Proceedings of the ASME 2025
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE 2025
August 17-20, 2025, Anaheim, CA, USA

IDETC/CIE 2025-168790

TRANSFORMCAD: MULTIMODAL TRANSFORMER FOR COMPUTER-AIDED DESIGN GENERATION

Yuewan Sun, Zhenghui Sha*
Walker Department of Mechanical Engineering
University of Texas at Austin
Austin, Texas 78712
Email: zsha@austin.utexas.edu

ABSTRACT

The creation of manufacturable and modifiable 3D shapes using Computer-Aided Design (CAD) remains a predominantly manual and time-consuming process, hindered by the complexity of boundary representations in 3D solids and the lack of intuitive design tools. This paper introduces TransformCAD, a CAD generation model that leverages both image and natural language descriptions as input to generate CAD sequences, producing editable 3D representations relevant to engineering design. TransformCAD incorporates a fine-tuned Contrastive Language-Image Pre-Training (CLIP) model to process multimodal input and employs two prediction branches—sketch and extrude—to enhance the parsing rate of CAD generation. Extensive evaluations demonstrate that TransformCAD outperforms existing models in terms of parsing rate, Chamfer distance, minimum matching distance, and Jensen-Shannon divergence. Furthermore, by analyzing the impact of training data, we show that TransformCAD exhibits strong potential for accurately generating long-sequence CAD models, which correspond to higher-complexity designs. Moreover, real-world 3D object images taken by a smartphone are used to validate TransformCAD's practicability, demonstrating its effectiveness in industrial applications. To the best of our knowledge, this is the first attempt at generating 3D CAD models integrating both image and natural language input. TransformCAD expands the boundaries of automated CAD modeling, enabling a more flexible and intuitive design process that bridges visual perception and structured command-based representations.

Keywords: Computer-Aided Design, Generative Design, Generative AI, CAD Sequence, Transformer

1 Introduction

Computer-aided design (CAD) systems are extensively used for 3D shape creation across various industrial sectors. However, constructing a parametric CAD model requires specialized expertise and is often a manual, time-consuming process. In mechanical design, particularly during the early stages, there is increasing interest in training computational agents to assist or collaborate with designers. These agents have the potential to enhance creativity, accelerate CAD modeling, and ultimately reduce design time and costs. Despite these advantages, developing a computational agent with the capabilities of a professional designer remains an extremely challenging machine-learning task.

Recent research has explored data-driven approaches for CAD generation, such as converting 3D point clouds [1, 2] or voxels [3, 4] into CAD models. However, these studies primarily focus on *reconstructing* 3D CAD models, so the methods are not generative in nature. A generative model can create diverse and novel CAD designs beyond reconstruction, enabling design exploration, customization, and optimization rather than replicating existing geometries. Moreover, those methods can-

*Corresponding author: zsha@austin.utexas.edu

not generate CAD sequences – an ordered series of modeling operations, such as drawing 2D sketches and extruding them into 3D shapes that are executed within CAD software to construct a 3D model [5]. Unlike 3D models, a CAD sequence provides insight into the historical construction process and embedded design knowledge, enabling efficient geometry modifications through parametric modeling.

To address this limitation, methods such as DeepCad [6] and SkexGen [7] have been developed for the unconditional generation of CAD sequences. However, these models generate CAD sequences without specific user input, meaning they lack interactive control and adaptation to user intent. This limits human-AI design collaboration, which is particularly important in the early design stage, where design intent and requirements are expected to continuously feed into the loop of AI-generated design content, so the design can be iterated and adjusted based on human feedback.

This paper advances the state of the art by introducing a multimodal generative model that takes advantage of image and natural language inputs for CAD sequence generation. Compared to point-cloud or voxel-based approaches, image and text data are more intuitive and accessible, thus enhancing user-interactive experience and control for better design interaction and space exploration. The generated CAD sequence can be easily converted into B-rep, STEP, and STL formats and rendered into a CAD model, as illustrated in Figure 1.

To comprehensively evaluate our model, we employed seven metrics and compared its performance with other image-to-CAD sequence models. The results show that TransformCAD outperforms existing models in terms of Parsing Rate, Chamfer Distance, Minimum Matching Distance, and Jensen-Shannon Divergence. Furthermore, by analyzing the impact of training data, we demonstrate that TransformCAD exhibits strong potential for accurately generating long-sequence CAD models, which correspond to higher-complexity designs. The contributions of the proposed model are summarized as follows.

- To the best of our knowledge, this study is the first attempt to predict a sequence of CAD operations given a combination of images and natural language inputs.
- We develop a multimodal CAD generation pipeline that leverages a fine-tuned CLIP model for processing image and text embeddings, along with Llama 3.2 [8] for generating structured text descriptions. This integration enhances the model's ability to interpret diverse input modalities, providing users with greater control over CAD generation.
- We integrate a two-branch model architecture with image and text inputs, enabling more effective latent space learning from multimodal data. This structured approach enhances the model's ability to capture distinct design operations, improving the precision and interpretability of CAD sequence generation.

This paper is organized as follows: Section 2 provides an overview of the background on CAD generation using Large Language Models (LLMs) and Deep Neural Networks (DNNs). Section 3 outlines the methodology for the development of TransformCAD, covering CAD sequence representation and model architecture. Section 4 details the experimental setup and presents analyses of quantitative and qualitative results. Section 5 discusses the research findings, limitations, and potential directions for future research. Finally, Section 6 summarizes the key insights and conclusions.

2 Literature Review

For CAD generation models, we can classify existing approaches into two categories: large language models (LLMs)-based models and deep neural network-based models. In this section, we first review CAD generation methods based on LLMs, which is currently a highly active research topic. Following this, we introduce CAD generation methods based on deep neural networks, which is the approach adopted in this study.

2.1 CAD Generation using Large Language Models

With the rapid development of Large Language Models (LLMs), researchers have begun exploring their potential for CAD generation. For multimodal input, LLM4CAD [9] first developed a dataset of mechanical components, collecting human-written descriptions for each component. Additionally, LLM4CAD introduced a framework utilizing GPT-4V to process both image and text inputs, while also employing GPT-4 as a CAD program debugger. For text-only input, LLM4CAD [10] fine-tuned GPT-3.5 based on that dataset to generate Python programs capable of constructing the corresponding CAD models. However, collecting human-written text descriptions is costly, and due to the inherent limitations of text descriptions in capturing geometric details, accurately describing complex geometries remains a significant challenge.

Another example is Query2CAD [11], which proposed a framework combining GPT-turbo and BLIP2 to generate executable CAD macros from user queries. However, the test set only included 57 curated user queries, and without fine-tuning, the success rate of generating correct CAD models on the first attempt was only 53.6%. To improve generation quality, the framework relies heavily on human refinement prompts — a process that is both labor-intensive and time-consuming.

With the emergence of Multimodal Large Language Models (MLLMs), researchers have further explored using images and sketches as inputs for CAD generation. For example, CAD-Assistant [12] employed GPT-4o as the Vision Large Language Model (VLLM) to recognize hand-drawn sketches and automatically extrude them into 3D CAD models. However, the reported generation accuracy was below 75%, and the generation process

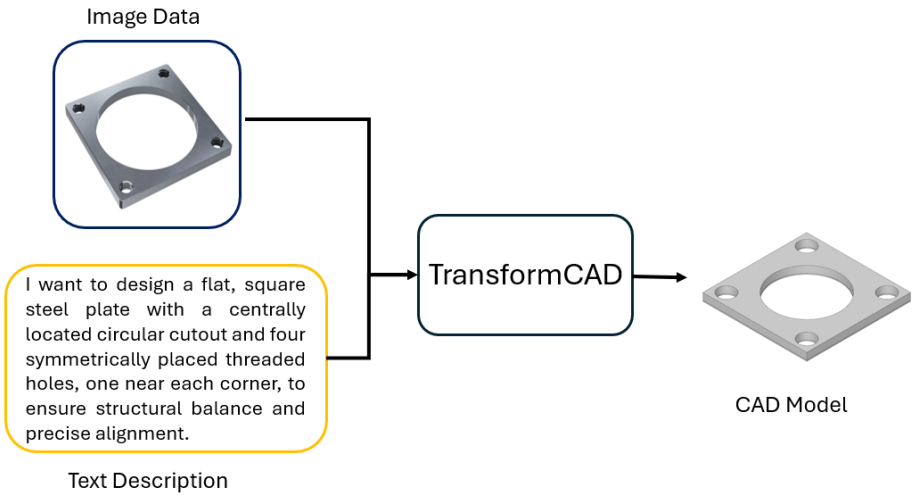

FIGURE 1. The overview of the proposed model function.

was highly sensitive to the wording of the prompts.

