

DTGBrepGen: A Novel B-rep Generative Model through Decoupling Topology and Geometry

Jing Li¹ Yihang Fu¹ Falai Chen^{1*}

¹ University of Science and Technology of China

Abstract

Boundary representation (B-rep) of geometric models is a fundamental format in Computer-Aided Design (CAD). However, automatically generating valid and high-quality B-rep models remains challenging due to the complex interdependence between the topology and geometry of the models. Existing methods tend to prioritize geometric representation while giving insufficient attention to topological constraints, making it difficult to maintain structural validity and geometric accuracy. In this paper, we propose DTGBrepGen, a novel topology-geometry decoupled framework for B-rep generation that explicitly addresses both aspects. Our approach first generates valid topological structures through a two-stage process that independently models edge-face and edge-vertex adjacency relationships. Subsequently, we employ Transformer-based diffusion models for sequential geometry generation, progressively generating vertex coordinates, followed by edge geometries and face geometries which are represented as B-splines. Extensive experiments on diverse CAD datasets show that DTGBrepGen significantly outperforms existing methods in both topological validity and geometric accuracy, achieving higher validity rates and producing more diverse and realistic B-reps. Our code is publicly available at <https://github.com/jinli99/DTGBrepGen>.

1. Introduction

Boundary representation (B-rep) [28] is the predominant format for 3D shape modeling in Computer-Aided Design (CAD). Unlike mesh-based representations, which rely on planar facets and linear edges, B-rep represents objects using parametric surfaces (faces), parametric curves (edges), and vertices to define both topology and geometry. This detailed structure enables B-rep models to be able to accurately capture complex surfaces and solids, making them indispensable in engineering and manufacturing.

However, automatically generating B-reps poses sig-

nificant challenges due to the need to accurately model both topological relationships and geometric representations. Many existing CAD generation methods address this challenge indirectly by generating sequences of modeling commands [16, 32–34], from which the B-rep can be recovered in postprocess using a solid modeling kernel. However, these methods are constrained by limited datasets [30, 32] and are typically confined to basic operations like sketching and extruding, making them suitable only for simpler shapes. Among current B-rep generative models, SolidGen [10] and BrepGen [35] have made strides in direct B-rep generation. However, these approaches predominantly emphasize on geometric attributes while neglecting the automatic generation of topological structures. As a result, SolidGen is restricted to simpler prismatic shapes and lacks support for more complex surfaces, while BrepGen struggles with reliably reconstructing topology from its generated geometric attributes. These limitations highlight the need for more robust approaches capable of handling both topological validity and geometric complexity in B-reps.

For this purpose, we present DTGBrepGen, a novel approach to B-rep generation that decouples topology from geometry, addressing the challenges of modeling both aspects simultaneously. DTGBrepGen first generates a valid topological structure, which defines the connectivity between faces, edges, and vertices. This step is particularly challenging due to strict topological constraints—such as each edge must connect two vertices, and edges on each face must form closed loops. To address this, we design a two-phase topology generation process with Transformer encoder-decoder architectures: the first phase generates edge-face adjacencies, followed by the second phase establishing edge-vertex connections. With this established topology, we employ Transformer-based diffusion models [3, 8, 24] to generate geometric attributes. In contrast to existing methods like UV-Net [9] and BrepGen [35] that rely on discrete point sampling, we employ B-spline representations for curves and surfaces. This enables direct learning of control point distributions, resulting in more accurate and mathematically precise geometric representations. In summary, the contributions of this paper are as follows:

*Corresponding author

- We introduce DTGBrepGen, a novel framework for B-rep generation that explicitly addresses both topology and geometry separately. Our key innovation is a two-phase topology generation strategy designed to effectively handle complex topological constraints.
- We develop an advanced geometric generation pipeline that leverages Transformer-based diffusion models and B-spline representations, resulting in more precise and compact geometric outputs.
- Comprehensive experiments are provided to show that DTGBrepGen surpasses existing approaches in producing valid, diverse, and high-quality B-rep models.

2. Related work

The generation of CAD models has emerged as a significant research area, with various approaches targeting different representation formats. This section reviews some key methodologies for CAD model generation.

2.1. CAD model generation from point clouds

Generating CAD models from point clouds plays a key role in reverse engineering [6, 14, 23]. While traditional methods converting point clouds to meshes before CAD reconstruction [1], recent approaches streamline this by directly predicting CAD structures. For instance, Point2Cyl [23] formulates extrusion cylinder decomposition using neural networks, while ComplexGen [6] directly generates B-rep models by identifying geometric primitives and relationships. Despite recent advances, generating accurate CAD models from point clouds remains challenging due to issues like data incompleteness, high geometric complexity, and the difficulty of inferring precise topological relationships.

2.2. Constructive solid geometry

Constructive Solid Geometry (CSG) creates 3D shapes by combining simple primitives like cubes and spheres using Boolean operations such as union and subtraction. This representation has been extensively used in shape programs and parametric modeling due to its simplicity and interpretability [4, 20, 22]. However, CSG’s reliance on predefined primitives makes it less flexible in capturing more intricate shapes. Furthermore, converting CSG models to B-rep often introduces geometric artifacts such as sliver faces, which complicates downstream tasks.

