MamTiff-CAD: Multi-Scale Latent Diffusion with Mamba+ for Complex Parametric Sequence

Liyuan Deng[1] Yunpeng Bai[2] Yongkang Dai[1] Xiaoshui Huang[3] Hongping Gan[1]
Dongshuo Huang[1] Hao Jiacheng[4] Yilei Shi[1]*
[1]Northwestern Polytechnical University [2]National University of Singapore
[3]Shanghai Jiao Tong University [4]Nanchang University
{dly,daiyongkang,ganhongping,huangdongshuo,yilei_shi}@mail.nwpu.edu.cn
bai_yunpeng99@u.nus.edu huangxiaoshui@sjtu.edu.cn haojiacheng@email.ncu.edu.cn

Abstract

Parametric Computer-Aided Design (CAD) is crucial in industrial applications, yet existing approaches often struggle to generate long sequence parametric commands due to complex CAD models' geometric and topological constraints. To address this challenge, we propose MamTiff-CAD, a novel CAD parametric command sequences generation framework that leverages a Transformer-based diffusion model for multi-scale latent representations. Specifically, we design a novel autoencoder that integrates Mamba+ and Transformer, to transfer parameterized CAD sequences into latent representations. The Mamba+ block incorporates a forget gate mechanism to effectively capture long-range dependencies. The non-autoregressive Transformer decoder reconstructs the latent representations. A diffusion model based on multi-scale Transformer is then trained on these latent embeddings to learn the distribution of long sequence commands. In addition, we also construct a dataset that consists of long parametric sequences, which is up to 256 commands for a single CAD model. Experiments demonstrate that MamTiff-CAD achieves state-of-the-art performance on both reconstruction and generation tasks, confirming its effectiveness for long sequence (60-256) CAD model generation.

1. Introduction

Computer-Aided Design (CAD) is a core tool in modern industrial product manufacturing, constructing complex 3D models through parametric command sequences (such as Sketch, Extrusion, and Boolean operations). It not only defines geometric shapes but also thoroughly records design logic and engineering intentions. With the growing demand for highly complex and long sequence CAD models in industrial design, existing generation methods are limited by computational efficiency and sequence modeling capabilities, making it difficult to effectively handle industrial-grade design processes that contain hundreds of commands.

*Corresponding author.

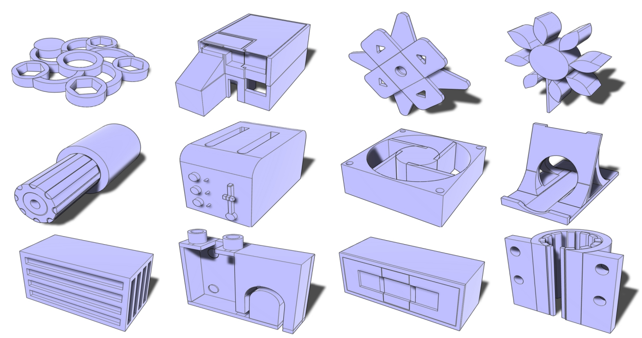

Figure 1. Gallery of Generated CAD Designs. Our generative model infers parametric CAD command sequences, enabling the creation of diverse and structurally valid CAD models. The resulting 3D shapes exhibit clean geometry, well-defined features, and full editability, allowing users to modify designs seamlessly.

The key challenges lie in overcoming the bottleneck of long sequence modeling and enabling the automated generation of complex CAD models. Significant progress has been made in deep learning-based parametric CAD modeling, with Transformer architectures being employed to genarate parametric CAD models [42, 45, 46]. While these approaches achieve strong performance in short-sequence command generation, the inherent quadratic complexity of the Transformer's self-attention mechanism limits its scalability to longer sequences [37]. This constraint makes it difficult to model intricate CAD structures, significantly restricting the applicability of existing methods for generating complex CAD models.

To address these challenges, we propose MamTiff-

CAD, a novel CAD command sequence generation framework that integrates Mamba+ and Transformer autoencoder with a multi-scale diffusion model. In the autoencoding stage, the Mamba+ encoder optimizes state transitions through a gating mechanism, improving its ability to model long-range dependencies. It works with a non-autoregressive Transformer decoder to reconstruct latent representations of CAD command sequences. Once trained, the encoder maps parametric CAD data into a compact, low-dimensional latent space. In the generation stage, a multi-scale Transformer-driven diffusion model is employed. Through hierarchical denoising, it synchronizes local geometric details with global topological constraints, generating logically coherent latent vectors from Gaussian noise. These vectors are then decoded into executable parametric CAD command sequences. We make following contributions in this paper:

- proposing a novel autoencoder architecture that combines Mamba+ and Transformer to model long sequence parametric CAD commands.
- proposing a multi-scale diffusion-based CAD sequence generator to fuse local and global topology, improving both generation quality and
- evaluations showing that our approach achieving state-of-the-art (SOTA) performance on a long sequence CAD dataset containing 13,705 samples with sequence lengths ranging from 60 to 256.

2. Related Work

2.1. CAD Model Generation

Parametric CAD generation has recently shifted from traditional geometric representations to deep learning approaches emphasizing sequence modeling. DeepCAD [42] pioneered the use of a Transformer architecture for autoregressive CAD command generation, but it was constrained to short sequences (length < 60), limiting its ability to capture hierarchical structures in complex models. Follow-up efforts introduced hierarchical representations to improve generation accuracy. SkexGen [45] proposed a decoupled approach for sketching and extrusion operations, facilitating geometric-topological separation, while HNC-CAD [46] built independent codebooks for rings, sketches, and extrusions via a VQ-VAE [36] framework, enabling multi-granularity control. Nevertheless, these methods still grapple with the quadratic computational overhead of Transformer-based architectures, hindering their ability to generate complex, long-sequence CAD models.