Beyond images and sketches, other modalities have also been considered. For instance, CAD-MLLM [13] introduced a unified CAD generation system capable of processing inputs in the form of textual descriptions, images, point clouds, or combinations of them all. Interestingly, their results showed that training the model using only point cloud data led to better performance than training on multiple modalities combined. This finding highlights a core challenge of MLLM-based CAD generation — effectively balancing contributions from multiple modalities is difficult, especially when the quality and information density of each modality vary significantly. This imbalance often introduces noise and reduces overall generation accuracy, particularly for complex geometries.

In summary, there are several limitations associated with using LLMs and MLLMs for CAD generation:

- **Limited descriptive power of text input**: Text descriptions alone are insufficient to capture complex geometric details, making text-only generation prone to inaccuracies.
- **Prompt sensitivity**: Both LLMs and MLLMs are highly sensitive to prompt variations, meaning they may generate different outputs even when given the same input — an undesirable property for industrial CAD applications that require consistency and repeatability.
- **Heavy reliance on human intervention**: Existing frameworks have challenges of generative CAD models with sufficient accuracy, depending on iterative human refinement, which increases both time and cost.
- **Lack of geometric understanding**: LLMs, by their nature, struggle with understanding geometric and topological relationships. To improve accuracy, fine-tuning is necessary — but collecting fine-tuning data is costly and resource-intensive, especially since these models often generate Python programs rather than direct CAD commands, adding complexity due to dependencies on external libraries, error handling requirements, and strict syntax rules.

2.2 CAD Generation using Deep Neural Networks (DNNs)

The other major category of CAD generation relies on Deep Neural Networks (DNNs) to train task-specific CAD generation models based on the input modality. An important consideration when designing these models is the design representation of CAD models for machine learning. To this end, several CAD representations have been explored.

One of the most common representations is the Boundary Representation (B-rep) format, a standard format for representing 3D shapes in CAD. B-rep defines objects based on their boundary surfaces, edges, and vertices, with specific topology relationships among these elements. Numerous machine learning-based approaches [18, 19] have emerged that aim to directly generate parametric curves and surfaces in the B-rep format. In parallel, several studies, including those by Kim et al. [20], Budroni et al. [21], and Liu et al. [22], have explored generating CAD models using 3D point clouds, enabling direct learning from scanned data.

Another popular representation is *Sketch-and-Extrude* operation sequences, which aligns closely with how CAD models are constructed in modern parametric modeling software. DeepCAD, proposed by Wu et al. [6], was the first approach to formalize CAD modeling as the generation of command sequences. DeepCAD used a transformer-based auto-encoder trained on a large dataset of 3D object models collected via the publicly available Onshape API. Each command was represented as a fixed-

TABLE 1. Summary of Modality-Specific CAD Generation Models

Study Name	Input Modality	Dataset	CAD Sequence Representation	Key Model Architecture
DeepCAD [6]	Unconditional	DeepCAD	Stacked Vector	Transformer Encoder & Decoder
SkexGen [7]	Unconditional	DeepCAD	Hierarchy Sequence	Transformer & VQ-VAE
GenCAD [14]	Multiple Images	DeepCAD	Stacked Vector	Transformer & Diffusion Model
ARE-Net [15]	Multiple Images	DeepCAD	Stacked Vector	Transformer Decoder & CNN
Image2CADSeq [16]	Single Image	Fusion 360 Gallery	Stacked Vector	Transformer-Based VAE
CADGen [17]	Unconditional	DeepCAD	Hierarchy Sequence	Transformer & VQ-VAE
TransformCAD	Single Image & Text Description	DeepCAD	Hierarchy Sequence	Transformer & VQ-VAE

length 16-dimensional vector, with different elements storing different types of parameters. These vectors were then stacked to form a matrix representing the full CAD sequence.

To enhance the expressiveness and learnability of this representation, Skexgen introduced a more structured approach, defining the sketch-and-extrude sequence as a hierarchy of primitives. This hierarchical representation separates the generation of topology (which entities exist and how they are created and related) from the generation of geometry (numerical values such as dimensions and positions). This hierarchical design significantly improves the successful parsing ratio, enhancing the ability to accurately reconstruct the 3D CAD model from the generated sequence [7].

Building on these two Sketch-and-Extrude sequence representations (fixed vector stacking and hierarchical primitives), several CAD generation models have been developed to accommodate different input modalities. ARE-Net [15] adopts the stacked vector method and combines a CNN encoder with a transformer decoder to generate CAD models directly from multi-view image inputs. GenCAD [14] leverages a diffusion model combined with a transformer encoder-decoder architecture to achieve CAD generation from multiple image views. Image2CADSeq [16] also relies on the stacked vector representation, embedding it within a transformer-based variational autoencoder (VAE) and training the model using the Fusion 360 Gallery dataset with single-image input. In contrast, CADGen [17] adopts the hierarchical operation sequence representation and focuses on unconditional CAD generation, employing a transformer model combined with a Vector Quantised Variational AutoEncoder (VQ-VAE) to learn a discrete latent space suitable for operation sequences. Each of these models demonstrates the flexibility of Sketch-and-Extrude representations across different modalities, from single- and multi-view images to purely latent space exploration.

Table 1 summarizes the input modality, dataset, sequence representation, and model type for the studies using *Sketch-and-Extrude* operation sequences mentioned above. Based on this review, it is evident that most Sketch-and-Extrude-based CAD generation models rely heavily on transformer architectures. However, no existing study has explored CAD generation from combined image and text inputs. To address this gap, our proposed model, TransformCAD, introduces a new approach that takes advantage of both image and natural language descriptions of a design as input to generate CAD sequence. TransformCAD adopts the hierarchical operation sequence representation and employs a Transformer architecture combined with the VQ-VAE codebook [23], allowing multi-modal CAD generation.

3 Methodology

3.1 Design Representation

In this study, we adopt the Sketch-and-Extrude Construction Sequence from Skexgen [7], a structured format that effectively encodes CAD modeling operations. This representation employs five special tokens to capture the topology of a CAD model: 1) A *topology token* to indicate the curve type (line/arc/circle); 2) a *geometry token* containing point coordinates; 3) an *end-primitive token* marking the completion of a primitive (curve, face, sketch, extruded); 4) an *extrusion token* containing extrusion parameters and Boolean operations; and 5) an *end-sequence token* indicating the termination of the sequence. An example is shown in Figure 2 to illustrate the usage of tokens and the hierarchical structure.