2.3. CAD command generation

Recent advances in CAD modeling enable sequence generation of CAD commands directly from parametric files, producing editable models. DeepCAD [32] pioneered the modeling of CAD command sequences, laying a foundation for further developments [16, 33, 34]. Despite this progress, these methods are mostly limited to basic operations (*e.g.*, sketch and extrude) and struggle with complex commands

like fillet and chamfer. Additionally, datasets containing CAD operations [30, 32] are smaller (around 190K models) than datasets without operations [13] (around 1M models). While progress has been made, challenges remain in broadening operation ranges and dataset sizes.

2.4. B-rep generation

B-rep models provide CAD representations by defining solids through vertices, edges, and faces, making them essential for capturing complex surface interactions in industrial CAD software. Prior works have explored B-rep tasks such as classification and segmentation [2, 31] and parametric surface generation [21, 27]. PolyGen [18] employs Transformers [24] and pointer networks [26] to generate n-gon meshes, which are simplified instances of B-rep models with planar faces and linear edges. Building on this, recent methods like SolidGen [10] extend to full B-rep structures via sequential generation, starting with vertices and conditionally predicting edges and faces. However, they remain limited to prismatic shapes. In contrast, BrepGen [35] represents B-reps as hierarchical tree structures, leveraging multiple diffusion models to generate geometric attributes, followed by a post-processing step to merge nodes and recover topology. Nevertheless, this method often struggles with accurately reconstructing topological structures. Thus, while these approaches advance B-rep generation, they typically couple topological and geometric processes, making it challenging to produce B-rep models that are both topologically correct and geometrically precise.

3. Generative framework of B-rep structures

In this section, we will first describe the B-rep generation problem, and then outline the basic idea of our approach.

3.1. Problem setup

Let $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_N\}$ denote a set of real B-rep models, where each $\mathcal{B}_i = (\mathcal{T}_i, \mathcal{G}_i)$ is characterized by its topological structure \mathcal{T}_i and its geometric attributes \mathcal{G}_i . A B-rep structure is composed of three fundamental elements: vertices, edges, and faces. The topological structure \mathcal{T}_i defines the connectivity between these elements without specifying their exact positions. It describes how vertices are connected to form edges, and how edges combine to form faces. In contrast, the geometric attributes \mathcal{G}_i provides the precise spatial definitions of each element. Specifically, it includes the 3D coordinates of each vertex, the curve definitions for each edge (such as lines or arcs), which are bounded by two vertices, and the surface definitions for each face (such as planes or spheres), which are enclosed by specific edges. For each \mathcal{B}_i , we denote the number of vertices, edges, and faces as N_v^i , N_e^i , and N_f^i respectively, and the sets of vertices, edges, and faces as \mathbf{V}_i , \mathbf{E}_i , and \mathbf{F}_i .

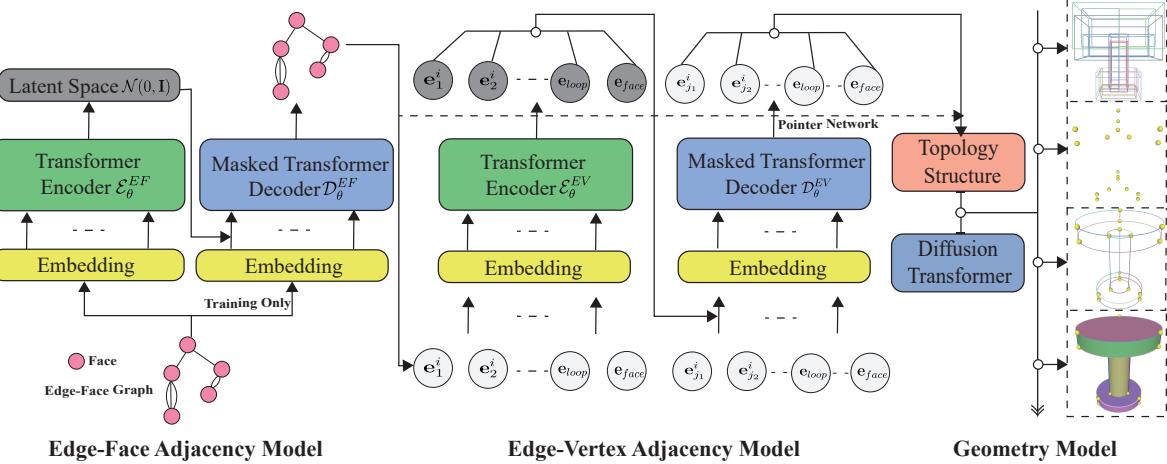


Figure 1. Overview of DTGBrepGen. The pipeline first generates a valid topological structure by two sequentially Transformer encoder-decoder networks: one to produce edge-face adjacencies, followed by another to establish edge-vertex connections. Based on this topology, the geometry generation process employs Transformer-based diffusion models to progressively generate face bounding boxes, vertex coordinates, edge geometries, and finally face geometries.

For simplicity, we limit our scope to closed B-rep models, where each edge is shared by exactly two distinct faces. Additionally, all closed faces and edges are split along seams [10], simplifying processing by ensuring each edge connects two distinct vertices. This preprocessing step allows us to more easily define and work with the topological structure. In particular, a valid topological structure must satisfy the following three conditions:

- C_1 : Each edge must be contained in exactly two distinct faces.
- C_2 : Each edge must connect exactly two distinct vertices.
- C_3 : The edges of each face must form closed loops.