The rise of multimodal generation techniques has opened new pathways for CAD generation. CAD-MLLM [44] constructed the first multimodal dataset supporting text, image, and point cloud inputs. However, its generative model still faces topological rupture issues in complex geometric overlapping regions. FlexCAD [49], by fine-tuning LLMs[13], has achieved controllable CAD generation, but the generated models are relatively simple and do not meet the complex design demands of industrial-grade applications.

2.2. Long-Sequence Modeling Techniques

A key challenge in long-sequence modeling is balancing computational efficiency with long-term dependency capture. Traditional methods [5, 12] suffer from the problem of vanishing gradients, making it difficult to effectively capture long-term dependencies. While Transformers address this via self-attention, their quadratic complexity in sequence length drives up memory and training costs. To address this issue, sparse attention mechanisms [2, 14, 29, 38] and local window strategies [22, 32, 33]have been proposed. These methods can alleviate computational bottlenecks to some extent. However, they sacrifice global context awareness, which can lead to design logic breakdowns and information loss when processing CAD sequence generation.

State Space Models [10] offer a novel approach for long-sequence modeling. Mamba [8] incorporates a selective scanning mechanism to achieve long-sequence modeling with linear complexity, demonstrating significant advancements in fields such as audio [3, 16, 40], vision [4, 17, 20, 21], and 3D [19, 39]. However, Mamba's application in the CAD domain remains underexplored. In this work, we adapt Mamba for CAD sequence modeling and refine its structure to better capture long-range dependencies.

2.3. 3D Generation with Diffusion Models

Diffusion models, as an advanced generative approach, achieve high-quality generation through the process of gradually adding Gaussian noise to the data and then performing denoising in reverse [11]. In the field of 3D generation, diffusion models have been widely applied to tasks such as point cloud to mesh [9, 26] image to 3D reconstruction [23, 28, 34, 43] and CAD model generation [7, 47].

Recent studies have made significant progress in combining diffusion models with 3D generation. Generating 3D models through point clouds remains the mainstream method [25, 35, 50]. For example, 3DShape2VecSet[48] proposed a latent set diffusion framework, achieving 3D shape generation and point cloud completion through multi-scale feature fusion. DiT-3D[30] is the first Diffusion Transformer architecture designed for 3D shape generation, leveraging the denoising process of DDPM on voxelized point clouds to efficiently generate high-fidelity 3D models. DiffCAD [7] employs diffusion models to learn the probabilistic model of CAD generation and alignment. GetMesh [27] uses a latent set diffusion model to generate meshes from point clouds, while Polydiff [1] uses a discrete denoising diffusion model to process mesh data, generating 3D triangular mesh structures. Diffusion-SDF [18] generates

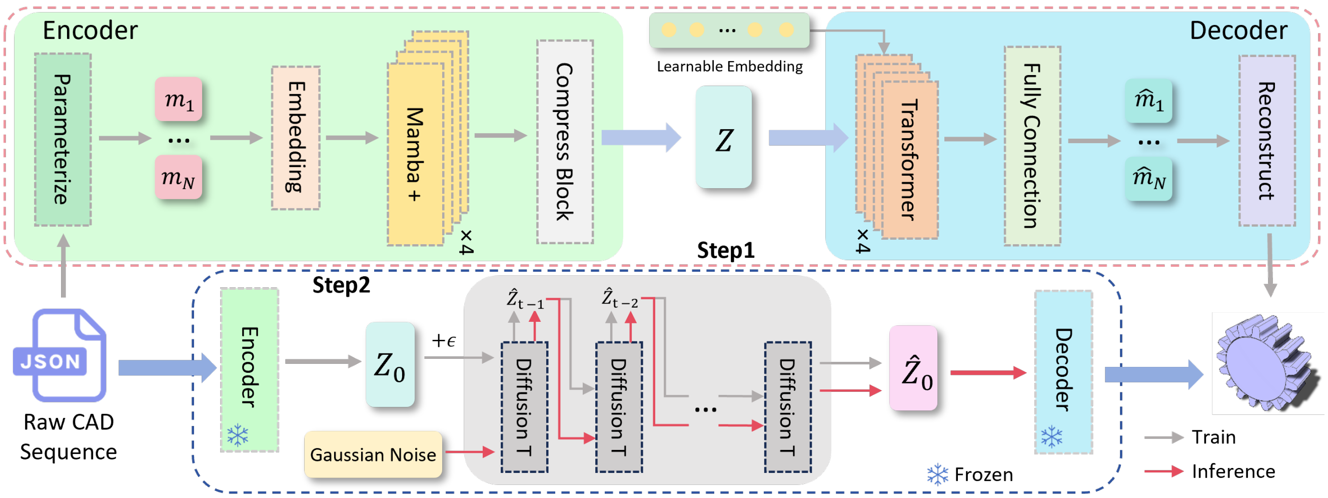

Figure 2. The framework of MamTiff-CAD consists of two main steps. In Step 1, an autoencoder integrating Mamba and Transformer architectures is used to learn the latent representation of CAD command sequences. In Step 2, a diffusion-based module is employed to model the generative distribution of the learned latent representations. During inference, the model directly generates latent variables, which are subsequently decoded to reconstruct the CAD command sequences.

3D shapes via diffusion models, offering flexibility in generating diverse shapes that match textual descriptions.

Although these methods have shown potential in generating complex 3D structures, they still face challenges when dealing with parametric CAD sequences. Existing methods primarily focus on generating geometric shapes, neglecting the design logic and parametric constraints of CAD models, which makes the generated results difficult to directly apply to industrial design. The MamTiff-CAD framework proposed in this paper is the first to introduce diffusion models into the field of parametric CAD sequence generation. By learning in latent space vectors and combining Mamba's efficient long-sequence modeling capabilities, it significantly enhances the stability of long-sequence generation.