This structured sequence-based representation offers several advantages. First, it provides a hierarchical abstraction of the design process, aligning well with how designers construct CAD models step-by-step. Second, it ensures that both geometric and

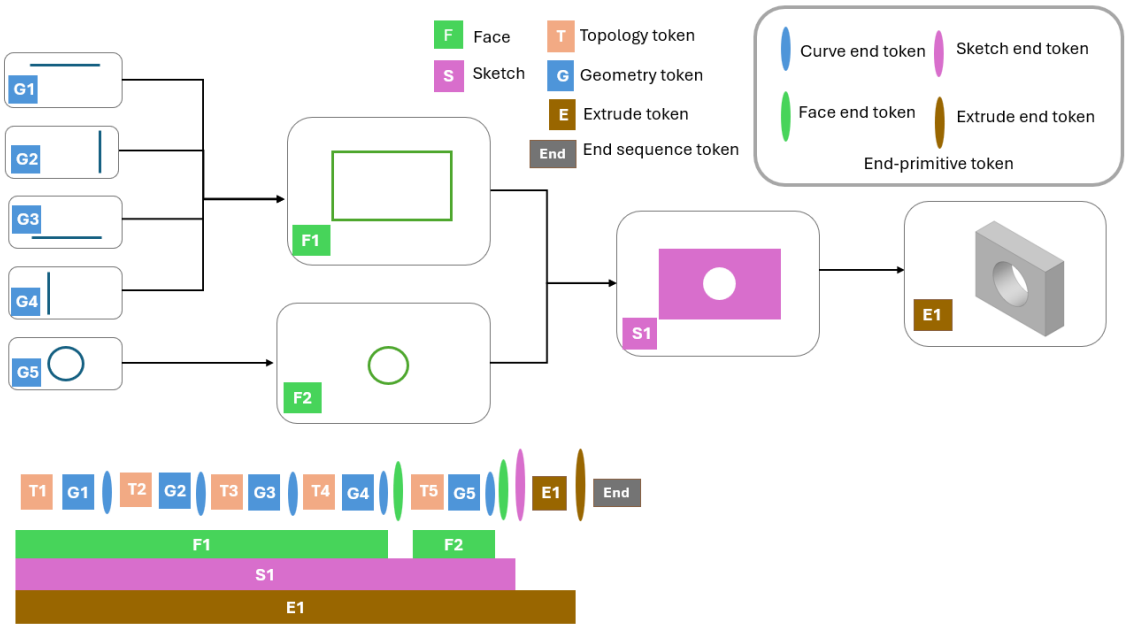

FIGURE 2. An example CAD model is represented as a hierarchy structure and a CAD operation sequence, with five types of tokens introduced. The end-primitive token includes four types (curve end token, face end token, sketch end token and extrude end token). Each curve begins with a topology token (T), followed by a geometry token (G), and ends with a curve end token. A face end token indicates the completion of a face, while a sketch end token marks the end of a sketch, which consists of two faces in this example. An extrude token (E) follows to extrude the sketch, concluding with an extrude end token. The sequence terminates with an end sequence token (End). Notably, faces and sketches are not explicitly represented as tokens but are inferred from the end-primitive tokens.

topological information are explicitly captured, enabling the machine learning model to understand both shape formation and spatial relationships. Third, the sequence format is well-suited for neural sequence models, allowing us to leverage powerful architectures such as transformers and sequence-to-sequence models.

3.2 Dataset

We use the DeepCAD dataset [6], which comprises 212,323 CAD construction sequences. To ensure data quality, we perform duplicate removal, eliminating redundant sketch and extrude subsequences. After preprocessing, the dataset contains 202,123 unique CAD models, maintaining the original 90% training, 5% validation, and 5% test split.

To incorporate multimodal learning, we first render an image for each CAD model in the preprocessed dataset and generate a corresponding text description using the Llama 3.2 multimodal large language model [8]. Each CAD model is then paired with its rendered image and text description. To obtain structured embeddings, we apply the CLIP model to extract both image and text embeddings.

To enhance CLIP's ability to understand CAD images, we fine-tune the model using a contrastive learning approach on the previously mentioned image-text pair dataset. After fine-tuning, we evaluated the model performance in a text-to-image retrieval task. Specifically, we randomly select 500 image-text pairs from the test set, and for each text query, rank the retrieved images based on cosine similarity. We then compute Recall@1, Recall@5, and Recall@10, where Recall@K measures the percentage of queries for which the correct result appears within the top-K predictions. Specifically, Recall@1 evaluates whether the correct result is ranked first, Recall@5 checks if it appears within the top-5 results, and Recall@10 assesses its presence among the top 10. Additionally, we calculate the average cosine similarity to assess retrieval accuracy. The results, presented in Table 2, demonstrate that the fine-tuned CLIP model significantly improves its understanding of CAD images and their textual descriptions. Following this, we use the fine-tuned CLIP model to generate image and text embeddings for each CAD model. In summary, each CAD model is represented by an image embedding and a text embedding, which serve as input to our framework, while the corresponding CAD sequence representation acts as the ground-truth output.

Model	Recall@1 ↑	Recall@5 ↑	Recall@10 ↑	Cos. Sim. ↑
CLIP	0.1441	0.3932	0.5636	0.37
Fine-tuned CLIP	0.6799	0.9480	0.9480	0.46

TABLE 2. Comparison of Recall and Cosine Similarity for CLIP and Fine-tuned CLIP.

3.3 Training Datasets

Since sequence length can influence model performance, generalization, and computational efficiency, understanding this relationship helps optimize training data and improve the model's ability to generate accurate and diverse CAD sequences. Therefore, in this study, we investigate the effect of CAD sequence length distribution in the training dataset. Here, sequence length refers to the number of tokens required to represent a CAD model using the construction sequence described in Section 3.1. To facilitate this investigation, we utilize two training datasets in our experiments. Training Dataset 1 is the original dataset obtained through the preprocessing method outlined earlier in Section 3.2. The sequence length distribution of this dataset is shown in Figure 3, where we also present a representative CAD model for each sequence length range along the X-axis to illustrate how sequence length correlates with a model's geometric complexity.

However, as shown in Figure 3, Training Dataset 1 exhibits a strong imbalance, with a disproportionately large number of short sequences compared to long sequences. To address this data skew issue, we apply a truncation process: for each sequence length bin below 42, we randomly sample 3000 CAD models. The resulting sequence length distribution after truncation is shown in Figure 4. The total amount of Training Dataset 2 is 95,997, which is 45.2% of Training Dataset 1.

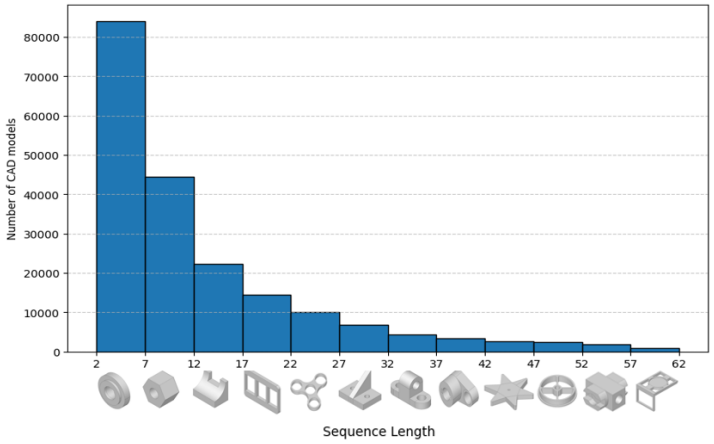

FIGURE 3. The CAD Sequence Length Distribution for Training Dataset1

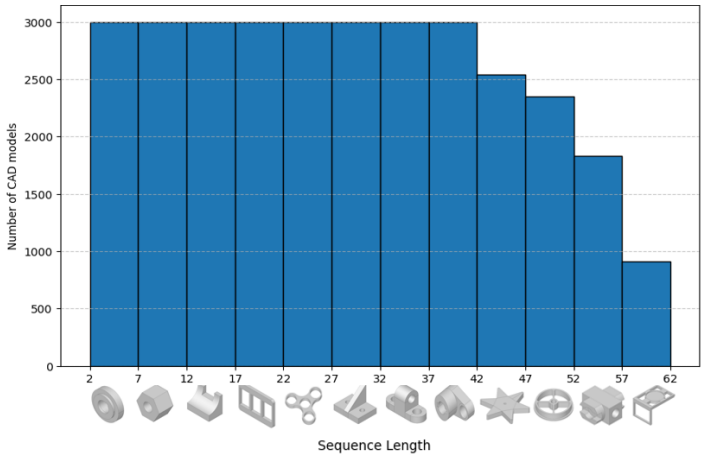

FIGURE 4. The CAD Sequence Length Distribution for Training Dataset2

3.4 Model Architecture

The model architecture of TransformCAD consists of two branches, as shown in Figure 5: the sketch branch and the extrusion branch. The unique two-branch design of model architecture separates the operational information of sketching and extrusion, making the machine learning process easier to capture the latent space of each operation sequence.