Our goal is to develop a generative model G_θ , parameterized by θ , which generates new B-rep models with valid topologies and accurate geometries from a known distribution (typically standard Gaussian distribution). The generated B-rep models should approximate the underlying distribution of real B-rep models \mathcal{B} . In the context of existing generative frameworks [5, 8, 11], a common approach is maximum likelihood estimation, where we aim to maximize the likelihood $\prod_{i=1}^N P_\theta(\mathcal{B}_i)$, with $P_\theta(\mathcal{B}_i)$ denoting the probability that the model G_θ generates \mathcal{B}_i . This optimization ensures that the generated B-rep models closely match the true distribution of real-world B-rep structures, thereby producing realistic and valid outputs.

3.2. Method overview

Unlike previous approaches (SolidGen and BrepGen), the core of DTGBrepGen is to decouple the generation of topology from geometry, structuring the probability of generating a B-rep model \mathcal{B}_i as:

$$P_\theta(\mathcal{B}_i) = P_\theta(\mathcal{T}_i)P_\theta(\mathcal{G}_i|\mathcal{T}_i), \quad (1)$$

We model each part using separate neural networks.

We first discuss the generation of the topological structure, with the primary challenge being to ensure compliance with the three previously defined constraints. There are six possible adjacency relations between vertices, edges, and faces, derived from their pairwise combinations. Our initial step is to select a subset of these relations that meet the following criteria:

- The relations should uniquely define the complete topological structure.
- The relations should be minimal, meaning none can be inferred from the others.
- The relations should be suitable for learning within a generative model.

The first two criteria ensure that the selected adjacency relations provide a “basis” for defining the topological structure. We found that the edge-face matrix $\mathbf{EF}_i \in \mathbb{N}^{N_e^i \times 2}$ and the edge-vertex matrix $\mathbf{EV}_i \in \mathbb{N}^{N_e^i \times 2}$ are such a basis for \mathcal{T}_i , where \mathbf{EF}_i stores the IDs of the two faces connected by each edge, and \mathbf{EV}_i stores the IDs of the two vertices connected by each edge in the B-rep model \mathcal{B}_i . Notably, when each row of \mathbf{EF}_i and \mathbf{EV}_i contains two distinct elements, constraints C_1 and C_2 are inherently satisfied, simplifying the topology generation process. Additionally, due to the structured representation of \mathbf{EF}_i and \mathbf{EV}_i , they are well-suited for generative model learning. Due to the challenges of generating these matrices simultaneously, we first generate the edge-face matrix \mathbf{EF}_i , followed by the generation of the edge-vertex matrix \mathbf{EV}_i . Both generative networks employ a Transformer encoder-decoder architecture [24], as illustrated in Fig. 1. However, not all configurations of \mathbf{EF}_i and \mathbf{EV}_i yield a valid topology, as they may fail to meet the

constraint C_3 , requiring edges of each face to form closed loops. This constraint is further discussed in Sec. 4.2.

When the topology is established, we sequentially generate geometric attributes: vertex coordinates are generated first, followed by edge geometry, and finally face geometry. At each stage, existing topological information is integrated. For instance, when generating edge geometry, the coordinates of the two connected vertices serve as conditional inputs. Additionally, we introduce a preliminary step for generating face bounding boxes, which significantly enhances the quality of the results. Consequently, our geometry generation architecture includes four diffusion-based generative models, each employing a Transformer encoder as the denoising network [3, 8, 24], as shown in Fig. 1. We represent curves and surfaces using B-splines, enabling direct learning of distributions of the control points. This approach provides precise geometric representations and avoids the need for discrete point sampling [9] or a separate VAE [11] for encoding high-dimensional point clouds [35], both of which can introduce geometric inaccuracies.

4. Decoupling topology and geometry

In this section, we discuss each phase of topology and geometry generation in detail.

4.1. Edge-face adjacency generation

Directly learning the distribution of the edge-face matrices $\{\mathbf{EF}_i\}_{i=1}^N$ is suboptimal, as it stores face IDs which do not inherently reflect the topological structure. To address this, we introduce a more suitable and topologically equivalent representation $\{\mathbf{FeF}_i\}_{i=1}^N$. Here, $\mathbf{FeF}_i \in \mathbb{N}^{N_f^i \times N_f^i}$, with each element $\mathbf{FeF}_i[k, l]$ denoting the number of edges shared between face k and face l in B-rep \mathcal{B}_i . This representation can also be viewed as a graph, as illustrated in Fig. 1, where nodes represent the faces of the B-rep, and edges between nodes indicate the number of shared edges between two faces. Notice that, reassigning face IDs within the same topological structure leads to different matrices \mathbf{FeF}_i , even if they represent the same topology. To standardize, we assign face IDs by sorting faces in ascending order according to their edge count. For faces with identical edge counts, we assign their order randomly, as we found this randomness assignment minimal impact on training performance. Given the symmetry of the \mathbf{FeF}_i matrix, we extract its upper triangular portion and convert it into a sequence:

$$\mathbf{EF}_i^{seq} := \{\mathbf{FeF}_i[1, 2], \mathbf{FeF}_i[1, 3], \dots, \mathbf{FeF}_i[N_f^i - 1, N_f^i]\}, \quad (2)$$

Shared-edges embedding. The sequence \mathbf{EF}_i^{seq} represents the number of shared edges between pairs of faces. To encode this information, we introduce a set of learnable embeddings with a cardinality of $M_e + 1$, where

$$M_e := \max \{\max(\mathbf{FeF}_i) \mid 1 \leq i \leq N\}, \quad (3)$$

which accounts for cases where faces have no shared edges.