3. Method

3.1. Architecture Overview

As shown in Figure 2, our MamTiff-CAD architecture consists of two steps. In the first step, we train a autoencoder to encode CAD command sequences into latent representation. The input CAD command sequences is represented as a set of parameterized commands $M = \{m_i\}_{i=1}^{N_c}$, where N_c is the fixed sequence length set to 256. This CAD first passes through an embedding layer and is then fed into Mamba+ blocks. Afterwards, it undergoes compression via a Compress Block for the convenience of following training, generating a latent vector Z. The Transformer-based decoder takes both a learnable embedding and Z as inputs and subsequently outputs the predicted command sequence $\hat{M} = \{\hat{m}_i\}_{i=1}^{N_c}$. In the second step, we use diffusion-based networks to learn the generation of encoded latent representation Z. During inference, MamTiff-CAD uses the trained diffusion network to generate the latent representation $\hat{Z}$, which is then decoded into a command sequence using the frozen decoder.

3.2. AutoEncoder for Latent Representation

Parametric CAD Representation. We follow DeepCAD [42] to represent each CAD model as a sequence of parametric commands $M = \{(C_1, p_1), \ldots, (C_n, p_n)\}$, where C_i is a command type and p_i is its associated parameter set. For neural network input, we set the sequence length to 256 and end with the special token $\langle\text{EOS}\rangle$. For sequences less than 256, we pad the length to 256. Each command's parameters p_i are encoded as a 16-dimensional vector[1]: $\mathbf{p}_i = [x, y, \alpha, f, r, \theta, \phi, \gamma, p_x, p_y, p_z, s, e_1, e_2, b, u] \in \mathbb{R}^{16}$,where unused entries are padded with -1. Continuous parameters are normalized to a $2 \times 2 \times 2$ cube and quantized into 256 discrete levels to preserve geometric consistency. All parameters are eventually converted to discrete tokens for learning.

Forget Gate Encoder. To encode CAD commands into the latent space Z, we employ a Forget Gate Encoder. We first embed the parametric representation by considering the command type C_i, parameters p_i, and the command's position within the sequence pos_i. The embedding of each CAD command, $\text{Emb}(m_i)$, is defined as follows:

$$\text{Emb}(m_i) = W_{\text{cmd}}\delta_{C_i} + W_{\text{pa}} \cdot \text{flatten}(W_{\text{pb}}\delta_{p_i}) + W_{\text{pos}}\delta_i \quad (1)$$

[1] See Supplementary Materials for details.

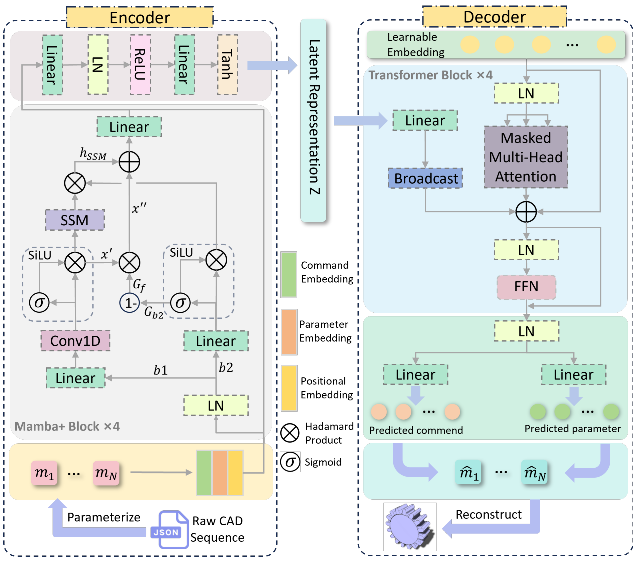

Figure 3. An overview of our autoencoder architecture. The input CAD command sequence is first parameterized and fused with command, parameter, and positional embeddings. It is then processed through four Mamba+ blocks and mapped to the latent representation Z via a compression bolck. Finally, the Transformer decoder reconstructs the CAD sequence, predicting command types and their corresponding parameters.

Here, $W_{\text{cmd}} \in \mathbb{R}^{d_E \times 6}$ is a learnable matrix and $\delta_{C_i} \in \mathbb{R}^6$ is a one-hot vector indicating the command type. The system defines six command types. Each CAD command comprises 16 parameters, each quantized into an 8-bit integer. We convert each integer into a one-hot vector of dimension 257, where 256 dimensions correspond to the quantized values and the remaining one dimension indicates that the parameter is unused. Stacking these 16 one-hot vectors forms a matrix $\delta_{p_i} \in \mathbb{R}^{257 \times 16}$. Then, using the learnable matrix $W_{\text{pb}} \in \mathbb{R}^{d_E \times 257}$, we embed each parameter individually, flatten the resulting embeddings, and apply a linear transformation via $W_{\text{pa}} \in \mathbb{R}^{d_E \times 16 d_E}$. Finally, $W_{\text{pos}} \in \mathbb{R}^{d_E \times N_c}$ is another learnable matrix, and $\delta_i \in \mathbb{R}^{N_c}$ is a one-hot vector that marks the command's position within the sequence, with $N_c = 256$ being the maximum sequence length.

We then use four Mamba+ blocks, i.e. optimized SSM-based modules integrated with a forget gate mechanism, as our main encoder block. Specifically, the Mamba+ block employs a dual-branch structure for information processing. The feature transformation branch ($b1$) extracts sequence features using 1D convolution and SSM blocks, while the gating mechanism branch ($b2$) generates control signals for information flow using the SiLU activation function. To further enhance the retention of historical information, we introduce a forget gate G_f as:

$$\mathrm{G}_f = 1 - \mathrm{G}_{b2} \tag{2}$$

G_{b2} is calculated using the Sigmoid function, controlling the flow of current input information. The forget gate G_f, modulates the output x' from $b1$, ensuring that some historical information is not completely discarded. The updated historical information is given by $x'' = \mathrm{G}_f \cdot x'$. The final output of Mamba+ blocks integrates the SSM computation result with the feature adjusted by the forget gate $h_{\text{out}} = x'' + h_{\text{SSM}}$. This design enables the Mamba+ block to effectively extract new features while preserving long-term dependencies, enhancing its ability to process complex CAD command sequences efficiently.