For the sketch branch, the model includes a linear image encoder and a linear text encoder, which map the 512-dimensional image and text embeddings to 256-dimensional latent representations. These two latent representations are then concatenated and passed through a Transformer-based topology encoder and a geometry encoder, producing two latent vectors representing the sketch. After concatenation, the resulting representation is fed into a Transformer-based sketch decoder, which generates a sketch sequence from the latent representation. Here, the topology encoder is included to extract explicit topological information, aiming to enhance the understanding of a sketch for the sketch decoder. Since the Extrusion branch predicts the extrude length and transformation matrix, totaling 10 parameters, it uses a single extrude encoder and decoder instead of separate topology and geometry encoders in the Extrusion branch.

The topology encoder processes the image and text latent representations and follows the standard Transformer architecture [24]. It is a Transformer module with four layers, where each layer consists of eight attention heads, pre-layer normalization, an input dimension of 256, and a feed-forward dimension of 512. The encoder outputs a Vector Quantised Variational AutoEncoder (VQ-VAE) codebook [23], containing 500 quantized code tokens, from which it selects the four closest code tokens.

The geometry encoder and extrude encoder share the same architecture as the topology encoder but differ in codebook size and the number of selected tokens. The geometry encoder has a

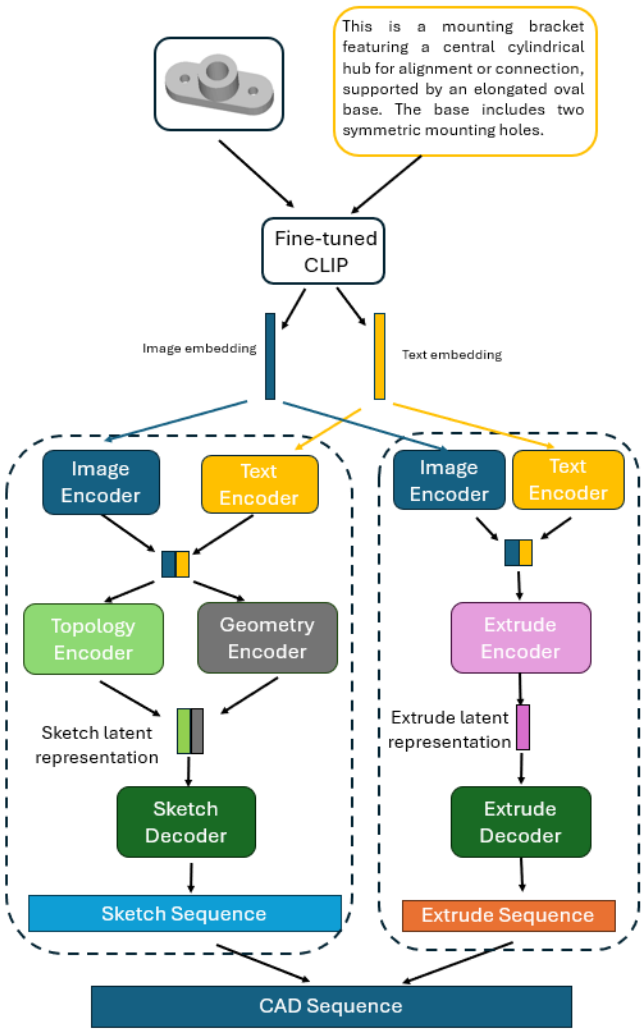

FIGURE 5. The architecture of the model.

codebook size of 1000 and selects the two closest tokens, while the extrude encoder has the same codebook size but selects ten closest code tokens. This difference arises from the nature of the information each encoder processes. The geometry encoder requires only two tokens because geometric features are highly localized, and a small number of tokens is sufficient to capture relevant details. In contrast, the extrude encoder selects ten tokens because extrusion operations involve more complex transformations that depend on a broader spatial and structural context. The sketch decoder takes the topology and geometry codebooks as input and generates geometry and end-primitive tokens (for curves, loops, faces, and sketches) to reconstruct the sketch subsequence. Notably, topology tokens are not generated explicitly, as they can be inferred from the number of geometry tokens in each curve (i.e., lines/arcs/circles correspond to 1, 2, and 4 tokens, respectively). The extrude decoder follows the same design as the sketch decoder.

3.5 Loss Function

The topology encoder, geometry encoder, and sketch decoder are jointly trained with three loss functions:

$$\begin{aligned} loss = &\sum_{K} \text{CrossEntropy}(h_K^{out}, h_K^{gt}) \\ &+ \left\| \text{sg}(Z_{tp}^e) - \mathbf{b}_{tp} \right\|_2^2 + \beta \left\| Z_{tp}^e - \text{sg}(\mathbf{b}_{tp}) \right\|_2^2 \\ &+ \left\| \text{sg}(Z_{ge}^e) - \mathbf{b}_{ge} \right\|_2^2 + \beta \left\| Z_{ge}^e - \text{sg}(\mathbf{b}_{ge}) \right\|_2^2 \end{aligned} \tag{1}$$

The first line of Equation (1) computes the sequence reconstruction loss, where h_K^{out} denotes the predicted probability dis-

tribution from the sketch decoder, and h_K^{gt} represents the ground-truth one-hot vector. The second and third lines correspond to the standard codebook loss and commitment loss used in VQ-VAE [23]. The function $sg(\cdot)$ is the stop-gradient operation, which behaves as the identity function during the forward pass but blocks gradient flow during the backward pass. The weight β scales the commitment loss, and is set to 0.25, ensuring that the encoder output commits to a single code vector.

4 Experiments and Results

4.1 Experiment Setting

During training, we first pre-train the encoder and decoder—excluding the image and text encoders—of each branch, following the architecture of the SkexGen model [7]. This stage uses CAD sequences from oringal DeepCAD dataset, allowing the encoder and decoder to learn the underlying patterns of CAD sequence generation. The model is initially trained for 300 epochs with a batch size of 128. After pre-training, we introduce the image and text encoders, using image and text embeddings for each dataset, as introduced in Section 3.3 and fine-tune the entire model for an additional 300 epochs while maintaining the same batch size. This process, when using two different training datasets, results in two model variants: **TransformCAD1** and **TransformCAD2**.

The model is implemented in PyTorch and trained on an RTX 5000 Ada GPU. We use a dropout rate of 0.1 and optimize the model with the Adam optimizer using a learning rate of 0.001.

4.2 Metrics

To evaluate the accuracy of 3D model generation and compare it with other conditional CAD generation models, we use the following seven metrics to assess model performance.These metrics collectively provide a comprehensive evaluation of the syntactic, geometric, and distributional accuracy of the generated 3D CAD models, categorized into CAD sequence-related metrics, which assess the correctness and structure of the generated sequences, and 3D model-related metrics, which evaluate the geometric fidelity and overall reconstruction quality.