Encoder-decoder. We employ a Transformer-based VAE architecture [11, 24] to model the distribution of edge-face sequences $\{\mathbf{EF}_i^{seq}\}_{i=1}^N$. The encoder network \mathcal{E}_θ^{EF} processes the input sequence by combining three types of embeddings: positional embeddings [24], shared-edges embeddings, and face ID embeddings. These combined embeddings are fed into the Transformer encoder to estimate the parameters of the latent distribution [11]. The decoder network \mathcal{D}_θ^{EF} then maps this latent representation to output distributions over the discrete states $\{0, 1, \dots, M_e\}$ at each sequence position, with the reconstructed sequence is sampled from these distributions. During inference, new edge-face sequences can be generated by sampling from the learned latent space and decoding through \mathcal{D}_θ^{EF} .

Loss function. We employ the standard loss function used in VAE training, consisting of two components,

$$\begin{aligned} \mathcal{L}_{EF} = & \frac{1}{N} \sum_{i=1}^N \left(CE(\mathbf{EF}_i^{seq}, \mathcal{D}_\theta^{EF}(\mathcal{E}_\theta^{EF}(\mathbf{EF}_i^{seq}))) \right. \\ & \left. + D_{KL}(\mathcal{E}_\theta^{EF}(\mathbf{EF}_i^{seq}) \parallel \mathcal{N}(0, \mathbf{I})) \right) \end{aligned} \quad (4)$$

where CE represents the cross-entropy loss between the input sequence and its reconstruction, and D_{KL} represents the KL divergence between the learned latent distribution and the standard multivariate Gaussian prior $\mathcal{N}(0, \mathbf{I})$.

4.2. Edge-vertex adjacency generation

After generating the edge-face adjacency, we know which edges are associated with each face, but the connectivity between these edges remains undetermined. The task of generating edge-vertex adjacency involves determining how these edges should be ordered and connected, while adhering to topological constraints (C_1 - C_3). The C_1 constraint is inherently satisfied due to the properties of the $\{\mathbf{FeF}_i\}_{i=1}^N$ matrices. For the C_2 constraint, we enforce that the two endpoints of the same edge must not connect to each other. The most difficulty lies in the C_3 constraint, which requires that the edges of each face form closed loops. Since each edge is shared by two faces, forming loops for one face may compromise the ability to form valid loops for the adjacent face, making this a non-trivial challenge. To address this, we use a Transformer architecture [24] combined with a pointer network [26] for edge-vertex adjacency generation.

Sequential edge-vertex representation. We reformulate edge connectivity generation (*i.e.*, edge-vertex adjacencies) as a sequence generation task. Beginning from the face with the smallest ID, we serialize the edge connections for each face. For model \mathcal{B}_i , the sequence is defined as:

$$\mathbf{EV}_i^{seq} := \{\mathbf{e}_{j_1}^i, \mathbf{e}_{j_2}^i, \dots, \mathbf{e}_{loop}, \mathbf{e}_{face}, \dots, \mathbf{e}_{face}\}, \quad (5)$$

where $\mathbf{e}_{j_k}^i$ denotes the edge at index j_k in \mathcal{B}_i , and \mathbf{e}_{loop} and \mathbf{e}_{face} are placeholders for the end of a loop and face, re-

spectively. This sequence specifies the edge connectivity, e.g., $e_{j_2}^i$ is connected to $e_{j_1}^i$, and $e_{j_3}^i$ is connected to $e_{j_2}^i$. Each edge appears twice in \mathbf{EV}_i^{seq} to account for its occurrence in both connected faces. To ensure sequence uniqueness, we assign a unique ID to each edge by lexicographically sorting the edges based on their connected face IDs. For edges that share the same faces, we randomly assign their order, mirroring the approach used for face ID assignment in Sec. 4.1. After assigning edge IDs, we start with the smallest ID edge for each face and sequentially arrange the remaining connected edges. One further challenge is that each edge has two endpoints, meaning there are four possible configurations for how two edges might connect. To handle this, we duplicate each edge in our experiments, treating both endpoints individually. This ensures the edge connections is unambiguous. Further details on our serialization method are provided in the supplementary materials.

Encoder. The Transformer encoder \mathcal{E}_θ^{EV} processes the embeddings of edges in \mathbf{E}_i along with two special tokens: \mathbf{e}_{loop} and \mathbf{e}_{face} . Each edge embedding comprises three components: aggregated face features, endpoint embeddings, and shared-edge embeddings. For the first component, we use a Graph Convolutional Network (GCN) [12] to extract features from the previously generated edge-face adjacencies, then average the features of each edge’s two adjacent faces to form part of the edge embedding. For the endpoint embeddings, we introduce two distinct embeddings to represent each endpoint of the edge. Additionally, the shared-edges embeddings help to distinguish edges sharing the same faces as discussed in Sec. 4.1. These three embeddings are summed to form the final edge embedding, which is then fed into the Transformer encoder, producing $2N_e^i + 2$ (accounting for edge duplication) contextual embeddings.