To efficiently encode long sequence CAD commands, we construct the Mamba+ block by stacking four layers of Mamba+ blocks, capable of modeling CAD command sequences ranging from 60 to 256 in length. The encoder first converts the input command sequence $M = \{m_1, m_2, \ldots, m_{256}\}$ into continuous representations through an embedding layer, then progressively captures local and global dependencies through hierarchical processing in the Mamba+ blocks.

Decoder. We employ a Transformer-based decoder, which consists of four blocks, to reconstruct parametric CAD command sequences. The decoder's input comprises two parts: 1) a latent vector Z, and 2) a set of learnable embedding vectors produced by a learnable embedding layer, which provides a representation for each position in the sequence. In our model, the maximum sequence length is set to $N_c = 256$. The decoder processes these inputs layer by layer to capture global dependencies among commands in long sequences and ultimately generates the predicted CAD command sequence $\hat{M} = \{\hat{m}_1, \hat{m}_2, \ldots, \hat{m}_{256}\}$, where each command $\hat{m}_i$ consists of a command type $\hat{C}_i$ and parameters $\hat{\mathbf{p}}_i$. Our objective is to maximize the generation probability:

$$p(\hat{M} \mid z, \Theta) = \prod_{i=1}^{N_c} p(\hat{C}_i, \hat{\mathbf{p}}_i \mid z, \Theta) \tag{3}$$

Here, Θ denotes the decoder's network parameters. Finally, the decoder's output hidden states are mapped through a linear transformation layer into the specific command type and parameter space, thereby completing the generation of the entire CAD command sequence.

Training of the Autoencoder. We train our autoencoder by reconstructing parameterized CAD command sequences to learn the latent representation Z of CAD models. The training objective is to minimize the discrepancy between the predicted outputs and the ground truth, ensuring that the model accurately captures the structural information of the parameterized CAD sequences. We employ a cross-entropy loss function to supervise the learning of both CAD command types and parameters.

Our output sequence is fixed to $N_c = 256$ commands, each with $N_p = 16$ parameters. For the i-th command, the ground truth command type is denoted by t_i (derived from the true command C_i), and the corresponding ground truth parameters are represented as $\{a_{i,j}\}_{j=1}^{N_p}$ (where each $a_{i,j}$ is an element of the previously defined parameter vector). The model predicts a probability distribution over command types p_i such that $p_i(t_i)$ represents the probability assigned to the true command type, and for each parameter $a_{i,j}$, it predicts a probability distribution $q_{i,j}$ over quantized parameter values.The total loss is defined as:

$$L = \sum_{i=1}^{N_c} \ell\big(p_i(t_i)\big) + \beta \sum_{i=1}^{N_c} \sum_{j=1}^{N_p} \ell\big(q_{i,j}(a_{i,j})\big) \tag{4}$$

where β is set to 2 to balance the parameter loss against the command type loss. Contributions from padding commands, such as the $\langle$EOS$\rangle$ token, and unused parameters labeled as -1 are ignored during loss computation. For further details on the experimental configurations, please refer to the Appendix.

3.3. Diffusion Generator

The above frozen autoencoder encodes the command sequences into a compact latent variable $Z \in \mathbb{R}^{N \times 256 \times 64}$. In the forward process, we employ a linear variance schedule $\{\beta_t\}$ to add noise into the latent space, while the denoising process is guided by a Multi-Scale Transformer-based Diffusion Generator (MST-D), as shown in Figure 4.

Multi-Scale Transformer Denoiser. The denoising network is composed of stacked multi-scale Transformer layers, each following a three-stage pipeline: *Hierarchical Attention*: Each layer contains three parallel attention branches with window sizes of 64, 128 and 256 to capture local geometric constraints, medium-range topological dependencies, and global semantic coherence respectively, processing the latent sequence at three different scales. *Adaptive Fusion*: The outputs of these attention branches are concatenated and fused via a learnable gating mechanism:

$$\mathbf{H} = \mathrm{MLP}\left(\sigma(\mathbf{W}_g[\mathbf{H}_l \| \mathbf{H}_m \| \mathbf{H}_g]) \odot [\mathbf{H}_l \| \mathbf{H}_m \| \mathbf{H}_g]\right) \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function, and $\|$ denotes concatenation. *Time-Step Embedding*: This module inputs temporal information into each Transformer block by transforming the diffusion time step t into an embedding vector. The embedding is then processed through an MLP to generate two sets of scale and shift parameters $\{\xi_i, \psi_i, \omega_i\}$. Here, ξ_i and ψ_i modulate the layer-normalized activations, while ω_i adjusts the strength of the residual connections. Together, these parameters enable the Transformer to adapt its behavior in a time-aware manner.

Sequence-Aware Positional Encoding. To preserve the sequential logic of CAD commands, we employ a scalable sinusoidal positional encoding PE, defined as

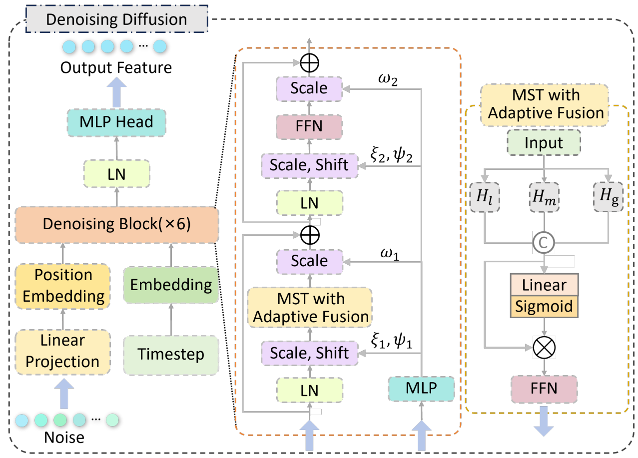

Figure 4. Our denoising diffusion model. The input noise undergoes linear projection with positional and timestep embeddings, followed by feature extraction through encoder blocks (×6). The core denoising structure, MST with Adaptive Fusion, integrates multi-scale attention and adaptive fusion to dynamically adjust feature distributions. Finally, the MLP Head reconstructs the denoised output features.

