. arXiv preprint arXiv:2601.09428 (2026)
26. Li, X., Li, J., Song, Y., Lou, Y., Zhou, X.: Seek-CAD: A self-refined generative modeling for 3d parametric cad using local inference via deepseek. arXiv preprint arXiv:2505.17702 (2025)
27. Li, Y., Lin, C., Liu, Y., Long, X., Zhang, C., Wang, N., Li, X., Wang, W., Guo, X.: CADDreamer: Cad object generation from single-view images. In: CVPR. pp. 21448–21457 (2025)
28. Liang, J., He, F., Fan, R., Chu, Y., Yan, X.: CADCL: Reconstruct parametric cad models from b-rep via contrastive learning. Journal of Computational Design and Engineering **12**(10), 176–184 (2025)
29. Lin, D., Yuan, M., Wang, Z., Wu, T., Liu, Y.: FreeCAD: A multimodal framework for 3d cad model generation from free-form prompts. In: ACM MM. pp. 1948–1956 (2025)
30. Ma, W., Chen, S., Lou, Y., Li, X., Zhou, X.: Draw step by step: Reconstructing cad construction sequences from point clouds via multimodal diffusion. In: CVPR. pp. 27154–27163 (2024)
31. Mallis, D., Karadeniz, A.S., Cavada, S., Rukhovich, D., Foteinopoulou, N., Cherenkova, K., Kacem, A., Aouada, D.: CAD-Assistant: tool-augmented vllms as generic cad task solvers. In: ICCV. pp. 7284–7294 (2025)
32. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
33. Niu, K., Chen, Z., Yu, H., Chen, Y., Fu, T., Zhao, M., Li, B., Xue, X.: CReFT-CAD: Boosting orthographic projection reasoning for cad via reinforcement fine-tuning. arXiv preprint arXiv:2506.00568 (2025)
34. Niu, K., Yu, H., Chen, Z., Zhao, M., Fu, T., Li, B., Xue, X.: From intent to execution: Multimodal chain-of-thought reinforcement learning for precise cad code generation. arXiv preprint arXiv:2508.10118 (2025)
35. OpenAI: Introducing gpt 5.2 (2025), `https://openai.com/index/introducing-gpt-5-2`

36. Qi, D., Wang, C., Xu, J., Chu, T., Zhao, Z., Liu, W., Ding, W., Ma, Y., Gao, S.: Pointer-CAD: Unifying b-rep and command sequences via pointer-based edges & faces selection. In: CVPR. pp. 17377–17387 (2026)
37. Qin, F., Lu, S., Hou, J., Wang, C., Fang, M., Liu, L.: Drawing2CAD: Sequence-to-sequence learning for cad generation from vector drawings. In: ACM MM. pp. 10573–10582 (2025)
38. Rukhovich, D., Dupont, E., Mallis, D., Cherenkova, K., Kacem, A., Aouada, D.: CAD-Recode: Reverse engineering cad code from point clouds. In: ICCV. pp. 9801–9811 (2025)
39. Schüpbach, A., San Miguel, R., Ferchow, J., Meboldt, M.: From text to design: a framework to leverage llm agents for automated cad generation. Proceedings of the Design Society **5**, 1893–1902 (2025)
40. Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., Liu, Y.: RoFormer: Enhanced transformer with rotary position embedding. Neurocomputing **568**, 127063 (2024)
41. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020)
42. Team, K., Bai, T., Bai, Y., Bao, Y., Cai, S., Cao, Y., Charles, Y., Che, H., Chen, C., Chen, G., et al.: Kimi K2. 5: Visual agentic intelligence. arXiv preprint arXiv:2602.02276 (2026)
43. Wang, C., Ye, J., Wang, W., Gao, C., Feng, F., He, X.: RecAD: Towards a unified library for recommender attack and defense. In: Proceedings of the 17th ACM Conference on Recommender Systems. pp. 234–244 (2023)
44. Wang, S., Chen, C., Le, X., Xu, Q., Xu, L., Zhang, Y., Yang, J.: CAD-GPT: Synthesising cad construction sequence with spatial reasoning-enhanced multimodal llms. In: AAAI. vol. 39, pp. 7880–7888 (2025)