Pointer decoder. Our decoder \mathcal{D}_θ^{EV} is designed to generate an edge-connection sequence based on the contextual embeddings from the encoder. We map each index in \mathbf{EV}_i^{seq} to its corresponding contextual embedding, creating an aligned sequence that preserves the edge connection order. This sequence, enhanced with positional embeddings, is then processed by the Transformer decoder. Finally, a pointer network [18, 26] is applied to produce a probability distribution over possible connections, predicting the most likely edge connection at each position.

Loss function. We employ cross-entropy loss between the predicted token probability distribution from the decoder and the ground truth sequence:

$$\mathcal{L}_{EV} = \frac{1}{N} \sum_{i=1}^N CE(\mathbf{EV}_i^{seq}, \mathcal{D}_\theta^{EV}(\mathcal{E}_\theta^{EV}(\mathbf{E}_i), \mathbf{EV}_i^{seq})), \quad (6)$$

Inference. During inference, tokens are generated autoregressively with three key aspects: 1) Since each edge is shared by two faces, once the edge connections for the ini-

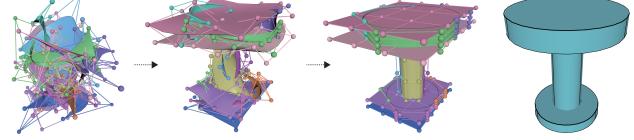


Figure 2. Illustration of B-spline control point generation via diffusion models. Our method learns the distribution of control points to establish face geometry in B-rep models, enabling precise geometric representation.

tial faces are determined, the connections for subsequent faces are partially fixed. Thus, we must identify these already-determined edges before processing each face. 2) When determining the next token, we first sample a candidate based on the predicted probability, then check it against topological constraints. If the candidate violates any constraints, it is discarded, and a new candidate is sampled from the remaining options. 3) Although topological constraints are not strictly enforced during training, experiments show high topological accuracy (see Sec. 5.2), indicating the model effectively captures underlying structures.

4.3. Geometry generation

As outlined in Sec. 3.2, our approach sequentially generates the face bounding boxes, vertex coordinates, edge geometries, and face geometries using Transformer-based diffusion models. We illustrate this process using face bounding boxes as an example. Let $\mathbf{FB}_i \in \mathbb{R}^{N_f^i \times 6}$ denote the bounding boxes representation for each face in \mathcal{B}_i . Following the standard diffusion process, we add noise to the $\{\mathbf{FB}_i\}_{i=1}^N$ tokens and use a Transformer-based denoiser to predict this noise. A key feature of our approach is the integration of known topological structures. Specifically, when calculating attention scores between faces, we incorporate the number of shared edges (i.e., $\{\mathbf{FeF}_i\}_{i=1}^N$), enhancing the model’s capacity to capture face-to-face interactions aligned with underlying topology [25, 36]. For vertex geometry generation $\mathbf{V}_i \in \mathbb{R}^{N_v^i \times 3}$, we include information on whether two vertices are connected by an edge, which informs the attention between vertices. Each edge is represented as a cubic B-spline curve with four control points, parameterized as $\mathbf{E}_i \in \mathbb{R}^{N_e^i \times 12}$ by these control points’ coordinates. For face geometry, we apply a similar method: each face is represented as a bi-cubic B-spline surface with a 4×4 grid of control points, resulting in face geometry $\mathbf{F}_i \in \mathbb{R}^{N_f^i \times 48}$ parameterized by the coordinates of these sixteen control points. Our diffusion models directly learn the distribution of these control points to capture precise geometric features. Fig. 2 illustrates an example of the generation process for face control points.

Post-processing. Since many surfaces correspond to basic types like planes or quadratic surfaces, when con-

structing the face geometry, we first attempt to fit the face’s boundary (discrete points sampled from the generated edge geometries) and interior (discrete points sampled from the generated face geometries) using basic primitives [17]. If the fitting error falls below a predefined threshold, we adopt the fitted primitives; otherwise, we resort to using the generated B-spline surface (*i.e.*, generated face geometries). Finally, we use OpenCascade [15] functions to seamlessly sew the generated topology and geometries into a coherent B-rep solid.

5. Experiments

5.1. Experiment setup

Datasets. We conduct experiments using the DeepCAD [32], ABC [13], and Furniture [35] datasets. B-reps with more than 50 faces or faces with over 30 edges are excluded. Following the filtering methods in [10, 32, 35], our final training dataset consists of 80,509 DeepCAD B-reps, 198,522 ABC B-reps, and 1,065 Furniture B-reps.

Network architecture. For edge-face adjacency generation, we employ a 4-layer Transformer encoder-decoder architecture with 128-dimensional embeddings and 4 attention heads. The edge-vertex adjacency model follows a similar architecture but uses 256-dimensional embeddings. Both models utilize masked Transformer decoders and are trained using the teacher-forcing method [29], where the ground-truth adjacency sequence is used as input to the decoder. During inference, tokens are generated in an autoregressive manner. For geometry generation, we employ four diffusion models, each featuring an 8-layer Transformer-based denoiser with an embedding dimension of 512 and 8 attention heads. Additional details on architecture and training are in the supplementary materials.

Evaluation metrics. Following [35], we use Distribution and CAD Metrics for a comprehensive evaluation. Distribution Metrics (Coverage (COV), Minimum Matching Distance (MMD), and Jensen-Shannon Divergence (JSD)) measure similarity between generated and ground-truth distributions, with COV and MMD are computed using Chamfer Distance (CD) in our experiment. CAD Metrics assess model quality and diversity, including Novel (models not in training set), Unique (models appearing only once), and Valid (watertight solid B-reps). Novel and Unique metrics are computed using the hashing procedure from [10], while our validity criteria require watertight, non-manifold structures with correct topology and geometry.