$$PE(Z) = Z + \eta \cdot \mathrm{PE}(\mathrm{pos}, D) \tag{6}$$

where $\eta \in \mathbb{R}$ is a trainable scalar that adaptively adjusts the positional weighting. Instead of writing the full vector in one long equation, we define the positional encoding function piecewise as follows:

$$\begin{aligned} \mathrm{PE}_{2i}(\mathrm{pos}, D) &= \sin\Big(\frac{\mathrm{pos}}{10000^{2i/D}}\Big) \\ \mathrm{PE}_{2i+1}(\mathrm{pos}, D) &= \cos\Big(\frac{\mathrm{pos}}{10000^{2i/D}}\Big) \end{aligned} \tag{7}$$

for $i = 0, 1, \ldots, \frac{D}{2} - 1$. Here, pos denotes the sequence position, and D is the embedding dimension. This encoding scheme provides both absolute and relative position information, enabling the model to capture the sequential dependencies of CAD commands more effectively.

Training and Generation. The denoising network learns to predict the noise injected at each timestep, modeling the distribution of the target latent space. Given a clean latent vector Z_0 from the frozen encoder, we first sample a timestep $t \sim U(1, T)$ and add noise according to the linear variance schedule $\{\beta_t\}$:

$$Z_t = \alpha_t Z_0 + \sqrt{1 - \alpha_t}\,\epsilon \tag{8}$$

where $\epsilon \sim \mathcal{N}(0, I)$, $\alpha_t = \prod_{s=1}^{t}(1 - \beta_s)$ controls the rate of noise accumulation. The network ϵ_θ is then trained to predict ϵ by minimizing the mean squared error:

$$L_{\text{diff}} = \mathbb{E}_{t, Z_0, \epsilon}\big[\|\epsilon - \epsilon_\theta(Z_t, t)\|_2^2\big] \tag{9}$$

Table 1. Statistical comparison between the DeepCAD dataset and our ABC-256 dataset, including the total number, the average length of parametric CAD sequences, and the length distribution.

Dataset	Total	Average Length	1-10	11-60	61-128	129-256
DeepCAD	178,238	15	44.58	55.42	-	-
ABC-256	13,705	99	-	-	82.89	17.11

Sampling Process: Starting from Gaussian noise Z_T, the latent vector is iteratively denoised using:

$$Z_{t-1} = \frac{1}{\alpha_t}\Big(Z_t - \beta_t \frac{1}{\sqrt{1-\alpha_t}}\,\epsilon_\theta(Z_t, t)\Big) + \beta_t z \qquad (10)$$

where $z \sim \mathcal{N}(0, I)$. Here, α_t and β_t denote the attenuation factor and variance schedule parameter at timestep t, respectively. Through this process, our diffusion generator progressively removes noise in the latent space to yield CAD command sequences with global consistency and rich structural information.

4. ABC-256 Dataset

Although several publicly available CAD datasets exist, such as ABC [15], DeepCAD [42], Fusion 360 Gallery [41], and CAD as Language [6], datasets that fully capture the CAD design process remain scarce. The ABC dataset contains 1 million B-rep models from Onshape [31], but lacks design operation sequences, making full process reconstruction difficult. The Fusion 360 Gallery dataset provides CAD command sequences based on sketch profile extrusions, but these sequences are short, limiting their suitability for long sequence CAD generation. DeepCAD includes 178,238 CAD models with parameterized sequences, but its maximum sequence length of 60 constrains its ability to model complex, long sequence CAD designs.

To address this, we constructed and publicly released a complex CAD command sequence dataset to advance research in this field. Using ABC dataset model links, we leveraged the Onshape API and FeatureScript to extract CAD commands from CSG representations and convert them into parameterized sequences. We then filtered models to retain only those with complete operations, excluding simpler cases with only sketching and extrusion, ensuring sequence lengths between 60 and 256. The final dataset includes 13,705 CAD models, split into 10,964 for training, 1,370 for validation, and 1,371 for testing. Compared to DeepCAD, our dataset has a 6.6× longer average sequence length (Table 1), offering greater complexity in both length and distribution, making it more suitable for long sequence CAD model generation.

5. Experiment

5.1. Implementation details

We implemented the MamTiff-CAD model based on the PyTorch framework and completed the training of the model on an NVIDIA RTX 4090 GPU. During training, we adopted the AdamW optimizer [24] with an initial learning rate of 0.001, applying linear warmup for the first 2000 steps. Additionally, we introduced gradient clipping to prevent gradient explosion issues. The training of MamTiff-CAD is divided into two stages. The first stage involves training for 300 epochs with a batch size of 32. In the second stage, training for 200,000 epochs with a batch size of 64 to ensure the model has enough time to fully learn and reach convergence.

5.2. Evaluation Metrics

We comprehensively evaluate the performance of the autoencoder using five metrics, including *Command Accuracy*, *Parameter Accuracy*, *Median Chamfer Distance*, *Invalid Ratio*, and *Step Ratio*.

- *Command Accuracy* ($\mathrm{ACC_c}$): the match between the predicted CAD command type and the ground truth.
- *Parameter Accuracy* ($\mathrm{ACC_p}$): the match of command parameters under the condition that the command type is correctly predicted.
- *Median Chamfer Distance* (MCD): the geometric error between the reconstructed 3D shape and the reference shape, computed as the median distance between corresponding points.
- *Invalid Ratio* (IR): the proportion of CAD models that cannot be converted into point clouds.
- *Step Ratio* (SR): the proportion of generated CAD sequences successfully converted to the STEP format.