4.2.1 CAD Sequence-Related Metrics

1. **Parsing Rate (PR)**: This metric measures the percentage of output CAD sequences that can be successfully rendered into a valid 3D model. A higher parsing rate indicates better syntactic correctness of the generated CAD sequences.
2. **Command Accuracy ($\mathbf{ACC_{cmd}}$)**: The command accuracy ACC_{cmd} measures the agreement of the predicted CAD command type $\hat{t}_i$ with the ground truth command type t_i for a CAD ground-truth sequence of N_t steps:

$$ACC_{\text{cmd}} = \frac{1}{N_t}\sum_{i=1}^{N_t}(t_i == \hat{t}_i) \tag{2}$$

3. **Parameter Accuracy ($\mathbf{ACC_{param}}$)**: The parameter accuracy ACC_{param} quantifies the agreement of a predicted CAD parameter $\hat{p}_{i,j}$ in one command to its ground-truth counterpart $p_{i,j}$. Only correctly predicted commands CAD sequences were evaluated and a threshold of $\eta = 0.05$ was used.

$$ACC_{param} = \frac{1}{N_t}\sum_{i=1}^{N_t}\sum_{j=1}^{|\hat{p}_i|}(|p_{i,j} - \hat{p}_{i,j}| \leq \eta) \tag{3}$$

4.2.2 3D Model-Related Metrics

1. **Chamfer Distance (CD)**: For the geometric evaluation of 3D models, Chamfer Distance (CD) measures the shortest distance from a point x on the surface S_p of the generated (predicted) model to the nearest point y on the surface S_t of the ground-truth model. This process is performed bidirectionally, ensuring symmetry in the evaluation. In this study, we compute the Chamfer Distance using 2,000 surface points per model.

$$CD = \frac{1}{S_t}\sum_{x\in S_t}\min_{y\in S_g}\|x-y\|_2^2 + \frac{1}{S_p}\sum_{x\in S_p}\min_{y\in S_t}\|y-x\|_2^2 \tag{4}$$

2. **Coverage (COV)**: Coverage represents the proportion of the ground truth model accurately matched by the generated model. This is computed based on the closest Chamfer distance of 2,000 uniformly sampled points on the surface. For each point cloud in ground truth model, we first find its nearest neighbor in the generated models. Coverage is measured as the fraction of the point clouds in a generated model that was matched to the point clouds in the ground truth model. Closeness can be computed using Chamfer Distance.

$$COV = \frac{1}{|S_t|}\sum_{x\in S_t}\mathbf{1}\left(\min_{y\in S_p}\|x-y\|_2 \leq \tau\right) \tag{5}$$

, where $\mathbb{1}(\cdot)$ is an indicator function, which returns 1 if the condition inside is true, otherwise 0. τ is a threshold distance to determine if a point is "covered" by the generated model.
3. **Minimum Matching Distance (MMD)**: This metric calculates the average minimum distance between each generated model and its nearest neighbor in the ground truth model

set. A lower MMD value indicates that the generated models closely resemble the ground truth models, demonstrating higher fidelity.

$$MMD = \frac{1}{|S_p|} \sum_{x \in S_p} \min_{y \in S_t} \|x - y\|_2 \tag{6}$$

4. **Jensen-Shannon Divergence (JSD)**: JSD quantifies the similarity between the marginal point distributions of the ground truth and generated models. In a 3D dataset, the marginal point distribution represents the likelihood of each randomly sampled 3D point occurring on the surface of the model, independent of other points. A smaller JSD value implies better alignment of the overall distributions, indicating that the generated models capture the variability of the ground truth dataset.

$$JSD(S_t \parallel S_p) = \frac{1}{2} D(S_t \parallel M) + \frac{1}{2} D(S_p \parallel M) \tag{7}$$

, where $M = \frac{1}{2}(S_t + S_p)$ and $D(\cdot \parallel \cdot)$ the KL-divergence between the two distributions.

4.3 Command and Parameter Accuracy

Figure 6 illustrates the distribution of CAD sequence lengths in the testing dataset, which consists of a total of 8,044 samples. Since the testing set is randomly sampled from the original training dataset, its distribution closely resembles that of the training set, as shown in Figure 3. Among the testing data, 5,396 samples have a sequence length of 10 or less, accounting for 67.1% of the dataset.

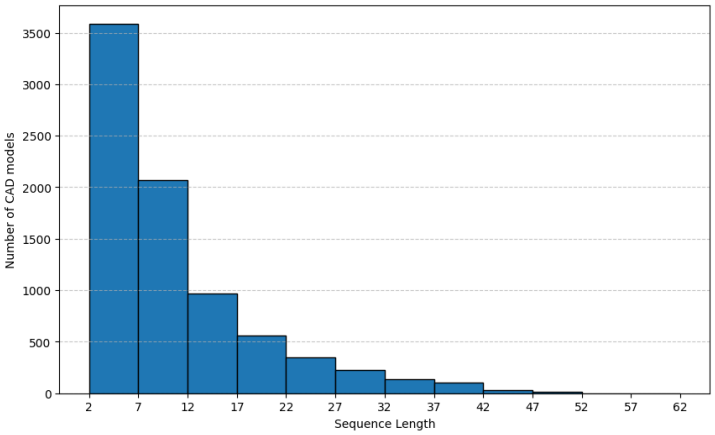

FIGURE 6. The CAD Sequence Length Distribution For Testing Data

Figure 7 presents the average Command Accuracy (ACC_{cmd}) of TransformCAD1 and TransformCAD2 across different CAD sequence lengths, including error bars. The dotted lines represent the exponential fits for each model, with R^2 values of 0.906 and 0.771 for TransformCAD1 and TransformCAD2, respectively. As observed in the figure, Command Accuracy decreases as the CAD sequence length increases for both models. For short sequences (length $\leq$ 10), both models achieve relatively high Command Accuracy (over 80%). However, as the sequence length increases—particularly for sequences longer than 30—the Command Accuracy drops to approximately 40% for TransformCAD1 and 50% for TransformCAD2.

Comparing the two models, TransformCAD2 outperforms TransformCAD1 for CAD sequences longer than 10. However, for short sequences, TransformCAD1 exhibits higher Command Accuracy than TransformCAD2. This is due to the truncation of short sequences during training, as shown in Figure 4, making TransformCAD2 have less training data of shorter sequences compared to that of TransformCAD1.

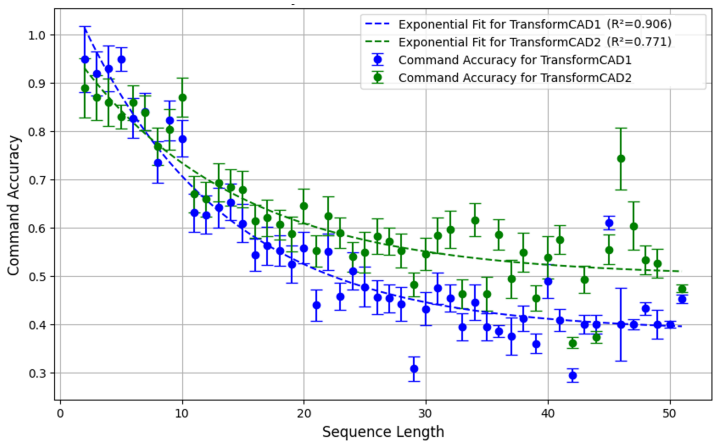

FIGURE 7. Command Accuracy (ACC_{cmd}) of TransformCAD1 and TransformCAD2 across different CAD sequence lengths.

Figure 8 illustrates the average Parameter Accuracy (ACC_{param}) of TransformCAD1 and TransformCAD2 across different CAD sequence lengths. Note that the x-axis of Parameter Accuracy is shorter than that of Command Accuracy because Parameter Accuracy is only computed for CAD data where Command Accuracy is perfectly predicted (i.e., $ACC_{cmd} = 100\%$). This is because the comparison of parameters only makes sense if the command types are the same. Similar to Figure 7, Figure 8 includes error bars for each sequence length, and the dotted lines indicate the exponential fits for each model, with R^2 values of 0.832 and 0.907 for TransformCAD1 and TransformCAD2, respectively.

As shown in the figure, Parameter Accuracy also decreases as the CAD sequence length increases for both models. For long sequences (length > 10), TransformCAD2 outperforms TransformCAD1, while for short sequences, both models perform similarly. Since the Geometry Encoder generates parameter information (see Figure 5) as a latent representation of parameters, this suggests that for short sequences, which involve fewer parameters to predict, the Geometry Encoder may not require a large amount of training data. However, with a greater proportion of long sequences, the Geometry Encoder has more opportunities to learn from long-sequence data, leading to improved performance.