5.2. Unconditional generation

Topology generation evaluation. We evaluate the performance of our topology generation model using three metrics: Novel, Unique, and Valid. These metrics follow CAD Metrics but are specifically applied to topology structure.

Datasets	Novel (%) ↑	Unique (%) ↑	Valid (%) ↑
DeepCAD	85.01	82.27	92.10
ABC	80.55	78.34	89.20
Furniture	81.63	80.56	88.10

Table 1. Quantitative evaluation of topology generation on DeepCAD, ABC, and Furniture datasets.

To assess its performance, we generate 1,000 topologies using our topology generation network, which includes both edge-face and edge-vertex models. A topology is considered valid if it satisfies all the constraints (C_1 , C_2 and C_3). Additionally, two topologies are considered equivalent if they share the same structural relationships among vertices, edges, and faces. As shown in Tab. 1, DTGBrepGen achieves high novelty, uniqueness and validity rates across all the datasets, demonstrating its ability to generate diverse topological structures while maintaining validity and effectively capturing underlying patterns in B-rep topology generation.

Comparison with baseline methods. We compare our approach with two representative B-rep generation methods: DeepCAD [32] and the state-of-the-art BrepGen [35]. For DeepCAD, we evaluate B-rep models reconstructed from generated sketches and extrusion sequences. We randomly sample 3,000 generated and 1,000 reference B-rep models. For each model, we sample 2,000 points from its surface to compute the Distribution Metrics. The CAD Metrics are calculated using the generated 3,000 B-rep models. As shown in Tab. 2, DTGBrepGen achieves better performance across almost all the metrics, with particularly substantial improvements in the Valid metric, demonstrating our method’s superior capability in generating well-formed and watertight B-rep models. The qualitative comparison in Fig. 3 further illustrates our DTGBrepGen’s ability to generate more realistic and geometrically precise B-rep models compared to baseline approaches.

5.3. Conditional generation

Our framework supports conditional generation by incorporating additional contextual information. We demonstrate this capability through two distinct tasks: class-conditioned generation and point cloud-conditioned generation.

Class-conditioned generation. We evaluate class-conditioned generation on the Furniture B-rep dataset (10 categories), by assigning each category a learnable embedding added to the input tokens and using classifier-free guidance [7] for controlled generation. As shown in Fig. 4, our method effectively captures category-specific features while maintaining structural diversity. For quantitative assessment, we compare our approach to BrepGen. Given data limitations in the Furniture dataset, we focus Distri-

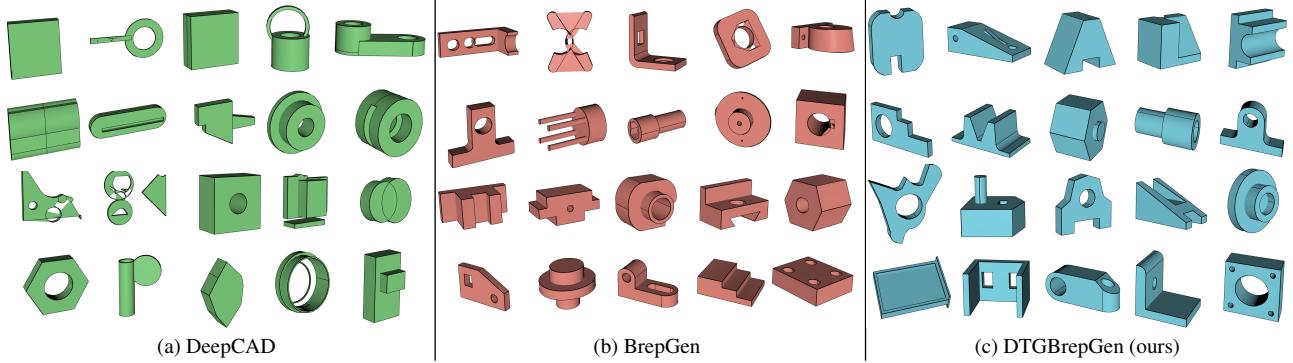


Figure 3. Qualitative comparison of B-rep models generated by our method, DeepCAD [32], and BrepGen [35] on the DeepCAD dataset.

Method	DeepCAD					
	COV↑	MMD↓	JSD↓	Novel↑	Unique↑	Valid↑
DeepCAD	70.81	1.31	1.79	93.80	89.79	58.10
BrepGen	72.38	1.13	1.29	99.72	99.18	68.23
Ours	74.52	1.07	1.02	99.79	98.94	79.80
Ours*	72.86	1.18	1.09	99.59	99.02	71.77

Method	ABC					
	COV↑	MMD↓	JSD↓	Novel↑	Unique↑	Valid↑
BrepGen	64.29	1.53	1.86	99.68	99.05	47.11
Ours	71.13	1.33	1.59	99.73	99.12	62.08
Ours*	69.60	1.37	1.65	99.56	98.97	56.63

Table 2. Quantitative comparison with baseline methods on DeepCAD and ABC datasets using Distribution Metrics (COV, MMD, JSD) and CAD Metrics (Novel, Unique, Valid). Note that MMD and JSD values are multiplied by 10^2 , while COV, Novel, Unique, and Valid are expressed as percentages. Ours* corresponds to an ablation variant of DTGBrepGen where B-spline representations are replaced with point cloud-based latent encoding.

bution Metrics on the four largest categories (table, chair, bed, sofa). As seen in Tab. 3, our method consistently outperforms BrepGen across all the metrics in these categories. We also evaluate generation quality with CAD Metrics, reporting only the Valid metric due to consistently high Novel and Unique scores (98 – 100%). In Tab. 4, our approach achieves approximately 10% higher Valid scores across all categories compared to BrepGen. Both methods show lower validity rates in certain categories, such as Lamp, likely due to limited training samples. Overall, these results emphasize DTGBrepGen’s effectiveness in generating structurally valid, category-specific furniture models.