45. Willis, K.D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J.G., Solar-Lezama, A., Matusik, W.: Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. ACM TOG **40**(4), 1–24 (2021)
46. Wu, R., Xiao, C., Zheng, C.: DeepCAD: A deep generative network for computer-aided design models. In: ICCV. pp. 6772–6782 (2021)
47. Wu, S., Khasahmadi, A.H., Katz, M., Jayaraman, P.K., Pu, Y., Willis, K., Liu, B.: CadVLM: Bridging language and vision in the generation of parametric cad sketches. In: ECCV. pp. 368–384. Springer (2024)
48. Xie, H., Ju, F.: Text-to-CadQuery: A new paradigm for cad generation with scalable large model capabilities. arXiv preprint arXiv:2505.06507 (2025)
49. Xu, J., Wang, C., Zhao, Z., Liu, W., Ma, Y., Gao, S.: CAD-MLLM: Unifying multimodality-conditioned cad generation with mllm. arXiv preprint arXiv:2411.04954 (2024)
50. Xu, X., Jayaraman, P.K., Lambourne, J.G., Willis, K.D., Furukawa, Y.: Hierarchical neural coding for controllable cad model generation. arXiv preprint arXiv:2307.00149 (2023)
51. Xu, X., Willis, K.D., Lambourne, J.G., Cheng, C.Y., Jayaraman, P.K., Furukawa, Y.: SkexGen: Autoregressive generation of cad construction sequences with disentangled codebooks. arXiv preprint arXiv:2207.04632 (2022)
52. Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report. arXiv preprint arXiv:2505.09388 (2025)
53. Yin, X., Lu, X., Shen, J., Ni, J., Li, H., Tong, R., Tang, M., Du, P.: RLCAD: Reinforcement learning training gym for revolution involved cad command sequence generation. Computer-Aided Design p. 104027 (2025)

54. You, Y., Uy, M.A., Han, J., Thomas, R., Zhang, H., Du, Y., Chen, H., Engelmann, F., You, S., Guibas, L.: Img2CAD: Reverse engineering 3d cad models from images through vlm-assisted conditional factorization. In: Proceedings of the SIGGRAPH Asia 2025 Conference. pp. 1–12 (2025)
55. Yu, N., Ferdous Alam, M., Hart, A.J., Ahmed, F.: GenCAD-three-dimensional: Computer-aided design program generation using multimodal latent space alignment and synthetic dataset balancing. Journal of Mechanical Design **148**(3), 031703 (2026)
56. Yuan, Z., Shi, J., Huang, Y.: OpenECAD: An efficient visual language model for editable 3d-cad design. Computers & Graphics **124**, 104048 (2024)
57. Zhang, A., Jia, W., Zou, Q., Feng, Y., Wei, X., Zhang, Y.: Diffusion-CAD: Controllable diffusion model for generating computer-aided design models. IEEE TVCG **31**(12), 10188–10199 (2025)
58. Zhang, C., Polette, A., Pinquié, R., Carasi, G., De Charnace, H., Pernot, J.P.: eCAD-Net: Editable parametric cad models reconstruction from dumb b-rep models using deep neural networks. Computer-Aided Design **178**, 103806 (2025)
59. Zheng, Q., Xia, X., Zou, X., Dong, Y., Wang, S., Xue, Y., Shen, L., Wang, Z., Wang, A., Li, Y., et al.: CodeGeeX: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In: Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining. pp. 5673–5684 (2023)
60. Zheng, W., Sun, C., Zhang, W., Lv, J., Liu, X.: Target-guided bayesian flow networks for quantitatively constrained cad generation. In: ACM MM. pp. 3330–3339 (2025)
61. Zhou, J., Camba, J.D., Company, P.: CADialogue: A multimodal llm-powered conversational assistant for intuitive parametric cad modeling. Computer-Aided Design p. 104006 (2025)