For the performance evaluation of the generator, we adopt six metrics, including *Coverage* (COV), *Jensen-Shannon Divergence* (JSD), *Minimum Matching Distance* (MMD), *Novelty*, *Uniqueness*, and *Step Ratio*. The specific metric descriptions are as follows:

- *Coverage* (COV): whether the generated data covers the distribution of real data.
- *Jensen-Shannon Divergence* (JSD): the difference between the generated data distribution and the real data distribution.
- *Minimum Matching Distance* (MMD): the minimum average distance between generated and real samples.
- *Unique*: the proportion of non-duplicate samples in the generated dataset.
- *Novel*: the proportion of samples in the generated dataset that do not appear in the training set.

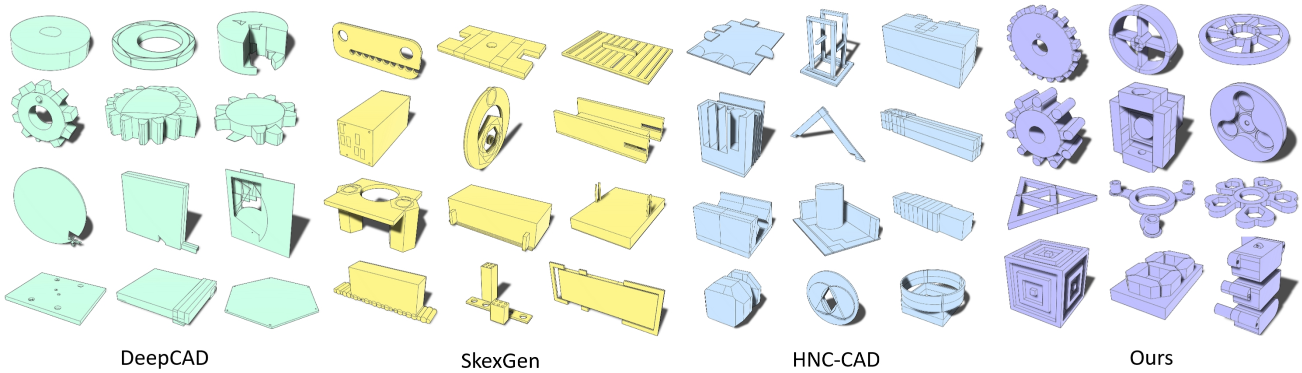

Figure 5. This figure compares the CAD models generated by DeepCAD, SkexGen, HNC-CAD, and MamTiff-CAD, highlighting differences in shape complexity and geometric details among the methods. It can be observed that MamTiff-CAD produces models with higher structural integrity and greater complexity.

5.3. Autoencoding Parametric CAD

To comprehensively evaluate the effectiveness of our proposed MamTiff-CAD framework, we conducted a comparative analysis across multiple models, including DeepCAD, MT-CAD, MLSTM-CAD, and our own model. Among them, MT-CAD integrates Mamba with Transformer, while MLSTM-CAD combines Mamba with LSTM. Experimental results confirm that our framework offers significant advantages in long sequence parametric CAD modeling, further reinforcing the rationale behind our design decisions.

Results. The experimental results are presented in Table 2. Overall, our model achieves SOTA performance across all evaluation metrics. Specifically, our proposed MamTiff-CAD attains a command accuracy of 99.99% and a parameter accuracy of 99.93%, demonstrating superior capability in long command sequence prediction. In terms of Median Chamfer Distance (MCD), MamTiff-CAD outperforms other baseline models, indicating that the reconstructed 3D shapes are geometrically closer to the reference data. Invalid Ratio and Step Ratio, our model achieves an IR of 8.50% and a Step Ratio of 93.93%, indicating a higher proportion of valid command sequences, thereby demonstrating the reliability of MamTiff-CAD. Furthermore, Figure 6 provides a visual comparison of CAD sequence reconstruction results across different models. MamTiff-CAD produces CAD models with better structural integrity and geometric accuracy, making them closer to the ground truth. This further validates the superior performance of MamTiff-CAD in long sequence parametric CAD prediction.

Table 2. Autoencoding performance on ABC-256 dataset. Comparison of MamTiff-CAD and other models in CAD sequence reconstruction. MCD is multiplied by 10^3, and ACC_c, ACC_p, and IR are multiplied by 100%.

Method	$\text{ACC}_c \uparrow$	$\text{ACC}_p \uparrow$	MCD $\downarrow$	IR $\downarrow$	SR $\uparrow$
DeepCAD	92.24	75.93	41.02	33.11%	70.46%
MT-CAD	89.72	66.87	121.35	39.89%	63.97%
MLSTM-CAD	86.09	65.55	112.89	42.53%	59.85%
OURS	**99.99**	**99.93**	**0.75**	**8.50%**	**93.93%**

Table 3. Quantitative evaluation of generalization ability on the Fusion360 reconstruction dataset.

Method	$\text{ACC}_c \uparrow$	$\text{ACC}_p \uparrow$	MCD $\downarrow$	IR $\downarrow$	SR $\uparrow$
DeepCAD	93.35	77.99	104.76	19.41%	82.57%
MT-CAD	91.85	60.18	300.21	11.81%	90.17%
MLSTM-CAD	88.42	62.13	261.80	23.39%	80.97%
OURS	**99.99**	**97.99**	**1.44**	**5.70%**	**95.16%**

Generalization Performance. To evaluate the generalization capability of MamTiff-CAD, we trained the model on the ABC-256 dataset and directly tested it on the Fusion360 dataset, which was not used during training. The Fusion360 dataset consists of approximately 8,000 parametric CAD sequences derived from Autodesk Fusion 360 designs. Although the CAD models in the Fusion360 dataset exhibit simpler structures, which narrows the performance gap between MamTiff-CAD and the baseline models in certain metrics, MamTiff-CAD still consistently outperforms all other models across all evaluation metrics, achieving the best overall performance (see Table 3).