Point cloud-conditioned generation. We adapt our Transformer for point cloud-conditioned generation by integrating a PointNet++ [19] that encodes 2,000 sampled points into a 512-dimensional embedding. Instead of using cross-attention, this embedding is added directly to each

Method	Table			Chair		
	COV↑	MMD↓	JSD↓	COV↑	MMD↓	JSD↓
BrepGen	65.08	0.97	4.30	61.24	0.91	3.50
Ours	70.20	0.63	2.69	69.48	0.67	2.36

Method	Bed			Sofa		
	COV↑	MMD↓	JSD↓	COV↑	MMD↓	JSD↓
BrepGen	59.30	0.75	3.66	67.82	0.45	1.86
Ours	71.11	0.58	2.47	73.46	0.42	1.75

Table 3. Comparison of Distribution Metrics across the four most populous categories in the Furniture dataset for class-conditioned generation. Note that MMD and JSD values are multiplied by 10^2 .

Method	Bathtub	Bed	Bench	Bookshelf	Cabinet
BrepGen	28.50	57.63	66.03	33.53	34.57
Ours	49.82	67.87	69.44	50.82	69.16
Method	Chair	Couch	Lamp	Sofa	Table
BrepGen	72.45	70.54	29.53	76.25	55.67
Ours	80.17	73.45	36.62	79.93	68.27

Table 4. Comparison of Valid metric (%) between our method and BrepGen across all categories in the Furniture dataset.

token before being input to the Transformer, allowing the network to integrate point cloud information without altering the core Transformer architecture. As shown in Fig. 5, our model generates diverse yet coherent B-reps that faithfully preserve the input geometry’s characteristics. These results validate the effectiveness of our approach in generating meaningful variations while preserving the essential features of the conditioning data. While effective in most cases, we observe that for point clouds with highly intricate geometric details, generation quality can be affected, indicating potential for future work on improving the encoding of fine-grained geometric features in complex point clouds.

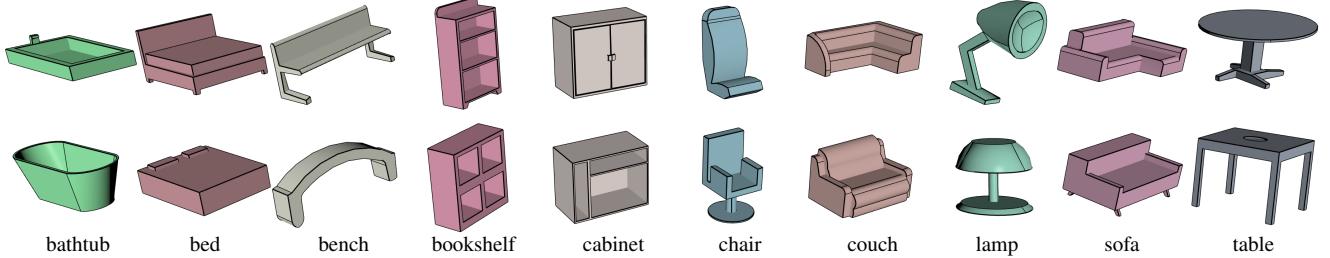


Figure 4. Examples of class-conditioned generation results for different furniture categories. Each column shows two instances from the same category, demonstrating our method’s ability to capture category-specific features while ensuring structural diversity.

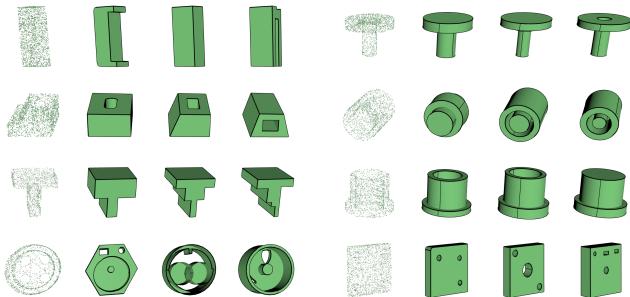


Figure 5. B-rep models generated based on input point clouds. Each example consists of a point cloud alongside three corresponding generated B-reps.

5.4. Discussion

Ablation study. To assess the impact of using B-spline parametrization for edges and faces, we conduct an ablation study by replacing the B-spline geometric representation with discrete points [9, 10, 35]. Specifically, within the context of edge and face generation, we substitute the B-spline representation with a latent representation obtained from a VAE-based point cloud encoder. As shown in Tab. 2, our original approach, which employs B-spline representations, outperforms the discrete point-based variant across all the metrics. These results underscore the effectiveness of directly learning control point distributions, which enables our model to capture the underlying geometric structure more accurately and effectively. Qualitative comparison results are available in the supplementary materials.