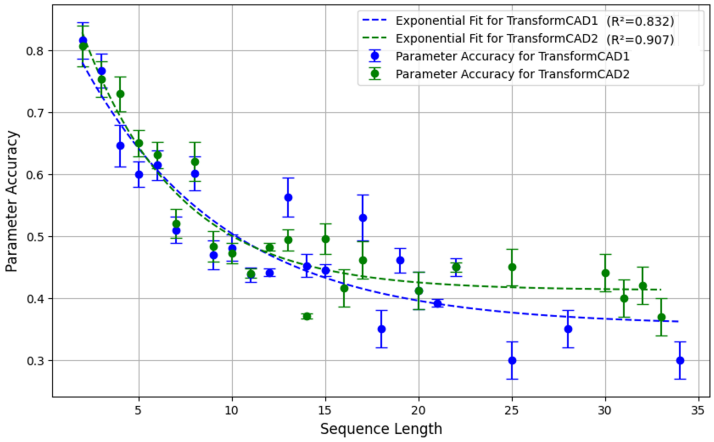

FIGURE 8. Parameter Accuracy (ACC_{param}) of TransformCAD1 and TransformCAD2 across different CAD sequence lengths.

4.4 Metrics Comparison

Since no prior CAD generation study has utilized a combination of a single image and text input, we compare Command Accuracy (ACCcmd), Parameter Accuracy (ACCparam), Parsing Rate (PR), and Chamfer Distance (CD) with ARE-Net [15], a CAD generation model based on multiple image inputs. ARE-Net also reports metric results for single-image input, allowing for direct comparison. Table 3 presents the metric comparison between ARE-Net and TransformCAD.

For Command Accuracy and Parameter Accuracy, ARE-Net with 20-image input outperforms all other models. This is expected, as multiple image inputs provide richer geometric information to the model. However, for single-image input, both TransformCAD models perform slightly worse than ARE-Net. It is important to note that even when ARE-Net is tested with a single image input, it was trained on 10 images per CAD model, whereas TransformCAD was trained exclusively on single-image inputs, which has less geometric information during training.

For Parsing Rate, TransformCAD models outperform ARE-Net models, with both TransformCAD models achieving a Parsing Rate above 90%. This improvement is attributed to the hierarchical representation of CAD models in TransformCAD. Its dual-branch architecture allows the model to process sketch and extrude information separately and then merge together during inference, thereby ensuring a higher success rate in CAD sequence generation.

For Chamfer Distance, TransformCAD also outperforms ARE-Net, indicating that even though TransformCAD achieves lower Command Accuracy and Parameter Accuracy, its generated CAD models closely match the ground truth in terms of geometric shape. Command Accuracy and Parameter Accuracy are stricter evaluation metrics than Chamfer Distance, as they assess correctness at the command and parameter levels rather than overall shape similarity.

For Coverage (COV), Minimum Matching Distance (MMD), and Jensen-Shannon Divergence (JSD), we compare TransformCAD with DeepCAD [6], Skexgen [7], and GenCAD [14]. We select these models for comparison because they reported these metrics and were trained on the same DeepCAD dataset, ensuring a fair and consistent evaluation. Table 4 presents the results for these metrics.

In terms of Coverage (COV), GenCAD slightly outperforms TransformCAD models. However, TransformCAD still achieves competitive coverage, indicating that it effectively generates a variety of CAD models. For MMD and JSD, TransformCAD significantly outperforms the other three models. A lower MMD suggests that TransformCAD produces CAD models that are structurally closer to real CAD models in the dataset, reducing deviations in feature space. Similarly, a lower JSD indicates that the distribution of generated CAD models closely aligns with the distribution of ground-truth models, reflecting higher fidelity and diversity in TransformCAD's output.

These results align with the Chamfer Distance (CD) results, further reinforcing that TransformCAD generates CAD models with a shape that closely matches the ground truth. The superior performance in MMD and JSD highlights TransformCAD's ability to learn and reconstruct geometric details, making it a more effective model for high-fidelity CAD generation.

4.5 Qualitative Results

To qualitatively evaluate our model's CAD generation capabilities, Figure 9 presents a gallery of correctly generated models categorized by sequence length. Here, "correct generation" is defined as cases where commands are perfectly predicted and the absolute parameter error remains within 5% of the ground truth values, given the corresponding image and text inputs.

The figure demonstrates that our model can successfully generate CAD models involving multiple sequential operations with high precision. Notably, it includes practical mechanical

	$\text{ACC}_{\text{cmd}} \uparrow$	$\text{ACC}_{\text{param}} \uparrow$	PR ↑	CD ↓
ARE-Net (20 Images) [15]	**92.83%**	**78.8%**	81.6%	1.75×10^3
ARE-Net (Single Image) [15]	88.0%	65.0%	81.6%	1.75×10^3
TransformCAD1	78.3%	60.5%	**95.7%**	$\mathbf{0.71 \times 10^3}$
TransformCAD2	76.2%	60.8%	94.6%	0.76×10^3

TABLE 3. Comparison of models based on Command Accuracy (ACC_{cmd}), Parameter Accuracy ($\text{ACC}_{\text{param}}$), Parsing Rate (PR) and Chamfer Distance (CD).

	COV ↑	MMD ↓	JSD ↓
DeepCAD [6]	76.8	1.68	2.01
Skexgen [7]	74.3	1.48	0.81
GenCAD [14]	**81.37**	1.38	3.49
TransformCAD1	79.2	**1.08**	**0.49**
TransformCAD2	78.3	1.16	0.52

TABLE 4. Comparison of models based on Coverage (COV), Minimum Matching Distance (MMD), and Jensen-Shannon Divergence (JSD).

components, such as nuts and pillow blocks, suggesting that our model possesses significant potential for industrial applications.

5 Discussion

5.1 Influence of Training Dataset

This study utilized two different training datasets to train the TransformCAD models and examine the influence of the training dataset on their performance. As shown in Figure 7 and Figure 8, TransformCAD2, which was trained on Training Dataset2, outperforms TransformCAD1 in generating long sequences. This suggests that increasing the proportion (without even increasing the quantity) of long sequences in the training dataset can enhance the model's ability to generate long-sequence CAD models, which are typically more complex.

However, as shown in Table 3 and Table 4, TransformCAD2 exhibits slightly lower average performance across all metrics compared to TransformCAD1, though the difference is minimal. This may be attributed to the reduced proportion of short sequences in Training Dataset2, leading to lower performance on short-sequence data. Meanwhile, the test dataset contains a relatively high proportion of short sequences, as illustrated in Figure 6, further contributing to the observed performance gap.

It is important to note that Training Dataset 2 contains only 45.2% of the data volume of Training Dataset 1, as reported in Section 3.3. Despite this, TransformCAD2 was trained with the same number of epochs, indicating that it is more efficient in learning from the available training data. Furthermore, if provided with a higher proportion of long-sequence data, the TransformCAD model demonstrates the potential to generate complex long-sequence CAD models more accurately.

5.2 Validation using Real-World Product Image

To evaluate the performance of the TransformCAD model in real-world applications, we designed an experiment where users take photos of actual mechanical components and instantly convert them into CAD sequences. Six CAD models were selected for this study, as listed in Figure 10. These physical objects were 3D printed and captured using a smartphone, with the background removed before being fed into the TransformCAD model, as shown in the first column of Figure 10. The second column presents the corresponding text descriptions for each CAD model. These real-world images and text descriptions were then fed into the TransformCAD model to generate CAD sequences. The third row of Figure 10 displays the rendered images of the resulting 3D CAD models.

The Parsing Rate for these six real-world images was 100%. In terms of accuracy, the first two generated models successfully captured the topology of the input, with all commands correctly identified; however, the proportions of the parameters were inaccurate. This discrepancy may be attributed to variations in object color and lighting conditions, which differ from the ideal rendered training data and make extracting precise parameter information from images more challenging.

For the third and fourth images, TransformCAD accurately generated both the commands and parameters, suggesting that it effectively understands the "rectangular" shape described in the text. The fifth and sixth images contain relatively complex geometries. In the fifth case, the model correctly identified the "curved, rounded shape" from the input text but failed to capture the subsequent requirement of a "rectangular block," indicating a need for improvement in understanding longer text descriptions. In the sixth case, the generated CAD model closely resembles the intended shape, but the contour does not perfectly match the input. Specifically, the original contour consists of three overlapping circles, whereas the generated CAD model features tangentially connected circles.