5.4. Unconditional Generation

We comprehensively evaluated latent vector-based CAD generation across several models, including DeepCAD, SkexGen, HNC-CAD, and our MamTiff-CAD. Each method randomly generated 10,000 samples, which were converted to point clouds for metric-based evaluation. The experimental results in Table 4 indicate that our method achieves state-of-the-art (SOTA) performance in all metrics, particularly excelling in JSD (3.19) and Step Ratio

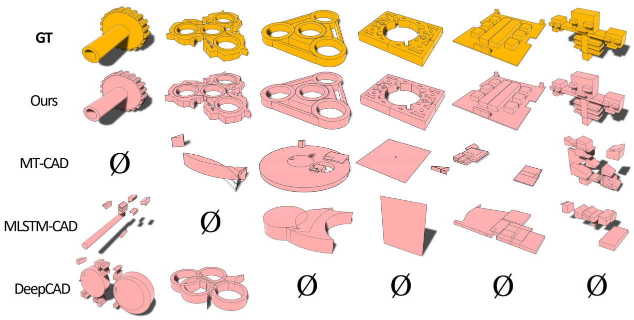

Figure 6. Visual illustrations of CAD sequence reconstruction. GT represents Ground Truth, and Ours refers to MamTiff-CAD. The symbol Ø indicates an invalid CAD sequence, resulting in the failure of 3D shape construction.

Table 4. The unconditional generation result compared with DeepCAD, SkexGen, HNC-CAD and ours.

Method	MMD ↓	JSD ↓	COV ↑	Unique ↑	Novel ↑	SR ↑
DeepCAD	2.66	6.49	56.66%	75.8	88.0	23.96%
SkexGen	2.31	4.53	57.76%	80.5	96.9	75.26%
HNC-CAD	1.63	4.25	62.03%	89.2	91.8	80.86%
OURS	**1.32**	**3.19**	**65.31%**	**99.6**	**99.4**	**85.38%**

(85.38%), demonstrating the higher quality and effectiveness of the generated sequences. Furthermore, as shown in Figure 5, MamTiff-CAD generates more complex CAD models compared to other methods, further validating its advantage in long sequence CAD generation tasks.

5.5. Ablation Studies

We conducted two ablation studies to systematically assess the impact of the two key techniques, Mamba+ and Multi-Scale Transformer (MST), on our model's performance.

Mamba+ Block. To investigate the role of the Mamba+ block in reconstructing long sequence parametric CAD commands, we conducted two ablation experiments: (1) replacing the Mamba+ block with Transformer block, and (2) replacing the Mamba+ block with the standard Mamba block. The experimental results (see Table 5) indicate that, compared to the standard Mamba block, the Mamba+ block achieves improvements across most evaluation metrics. This confirms that the incorporation of the forget gate mechanism enhances the model's ability to capture long-range dependencies in CAD command sequences, with particularly notable improvements in common CAD sequence tasks over the Transformer-based approach.

MST or Non-MST. To assess the impact of MST on our generative model, we conducted an ablation study by randomly generating 10,000 models and comparing the generation capabilities with and without the MST module. The experimental results in Table 6 demonstrate that incorporating MST significantly improves model performance, particularly in terms of generation quality and validity, confirming the critical role of MST in the diffusion process.

Table 5. Comparison of reconstruction performance between Mamba and Mamba+ block for parametric CAD sequences.

Method	ACC_c ↑	ACC_p ↑	MCD ↓	IR ↓	SR ↑
Transformer	77.29	63.62	64.30	67.46%	34.28%
Mamba	99.98	99.90	0.76	10.35%	92.01%
Mamba+	**99.99**	**99.93**	**0.75**	**8.50%**	**93.93%**

Table 6. The unconditional generation result compared between w/o MST and MST.

Method	MMD ↓	JSD ↓	COV ↑	Unique ↑	Novel ↑	SR ↑
w/o MST	1.47	4.92	61.69%	**99.5**	**99.6**	77.05%
w/ MST	**1.32**	**3.19**	**65.31%**	99.6	99.4	**85.38%**

6. Discussion and Conclusion

Although MamTiff-CAD has demonstrated exceptional performance in long-sequence parameterized CAD modeling tasks, several limitations remain that warrant further investigation. Firstly, the dataset used in this study does not cover all CAD command types in industrial design, which somewhat limits the model's ability to handle more complex topological structures. Secondly, this research primarily focuses on the generation and parsing of CAD command sequences and does not directly extract features from Boundary Representation (Brep). Therefore, future research is required to address how to effectively integrate CSG (Constructive Solid Geometry) and Brep representations to enhance the model's generalization ability and applicability. This remains a challenging but promising direction.

In conclusion, this paper presents MamTiff-CAD, a novel framework that integrates an improved Mamba module with a diffusion model to efficiently model long parameterized CAD command sequences and generate complex 3D shapes. We also introduce a dataset of 13,705 CAD models with sequence lengths of 60 to 256, significantly surpassing existing datasets. Experimental results validate MamTiff-CAD's superiority in sequence reconstruction and generation. Future work will focus on refining the model and exploring its integration with advanced design processes and interactive methods to enhance intelligent CAD design.