Limitations and future work. While DTGBrepGen demonstrates significant improvements over existing methods, several limitations remain to be addressed. First, although we achieve higher B-rep validity rates, the overall validity still leaves room for improvement. This limitation mainly arises from a gap between the learned and true topology distributions. As shown in Fig. 6a, geometry generation occasionally fails even with valid sampled topologies. This phenomenon indicates that while our topology network can reconstruct topological structures with high accuracy (see

Datasets	DeepCAD	ABC	Furniture
Valid (%) ↑	90.24	84.61	82.59

Table 5. B-rep validity rates for geometry generation using ground-truth topologies.



Figure 6. Failure cases. (a) A valid topology sampled from our topology network that leads to unsuccessful geometry generation across multiple attempts. (b) Instances where our method produces non-watertight solids and self-intersecting geometries.

Tab. 1), not all valid topologies are necessarily suitable for geometry generation. This observation is further supported by our experiments using ground-truth topologies, which yield a higher B-rep validity rate (see Tab. 5). Additionally, similar to existing methods, DTGBrepGen faces challenges when generating complex models, particularly those with intricate geometric details and complicated structural relationships. Future work could focus on advanced topology validation mechanisms, improved designs to bridge topology and geometry generation, and hierarchical strategies to better handle complex models.

6. Conclusion

In this paper, we presented DTGBrepGen, a framework for B-rep generation that decouples topology and geometry, addressing key limitations in existing methods. By separately modeling topological and geometric attributes, DTGBrepGen enhances the validity and diversity of generated CAD models, offering a promising tool for CAD design exploration and synthesis. We believe this decoupled methodology contributes a valuable perspective to the field, potentially supporting future advances in automatic B-rep generation.

References

- [1] Pál Benkő and Tamás Várady. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design*, 36(6):511–523, 2004. 2
- [2] Weijuan Cao, Trevor Robinson, Yang Hua, Flavien Bousuge, Andrew R Colligan, and Wanbin Pan. Graph representation of 3d cad models for machining feature recognition with deep learning. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, page V11AT11A003. American Society of Mechanical Engineers, 2020. 2
- [3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 1, 4
- [4] Kevin Ellis, Maxwell Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama. Write, execute, assess: Program synthesis with a repl. *Advances in Neural Information Processing Systems*, 32, 2019. 2
- [5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014. 3
- [6] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022. 2
- [7] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 6
- [8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 3, 4
- [9] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11703–11712, 2021. 1, 4, 8
- [10] Pradeep Kumar Jayaraman, Joseph George Lambourne, Nishkrit Desai, Karl Willis, Aditya Sanghi, and Nigel J. W. Morris. Solidgen: An autoregressive model for direct b-rep synthesis. *Transactions on Machine Learning Research*, 2023. 1, 2, 3, 6, 8
- [11] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings*, 2014. 3, 4
- [12] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*, 2017. 5
- [13] 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. 2, 6
- [14] Joseph George Lambourne, Karl Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. Reconstructing editable prismatic cad from rounded voxel models. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 2
- [15] Trevor Laughlin. pyocct – python bindings for opencascade via pybind11, 2020. <https://github.com/trelau/pyOCCT>. 6
- [16] Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16826, 2023. 1, 2
- [17] Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. Point2cad: Reverse engineering cad models from 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3763–3772, 2024. 6
- [18] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning*, pages 7220–7229. PMLR, 2020. 2, 5
- [19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017. 7
- [20] Daniel Ritchie, Paul Guerrero, R Kenny Jones, Niloy J Mitra, Adriana Schulz, Karl DD Willis, and Jiajun Wu. Neurosymbolic models for computer graphics. In *Computer Graphics Forum*, pages 545–568. Wiley Online Library, 2023. 2
- [21] Dmitriy Smirnov, Mikhail Bessmel'tsev, and Justin Solomon. Learning manifold patch-based representations of man-made shapes. In *International Conference on Learning Representations*, 2021. 2
- [22] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Learning to infer and execute 3d shape programs. In *International Conference on Learning Representations*, 2019. 2
- [23] Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11850–11860, 2022. 2
- [24] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1, 2, 3, 4
- [25] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023. 5
- [26] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in Neural Information Processing Systems*, 28, 2015. 2, 4, 5
- [27] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. *Advances in Neural Information Processing Systems*, 33:20167–20178, 2020. 2

- [28] Kevin J Weiler. *Topological structures for geometric modeling (Boundary representation, manifold, radial edge structure)*. Rensselaer Polytechnic Institute, 1986. [1](#)
- [29] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. [6](#)
- [30] Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lamourne, 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. [1, 2](#)
- [31] Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lamourne, Armando Solar-Lezama, et al. Joinable: Learning bottom-up assembly of parametric cad joints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15849–15860, 2022. [2](#)
- [32] 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, 6, 7](#)
- [33] Xiang Xu, Karl DD Willis, Joseph G Lamourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. In *International Conference on Machine Learning*, pages 24698–24724. PMLR, 2022. [2](#)
- [34] Xiang Xu, Pradeep Kumar Jayaraman, Joseph G. Lamourne, Karl D.D. Willis, and Yasutaka Furukawa. Hierarchical neural coding for controllable cad model generation. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. [1, 2](#)
- [35] Xiang Xu, Joseph Lamourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024. [1, 2, 4, 6, 7, 8](#)
- [36] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021. [5](#)