Despite occasional parameter prediction errors, Transform-

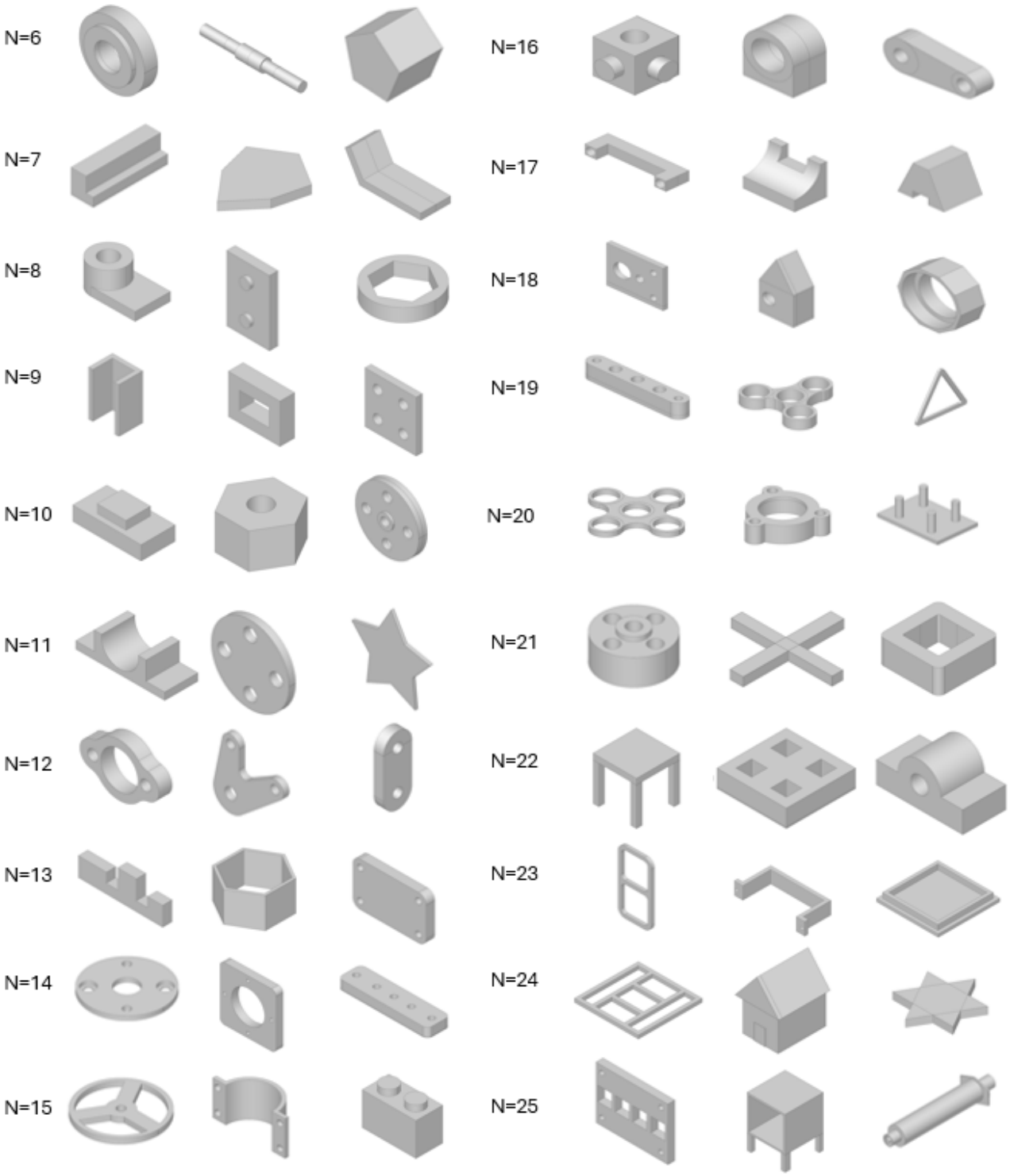

FIGURE 9. A gallery of correctly generated models. *N* denotes the sequence length.

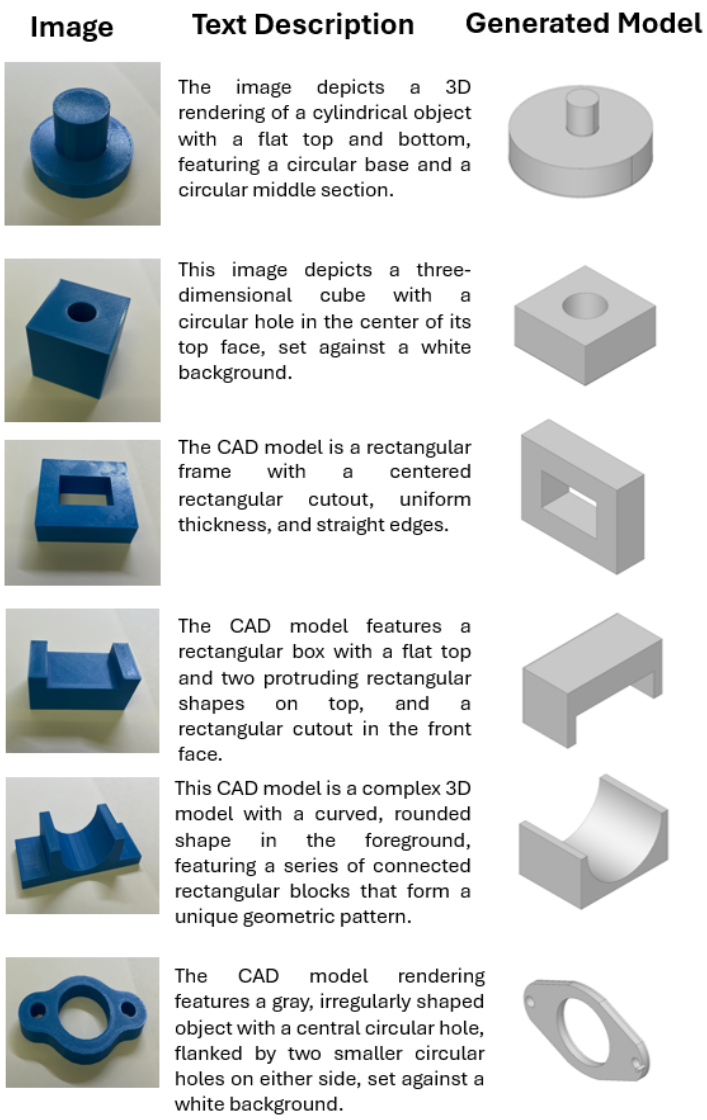

FIGURE 10. CAD generation real-world images.

CAD generates CAD sequences that are easy to edit, highlighting its potential for industrial applications.

5.3 Flaw Generation

Figure 11 illustrates examples where our model generated incorrect CAD models based on the input image and text. One common issue is incorrect extrude direction, as observed in the first and second columns of the figure. In these cases, the extrusion is performed in the opposite direction, indicating that the Extrude Decoder requires further improvement, such as increasing the number of Transformer layers to enhance its ability to capture long-range dependencies and expanding the latent representation dimension to improve feature expressiveness and directional accuracy. Another limitation is the model's difficulty in understanding geometry topology. As shown in the third column, the model incorrectly places a hole on the wrong face, suggesting challenges in learning spatial relationships.