Acknowledgments

This research was funded by the Fundamental Research Funds for the Central Universities, the Natural Science Basic Research Program of Shaanxi Province under Grant 2024JC-YBQN-0702 and NUS Research Scholarship.Partially supported by Hunan Natural Science Foundation Project (No. 2025JJ50338) and Shanghai Education Committee AI Project(No. JWAIYB-2)

References

[1] Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. Polydiff: Generating 3d polygonal meshes with diffusion models. *arXiv preprint arXiv:2312.11417*, 2023. 2

[2] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 2

[3] Rong Chao, Wen-Huang Cheng, Moreno La Quatra, Sabato Marco Siniscalchi, Chao-Han Huck Yang, Szu-Wei Fu, and Yu Tsao. An investigation of incorporating mamba for speech enhancement. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 302–308. IEEE, 2024. 2

[4] Yubo Cui, Zhiheng Li, Jiaqiang Wang, and Zheng Fang. Loma: Language-assisted semantic occupancy network via triplane mamba. *arXiv preprint arXiv:2412.08388*, 2024. 2

[5] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 2

[6] Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34:5885–5897, 2021. 6

[7] Daoyi Gao, Dávid Rozenberszki, Stefan Leutenegger, and Angela Dai. Diffcad: Weakly-supervised probabilistic cad model retrieval and alignment from an rgb image. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024. 2

[8] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 2

[9] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023. 2

[10] James D Hamilton. State-space models. *Handbook of econometrics*, 4:3039–3080, 1994. 2

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 2

[13] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 2

[14] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020. 2

[15] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9601–9611, 2019. 6, 1

[16] Kai Li, Guo Chen, Runxuan Yang, and Xiaolin Hu. Spmamba: State-space model is all you need in speech separation. *arXiv preprint arXiv:2404.02063*, 2024. 2

[17] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. In *European Conference on Computer Vision*, pages 237–255. Springer, 2024. 2

[18] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12642–12651, 2023. 2

[19] Jiuming Liu, Ruiji Yu, Yian Wang, Yu Zheng, Tianchen Deng, Weicai Ye, and Hesheng Wang. Point mamba: A novel point cloud backbone based on state space model with octree-based ordering strategy. *arXiv preprint arXiv:2403.06467*, 2024. 2

[20] Xiao Liu, Chenxu Zhang, and Lei Zhang. Vision mamba: A comprehensive survey and taxonomy. *arXiv preprint arXiv:2405.04404*, 2024. 2

[21] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. *Advances in neural information processing systems*, 37:103031–103063, 2025. 2

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 2

[23] Zhen Liu, Yao Feng, Michael J Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. Meshdiffusion: Score-based generative 3d mesh modeling. *arXiv preprint arXiv:2303.08133*, 2023. 2

[24] Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5: 5, 2017. 6

[25] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2837–2845, 2021. 2

[26] Zhaoyang Lyu, Jinyi Wang, Yuwei An, Ya Zhang, Dahua Lin, and Bo Dai. Controllable mesh generation through sparse latent point diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 271–280, 2023. 2

[27] Zhaoyang Lyu, Ben Fei, Jinyi Wang, Xudong Xu, Ya Zhang, Weidong Yang, and Bo Dai. Getmesh: A controllable model for high-quality mesh generation and manipulation. *arXiv preprint arXiv:2403.11990*, 2024. 2

[28] Baorui Ma, Yu-Shen Liu, and Zhizhong Han. Learning signed distance functions from noisy 3d point clouds via noise to noise mapping. 2023. 2

[29] Yiqun Mei, Yuchen Fan, and Yuqian Zhou. Image super-resolution with non-local sparse attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3517–3526, 2021. 2

[30] Shentong Mo, Enze Xie, Ruihang Chu, Lanqing Hong, Matthias Niessner, and Zhenguo Li. Dit-3d: Exploring plain

diffusion transformers for 3d shape generation. *Advances in neural information processing systems*, 36:67960–67971, 2023. 2

[31] Onshape. Onshape featurescript. https://cad.onshape.com/FsDoc/, 2019. 6, 1

[32] Xiang Pan, Hongyan Zhou, Chenwei Yang, Luming Sun, Paul S Smith, Tuo Ji, Ning Jiang, Peng Jiang, Wenjuan Liu, Honglin Lu, et al. Mrk 1239: a type-2 counterpart of narrow-line seyfert-1? *The Astrophysical Journal*, 912(2):118, 2021. 2

[33] Pengzhen Ren, Changlin Li, Guangrun Wang, Yun Xiao, Qing Du, Xiaodan Liang, and Xiaojun Chang. Beyond fixation: Dynamic window visual transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11987–11997, 2022. 2

[34] Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. Gecco: Geometrically-conditioned point diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2128–2138, 2023. 2

[35] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022. 2

[36] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 2

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[38] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 2

[39] Zicheng Wang, Zhenghao Chen, Yiming Wu, Zhen Zhao, Luping Zhou, and Dong Xu. Pointramba: A hybrid transformer-mamba framework for point cloud analysis. *arXiv preprint arXiv:2405.15463*, 2024. 2

[40] Zhong-Qiu Wang, Samuele Cornell, Shukjae Choi, Younglo Lee, Byeong-Yeol Kim, and Shinji Watanabe. Tf-gridnet: Making time-frequency domain models great again for monaural speaker separation. In *ICASSP 2023-2023 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1–5. IEEE, 2023. 2

[41] Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4): 1–24, 2021. 6

[42] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782, 2021. 1, 2, 3, 6

[43] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024. 2

[44] Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm. *arXiv preprint arXiv:2411.04954*, 2024. 2

[45] Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. *arXiv preprint arXiv:2207.04632*, 2022. 1, 2

[46] Xiang Xu, Pradeep Kumar Jayaraman, Joseph G Lambourne, Karl DD Willis, and Yasutaka Furukawa. Hierarchical neural coding for controllable cad model generation. *arXiv preprint arXiv:2307.00149*, 2023. 1, 2

[47] Xiang Xu, Joseph Lambourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Bregpen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43 (4):1–14, 2024. 2

[48] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions On Graphics (TOG)*, 42(4):1–16, 2023. 2

[49] Zhanwei Zhang, Shizhao Sun, Wenxiao Wang, Deng Cai, and Jiang Bian. Flexcad: Unified and versatile controllable cad generation with fine-tuned large language models. *arXiv preprint arXiv:2411.05823*, 2024. 2

[50] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5826–5835, 2021. 2