For complex CAD models, the model sometimes captures only the overall shape while failing to generate finer details, as demonstrated in the fourth column. This is likely because intricate features are difficult to infer from both image and text descriptions. Additionally, our findings indicate that because the model is primarily trained on practical CAD models, it struggles with creative or unconventional designs. For instance, in the

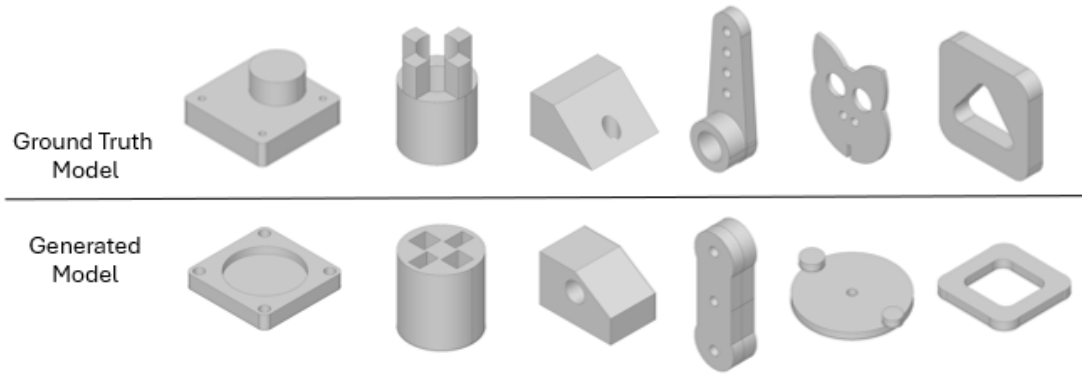

FIGURE 11. The examples of flaw generation.

fifth column, the model fails to generate the cat-face design accurately, highlighting the challenge of handling less conventional geometries.

The last column showcases a hallucination issue, a common problem in generative models. Even though the image and text description clearly indicate a triangular hole, the model incorrectly generates a rectangular hole instead, suggesting a bias or uncertainty in shape generation.

5.4 Limitation and Future Direction

During this study, we realized that TransformCAD's performance is limited by the single-image input, which provides only a restricted amount of geometric information. Since text descriptions do not include the CAD model's parameters, the input image has a significant influence on parameter prediction. To address this limitation, our next step will be to incorporate multiple images along with text descriptions as input to enhance the model's understanding of geometry.

Additionally, to improve robustness for real-world images, data augmentation could be applied to the image data. This includes modifying the CAD model's color, altering the background, and adjusting the image proportions. Such augmentations could lead to a more robust performance in real-world scenarios, where lighting and background conditions differ from ideal rendered images.

To further enhance parameter prediction accuracy, we plan to increase the number of training epochs and expand the size of the codebook in the Geometry Encoder, which may improve the model's ability to capture geometric details more effectively.

Finally, we aim to develop a user interface that allows users to modify the parameters of generated CAD models. Incorporating human-computer interaction will enable users to refine and adjust generated designs, ensuring greater precision and alignment with their specific design intent. This interactive system could also provide real-time feedback, helping users explore different design variations efficiently.

6 Conclusion

This study presents TransformCAD, a generative model of 3D CAD that takes both image and text descriptions as input. Using the fine-tuned CLIP model, TransformCAD converts images and text into embeddings, which are then processed through a structured prediction framework based on the SkexGen model. Two prediction branches—sketch and extrude—are applied to generate the final CAD operation sequence. By applying the hierarchical sketch-extrude representation, the model ensures a high parsing rate by effectively capturing geometric structures and dependencies. Performance evaluations show that TransformCAD outperforms other models in terms of Parsing Rate, Chamfer Distance, Minimum Matching Distance, and Jensen-Shannon Divergence.

In summary, the combined image and text input offers greater control to users, making TransformCAD highly applicable in real-world industrial applications. Additionally, the CAD sequence output is easily editable and convertible into other formats, such as B-rep and STL. Furthermore, by investigating the influence of training data, we demonstrate that TransformCAD has strong potential for accurately generating long-sequence CAD models, which correspond to higher-complexity designs.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support from the National Science Foundation through Award 2207408.

REFERENCES

[1] Uy, M. A., yu Chang, Y., Sung, M., Goel, P., Lambourne, J., Birdal, T., and Guibas, L., 2022. "Point2cyl: Reverse

engineering 3d objects from point clouds to extrusion cylinders". In Conference on Computer Vision and Pattern Recognition (CVPR).

[2] Ren, D., Zheng, J., Cai, J., Li, J., and Zhang, J., 2022. "Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing". In Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II, Springer-Verlag, p. 482–498.

[3] Lambourne, J. G., Willis, K., Jayaraman, P. K., Zhang, L., Sanghi, A., and Malekshan, K. R., 2022. "Reconstructing editable prismatic cad from rounded voxel models". In SIGGRAPH Asia 2022 Conference Papers, SA '22, Association for Computing Machinery.

[4] Li, P., Guo, J., Zhang, X., and Yan, D., 2023. "Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations". *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16816–16826.

[5] Wang, S., Chen, C., Le, X., Xu, Q., Xu, L., Zhang, Y., and Yang, J., 2024. Cad-gpt: Synthesising cad construction sequence with spatial reasoning-enhanced multimodal llms.

[6] Wu, R., Xiao, C., and Zheng, C., 2021. "DeepCAD: A Deep Generative Network for Computer-Aided Design Models". pp. 6772–6782.

[7] Xu, X., Willis, K. D., Lambourne, J. G., Cheng, C.-Y., Jayaraman, P. K., and Furukawa, Y., 2022. "Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks". In International Conference on Machine Learning, PMLR, pp. 24698–24724.

[8] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G., 2023. Llama: Open and efficient foundation language models.

[9] Li, X., Sun, Y., and Sha, Z., 2024. "Llm4cad: Multimodal large language models for three-dimensional computer-aided design generation". *Journal of Computing and Information Science in Engineering,* ***25***(2), 12, p. 021005.

[10] Sun, Y., Li, X., and Sha, Z., 2025. "Large language models for computer-aided design fine tuned: Dataset and experiments". *Journal of Mechanical Design,* ***147***(4), 02, p. 041710.

[11] Badagabettu, A., Yarlagadda, S. S., and Farimani, A. B., 2024. Query2cad: Generating cad models using natural language queries.

[12] Mallis, D., Karadeniz, A. S., Cavada, S., Rukhovich, D., Foteinopoulou, N., Cherenkova, K., Kacem, A., and Aouada, D., 2024. Cad-assistant: Tool-augmented vllms as generic cad task solvers?

[13] Xu, J., Wang, C., Zhao, Z., Liu, W., Ma, Y., and Gao, S., 2024. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm.

[14] Alam, M. F., and Ahmed, F., 2024. Gencad: Image-conditioned computer-aided design generation with transformer-based contrastive representation and diffusion priors.

[15] Jobczyk, H., and Homann, H., 2023. Automatic reverse engineering: Creating computer-aided design (cad) models from multi-view images.

[16] Li, X., and Sha, Z., 2025. Image2cadseq: Computer-aided design sequence and knowledge inference from product images.

[17] Zhou, S., Zan, X., Li, Z., and Zhou, B., 2024. "CAD-Gen: Computer-aided design sequence construction with a guided codebook learning". *Digital Twins and Applications,* ***1***(1), Sept., pp. 75–87.

[18] Wang, X., Xu, Y., Xu, K., Tagliasacchi, A., Zhou, B., Mahdavi-Amiri, A., and Zhang, H., 2020. "Pie-net: Parametric inference of point cloud edges". *Advances in neural information processing systems,* ***33***, pp. 20167–20178.

[19] Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S., and Měch, R., 2020. Parsenet: A parametric surface fitting network for 3d point clouds.

[20] Kim, C., Lee, J., Cho, M., and Kim, C., 2011. "Fully automated registration of 3d cad model with point cloud from construction site". In 28th International Symposium on Automation and Robotics in Construction (ISARC 2011), S. Kwon, ed., International Association for Automation and Robotics in Construction (IAARC), pp. 917–922.

[21] Budroni, A., and Boehm, J., 2010. "Automated 3d reconstruction of interiors from point clouds". *International Journal of Architectural Computing,* ***8***(1), pp. 55–73.

[22] Liu, J., 2020. "An adaptive process of reverse engineering from point clouds to cad models". *International Journal of Computer Integrated Manufacturing,* ***33***(9), pp. 840–858.

[23] van den Oord, A., Vinyals, O., and Kavukcuoglu, K., 2018. Neural discrete representation learning.

[24] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I., 2017. "Attention is all you need". In Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Vol. 30, Curran Associates, Inc.